Classification non-supervisée à l'aide d'ensembles dominants

Nicolas Seinturier - Nicolas Sivignon

M1 - Mathématiques Appliquées — Dauphine-PSL

Contents

1 Introduction

1.1 vue d'ensemble

Ce mémoire porte sur l'analyse et l'implémentation des méthodes proposées par le papier suivant : *A New Graph-Theoretic Approach to Clustering and Segmentation* publié par Massimiliano Pavan et Marcello Pelillo à l'occasion de la Conference on Computer Vision and Pattern Recognition en 2003 [1].

Il s'agit d'une méthode issue de la théorie des graphes qui généralise la notion de clique maximale aux graphes pondérés. Elle est utilisée pour trouver un sous-ensemble compact, cohérent et bien séparé de sommets dans un graphe, c'est-à-dire un ensemble dominant (Dominant Set).

Cette méthode établit un lien entre les notions de clique pondérée, ensemble dominant et cluster tout en fournissant un algorithme qui permet d'extraire les ensembles dominants d'un graphe. La notion d'ensemble dominant formalise le concept de cluster, à l'aide de deux propriétés cruciales possédée par toutes les techniques de clustering : l'homogénéité intra-cluster et l'hétérogénéité inter-cluster.

1.2 Exposé synthétique

Le jeu de données peut-être représenté par un graphe pondéré non-orienté et sans-boucle $G = (V, E, \omega)$. L'ensemble des sommets V correspond aux données et l'ensemble des arrêtes $E \subseteq V \times V$ représente les relations deux à deux entre les sommets, quantifiées par la fonction $\omega : E \to \mathbb{R}^+$. Un graphe est ainsi représenté par sa matrice d'adjacence A, qui est symétrique.

Soit $S \subseteq V$, on note A_S la matrice d'adjacence liée au graphe formé uniquement par les sommets de S.
Soit $\mathbf{x} \in \mathbb{R}^n$, on définit le support de $\mathbf{x}$ comme l'ensemble des indices pour lesquels ses composantes sont non nulles, c'est-à-dire :

$$\sigma(\mathbf{x}) = \{i \in V : x_i \neq 0\}$$

Trouver un ensemble dominant dans le graphe revient à résoudre le problème d'optimisation suivant :

$$\min_{\mathbf{x} \in \Delta} -\frac{1}{2}\mathbf{x}^T A\mathbf{x} \tag{1}$$

où

$$\Delta = \{\mathbf{x} \in \mathbb{R}^n : \forall i, x_i \geq 0 \text{ et } \sum_i x_i = 1\}$$

est le simplexe de $\mathbb{R}^n$

Soit $\mathbf{x}$ une solution du problème (1). L'ensemble $\sigma(\mathbf{x})$ correspond à un ensemble dominant et donc un cluster. On retire les sommets associés au support de $\mathbf{x}$ et on recommence l'opération pour $A_{V \setminus \sigma(x)}$. On réitère ainsi le processus jusqu'à ce que tous les sommets soient associés à un cluster. En pratique, pour éviter la formation de petits clusters sans signification, nous répétons le processus jusqu'à ce que 90 % des pixels soient associés à un cluster. Les pixels restants étant affectés aux clusters les plus proches.

Un maximum local du problème (1) peut-être trouvé grâce aux équations de réplication dynamique :

$$x(t+1) = x_i(t)\frac{(A\mathbf{x}(t))_i}{\mathbf{x}^T A\mathbf{x}(t)} \tag{2}$$

Le fonctionnement de ces équations ainsi que d'autres méthodes permettant de résoudre (1) seront détaillées dans la partie I.C) du mémoire.

2 Rappels

2.1 Notions de base sur les graphes

Mathématiquement, un graphe G est une structure composée :

(i) d'un ensemble de sommets (également appelés noeuds) qu'on note V;

(ii) d'un ensemble d'arêtes reliant certains sommets entre eux qu'on note $E \subseteq \binom{V}{2}$ où $\binom{V}{2}$ correspond à tous les sommets pris deux-à-deux.

On note $G = (V, E)$.

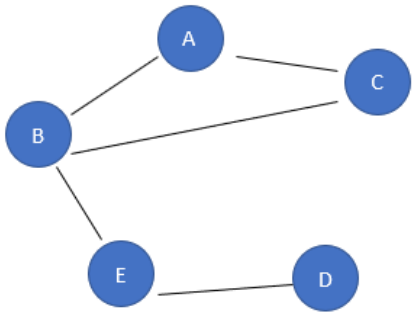
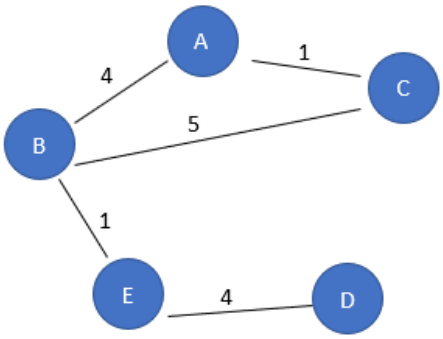


Figure 1

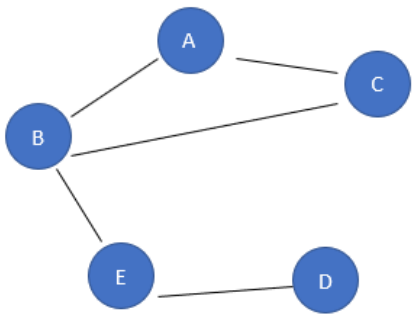
Dans l'exemple ci-dessus, l'ensemble des sommets du graphe est $V = \{A, B, C, D, E\}$ et l'ensemble de ses arêtes est $E = \{(A, B), (A, C), (B, A), (B, C), (B, E), (C, A), (C, B), (D, E), (E, B), (E, D)\}$

2.1.1 Graphe pondéré

Un graphe pondéré est un graphe où chaque arête est affectée d'un poids, qui est en fait un nombre réel positif.



(a) graphe pondéré



(b) graphe non pondéré

Figure 2

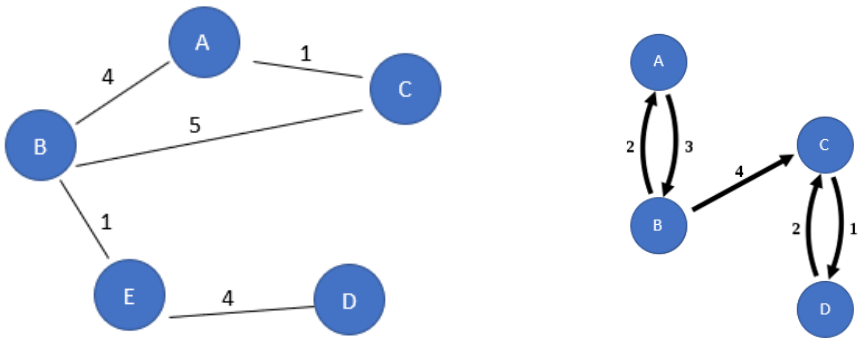
Dans cet exemple, le graphe de la figure 2-(b) n'a pas de poids sur ses arêtes et n'est donc pas pondéré.

On peut cependant considérer tous les graphes non pondérés comme des graphes pondérés en assignant un poids de 1 à chacune de leurs arêtes (toutes les arêtes ont le même poids).

Dans la suite nous nous concentrerons sur les graphes pondérés et pour un graphe G on notera $G = (V, E, w)$ où V est l'ensemble des sommets de G, $E \subseteq V$ x V l'ensemble de ses arêtes et $w : E \mapsto \mathbb{R}^{+*}$ est la fonction de poids qui à chaque arête du graphe, associe son poids. On notera $|V| = N$ et $w(i, j) = w_{i,j}$.

2.1.2 Orientation d'un graphe

Les graphes non orientés sont des graphes dans lesquels les arêtes ne possèdent pas d'orientation: Si une arête va du sommet A au sommet B alors cette même arête va également du sommet B au somme A, contrairement aux graphes orientés où les arêtes ne vont que dans une seule direction.

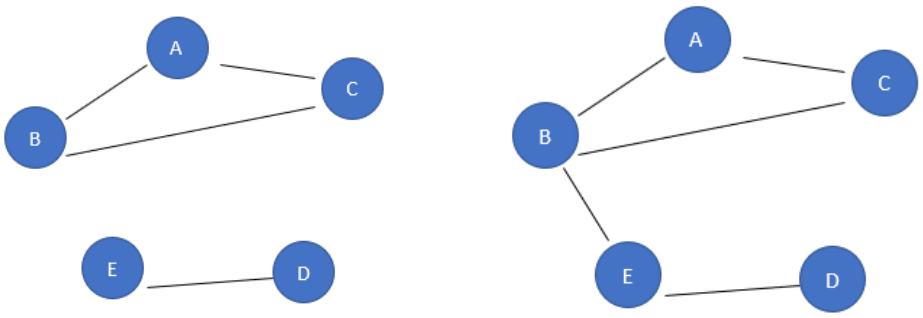


(a) graphe non orienté (b) graphe orienté

Figure 3

2.1.3 Connexité d'un graphe

Un graphe non orienté est *connexe* si pour chaque sommet du graphe i il existe au moins un chemin reliant i à chacun des autres sommets du graphe.



(a) graphe non connexe (b) graphe connexe

Figure 4

Ici, le graphe (e) n'est pas connexe, en effet il n'existe aucun chemin reliant le sommet E au sommet A. En revanche, le graphe (f) est connexe, le sommet E est par exemple relié à A par les arêtes (E,B) et (B,A).

2.1.4 Matrice d'adjacence

La *matrice d'adjacence du graphe* G est la matrice A de taille N x N telle que

$$a_{i,j} = w_{i,j}$$

avec la convention $w_{i,j} = 0$ si les sommets i et j ne sont pas reliés entre eux.

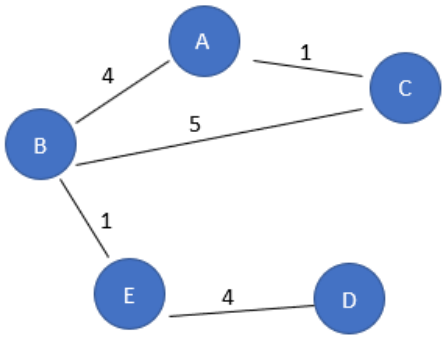


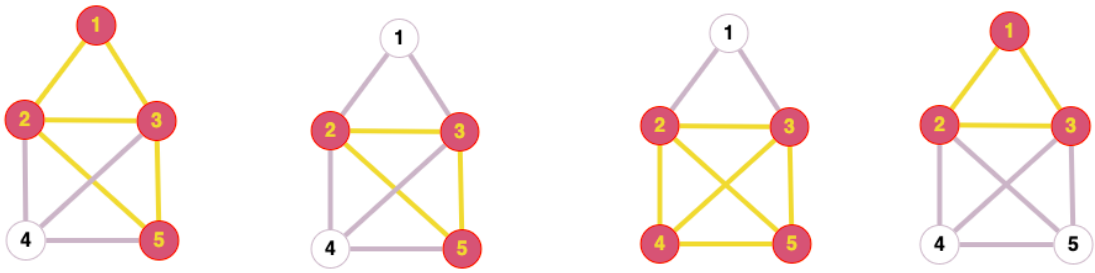
Figure 5

La matrice d'adjacence A du graphe ci-dessus est: $A = \begin{pmatrix} 0 & 4 & 1 & 0 & 0 \\ 4 & 0 & 5 & 0 & 1 \\ 1 & 5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 4 \\ 0 & 1 & 0 & 4 & 0 \end{pmatrix}$

Remarquons que la matrice d'adjacence est ici symétrique. Cela découle du fait que les graphes ne sont pas orientés et que les arêtes sont donc bidirectionnelles.

2.2 Clique

Une clique C d'un graphe G est un sous ensemble de sommets deux-à-deux adjacents. Une clique C est dite *maximale* si il n'existe aucun sommet appartenant à $G \backslash \{C\}$ qui soit adjacent à tous les sommets de C. L'exemple suivant permet d'illustrer cette notion :



1 - pas une clique 2 - pas une clique maximale 3 - clique maximale 4 - clique maximale

Figure 6

Figure 6-1 : le sous ensemble $\{1, 2, 3, 5\}$ n'est pas une clique car aucune arête ne relie les sommets 1 et 5.
Figure 6-2 : le sous ensemble $\{2, 3, 5\}$ est une clique. Cette clique n'est pas maximale car le sommet 4 n'appartient pas à l'ensemble mais est adjacent à tous ses sommets.
Figure 6-3 : le sous ensemble $\{2, 3, 4, 5\}$ correspond à une clique maximale. Tous les sommets sont reliés entre eux et le sommet 1 n'est pas relié à tous les autres de la clique.
Figure 6-3 : le sous ensemble $\{1, 2, 3\}$ correspond à une clique maximale. Tous les sommets sont reliés entre eux et les sommets 4 et 5 ne sont pas reliés au sommet 1.

Une *clique maximum* est une clique qui possède le plus grand cardinal. Une clique maximum est ainsi également maximale, mais l'inverse n'est pas forcément vrai. Une clique maximum n'est pas nécessairement unique (il peut exister plusieurs cliques de taille maximale).
La taille maximale d'une clique dans G est appelée le *nombre de clique* (de G) et est notée $\omega(G)$.

Trouver le nombre de cliques d'un graphe est un problème classique de l'optimisation combinatoire. Il s'agit d'un problème *NP-hard* pour un graphe quelconque [3].

Soit le simplexe sur $\mathbb{R}^n$:

$$\Delta = \{\mathbf{x} \in \mathbb{R}^n : \forall i, x_i \geq 0 \text{ et } \sum_i x_i = 1\}$$

et le *Lagrangien* du graphe G définie par la formule suivante :

$$f_G(\mathbf{x}) = \sum_{\{i,j\} \in E} x_i x_j$$

En 1965, Motzkin et Straus [4] ont établi un lien entre le maximum du Lagrangien d'un graphe sur Δ et son nombre de cliques :
Si $f_G(\mathbf{x}^*)$ est un maximum de f_G dans Δ, alors :

$$\omega(G) = \frac{1}{1 - 2f_G(\mathbf{x}^*)}$$

Ces auteurs ont également montré qu'un sous-ensemble $S \subseteq G$ est une clique maximum si et seulement si son *vecteur caractéristique* (non pondéré) $\mathbf{x}^{S,u} \in \Delta$ définit par :

$$x_i^{S,u} = \begin{cases} \frac{1}{|S|} & \text{si } i \in S \\ 0 & \text{sinon.} \end{cases}$$

est un maximum global de f_G sur Δ [ref].

Soit un graphe pondéré non orienté $G = (V, E)$ avec une fonction de poids positive $w : E \to \mathbb{R}_+^*$, et de matrice d'adjacence A. On introduit le *Lagrangien pondéré* de G définit par :

$$f_G(\mathbf{x}) = \sum_{\{i,j\} \in E} w(i,j) x_i x_j = \mathbf{x}^T A \mathbf{x}$$

La suite de ce mémoire présente une extension du théorème de Motzkin-Straus pour les graphes pondérés non orientés, où la fonction objectif à étudier par rapport à la structure des sous-ensembles correspond au Lagrangien pondéré, i.e. : $\mathbf{x}^T A \mathbf{x}$.

3 Ensembles dominants

3.1 Concept des poids

Dans toute cette partie nous considèrerons le graphe $G = (V, E, \omega)$ de matrice d'adjacence A symétrique à coefficients positifs. On note $(a_{ij})_{\substack{1 \leq i \leq n \\ 1 \leq j \leq n}}$ ses coefficients.

Définitions
Pour $i \in S$, on définit son *degré de poids moyen par rapport à* S par:

$$\text{awdeg}_S(i) = \frac{1}{|S|} \sum_{j \in S} a_{ij}$$

Et on pose:

$$\phi_S(i,j) = a_{ij} - \text{awdeg}_S(i), \, \forall j \notin S$$

On remarquera que $\phi_S(i,j)$ peut-être positif ou négatif.

Le *poids d'un sommet* $i \in S$ est défini récursivement par:

$$w_S(i) = \begin{cases} 1 & \text{si } S = \{i\} \\ \sum_{j \in S \setminus \{i\}} \phi_{S\setminus\{i\}}(j,i) w_{S\setminus\{i\}}(j) & \text{sinon.} \end{cases}$$

Et le *poids total* du sous-ensemble S vaut alors:

$$W(S) = \sum_{i \in S} w_S(i)$$

Remarque
Si S est une paire, i.e. si $S = \{i, j\}$, alors on a:

$$\begin{aligned} w_S(i) &= \sum_{j \in S\setminus\{i\}} \phi_{S\setminus\{i\}}(j,i) w_{S\setminus\{i\}}(j) \\ &= \phi_j(j,i) w_j(j) \end{aligned}$$

Par définition, $w_j(j) = 1$ et on a:
$\phi_j(j,i) = a_{ij} - \text{awdeg}_j(j)(i \notin \{j\})$, Or $\text{awdeg}_j(j) = \frac{1}{1}\sum_{j \in \{j\}} a_{jj} = 0$, D'où $w_S(i) = a_{ij}$ si $S = \{i, j\}$

Exemple
Soit le graphe suivant:

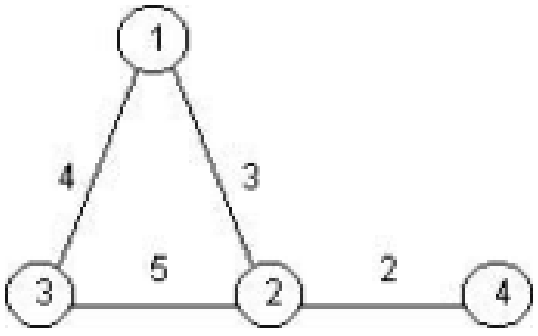


Figure 7

qui admet pour matrice d'adjacence : $\begin{pmatrix} 0 & 3 & 4 & 0 \\ 3 & 0 & 5 & 2 \\ 4 & 5 & 0 & 0 \\ 0 & 2 & 0 & 0 \end{pmatrix}$

On a que

$$\phi_{1,2}(1,3) = a_{1,3} - \text{awdeg}_{1,2}(1) = 4 - \frac{0+3}{2} = \frac{5}{2}$$

Et donc :

$$\begin{aligned} w_{1,2,3}(3) &= \phi_{1,2}(1,3)w_{1,2}(1) + \phi_{1,2}(2,3)w_{1,2}(2) \\ &= [4-(0+3)/2]3 + [5-(0+3)/2]3 \\ &= 18 \end{aligned}$$

Interprétation intuitive

Tout d'abord, soit un sous ensemble de sommets S, $\text{awdeg}_S(i), i \in S$ peut être considéré comme une mesure de la similarité moyenne entre le sommet i et ses voisins dans S. De façon similaire, si $j \notin S$, la valeur de $\phi_S(i,j)$ peut être vue comme une mesure de la similarité entre les sommets i et j, par rapport à la similarité moyenne entre le sommet i et ses voisins dans S.
La quantité $w_S(i)$ peut alors être interprétée comme une mesure de similarité globale entre le sommet i et les sommets de $S\backslash\{i\}$, tout en prenant en compte la similarité globale des sommets de $S\backslash\{i\}$ entre eux. Afin de donner une intuition, considérons l'exemple suivant :

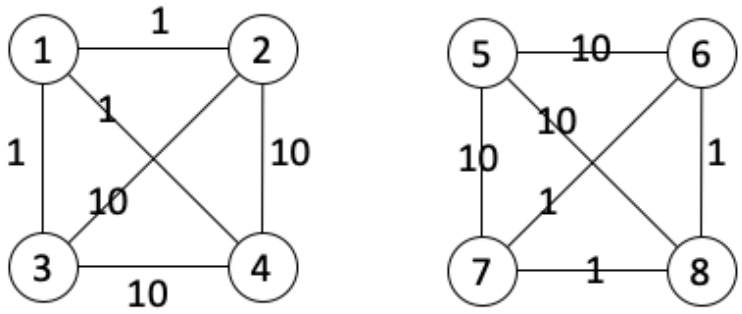


Figure 8

En regardant le graphe de la figure 8, il vient que $w_{\{1,2,3,4\}}(1) < 0$ et $w_{\{5,6,7,8\}}(5) > 0$. Cette relation peut-être comprise intuitivement en regardant les poids des arêtes associées aux sommets 1 et 5 : Les poids des arêtes partant du sommet 1 sont tous plus petit que ceux du sous-ensemble de l'ensemble $\{2,3,4\}$. De la même façon, ceux associés au sommet 5 sont significativement plus élevés que ceux du sous-ensemble $\{6,7,8\}$

N.B: tous les calculs de poids on été fait grâce au code Matlab disponible sur Github et peuvent être reproduits.

3.2 Calcul pratique des poids

Définitions

Soit $S \subseteq V$, on note A_S la matrice d'adjacence liée au graphe formé uniquement par les sommets de S :
$A_S = (a_{ij})_{\substack{i \in S \\ j \in S}}$

On définit la matrice B_S comme suit :
$B_S = \begin{pmatrix} 0 & \mathbf{e}^T \\ \mathbf{e} & A_S \end{pmatrix}$
où $\mathbf{e}$ correspond au vecteur constitué de 1 de la taille apropriée.
Soit $S = \{i_1, ..., i_m\}$ avec $i_1 < ... < i_m$, on définit la matrice ${}^jB_S$ comme suit :

$${}^jB_S = \begin{pmatrix} 0 & & & & \mathbf{e}^T & & & \\ \mathbf{e} & A_S^1 & ... & A_S^{j-1} & 0 & A_S^{j+1} & ... & A_S^m \end{pmatrix}$$

où A_S^j correspond à la jème colonne de A_S

Lemme 1
Soit $S \subseteq V$ un ensemble non vide tel que $S = \{i_1, ..., i_m\}$ avec $i_1 < ... < i_m$. On a alors $\forall i_h \in S$:

$$w_S(i_h) = (-1)^m \det({}^hB_S)$$

et

$$W(S) = (-1)^m \det(B_S)$$

Exemple
retour sur le graphe précédent

$$A_{\{1,2,3\}} = \begin{pmatrix} 0 & 3 & 4 \\ 3 & 0 & 5 \\ 4 & 5 & 0 \end{pmatrix}$$

$${}^3B_{\{1,2,3\}} = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 3 & 0 \\ 1 & 3 & 0 & 0 \\ 1 & 4 & 5 & 0 \end{pmatrix}$$

$\det({}^3B_{\{1,2,3\}}) = -18$

$$\begin{aligned} w_{\{1,2,3\}}(3) &= (-1)^3(-18) \\ &= 18 \end{aligned}$$

Autre méthode de calcul :
Une autre façon de calculer les $w_S(i)$ est donnée par la formule suivante :

$$w_S(i) = \sum_{j \in S \setminus \{i\}} (a_{ij} - a_{hj}) w_{S\setminus\{i\}}(j) \tag{3}$$

où h est un élément arbitraire appartenant à $S\setminus\{i\}$, le résultat ne dépend donc pas de h.

3.3 Ensemble dominant

On dit que $S \subseteq V$ est un *ensemble dominant* si:

1. $W(T) > 0, \forall T$ non vide $\subseteq S$
2. $w_S(i) > 0$, $\forall i \in S$
3. $w_{S\cup\{j\}}(j) < 0$, $\forall j \notin S$

Il est à noter que 1. $\not\Rightarrow$ 2., considérons le contre exemple suivant :

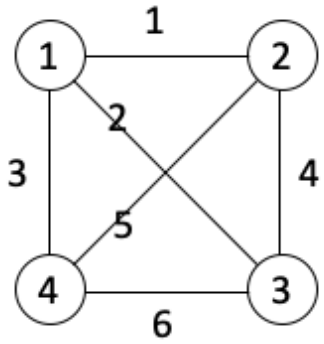


Figure 9

Soit l'ensemble $S = \{1, 2, 3\}$ et $T_1 = \{1, 2\}, T_2 = \{1, 3\}, T_3 = \{2, 3\}$.
On a :
$W(S) = 7, W(T_1) = 2, W(T_2) = 4, W(T_3) = 8$
et
$w_S(1) = -4$

3.3.1 Ensemble dominant et clusters

Il s'agit ici de montrer que les ensembles dominants peuvent être interprétés comme des "clusters" d'élements homogènes.

On considère des données représentées à l'aide d'un graphe pondéré non orienté et sans boucle: $G = (V, E)$ où V représente l'ensemble des sommets, $E \subseteq V\text{x}V$ l'ensemble des arêtes et $w : E \to \mathbb{R}_+^*$ la fonction de poids.
Les sommets représentent les points de données (par exemple les pixels pour des images), les arêtes représentent les relations de voisinage, et les poids des arêtes correspond à la similarité entre les paires de sommets.

De manière informelle un cluster est couramment défini comme " un ensemble d'entités qui se ressemblent, et tel que les entités des différents clusters ne se ressemblent pas "[5]. Dès lors, un cluster doit satisfaire 2 conditions fondamentales :

($C1$) : Avoir une forte homogénéité interne
($C2$) : Avoir une forte hétérogénéité entre les entités d'un cluster et celles à l'extérieur de ce cluster.

Lorsque les entités sont représentées sous la forme d'un graphe pondéré, ces deux conditions reviennent à dire que les poids sur les arêtes à l'intérieur d'un cluster doivent être élevés, et

que ceux sur les arêtes reliant les sommets du cluster aux sommets extérieurs doivent être faibles.

On a vu dans la partie "Interprétation intuitive" de la section 1, que le concept des poids pouvait être interprété comme une notion relative aux clusters.
De plus, les conditions $w_S(i) > 0$ pour tout $i \in S$ et $W(T) > 0$ pour tout $T \subseteq S$ pour les ensembles dominants traduisent le fait que pour tout sommet i de S le poids des liaisons entre le sommet i et les autres sommets de S est grand et donc les sommets de S sont similaires entre eux, ce qui satisfait le condition (C1) d'homogénéité interne d'un cluster.

La seconde condition $w_{S\cup\{i\}}(i) < 0$ pour tout $i \notin S$ traduit le fait que pour tous les sommets qui ne sont pas dans S le poids de la liaison entre i et les éléments de S sera petit et indique donc que i ne peut pas faire partie du même cluster que les sommets contenus dans S. Ce qui satisfait la condition (C2) d'un cluster.
Par conséquent, d'un point de vue qualitatif, un ensemble dominant satisfait les deux principales conditions pour être considéré comme un cluster. Afin d'illustrer ceci, regardons l'exemple suivant :

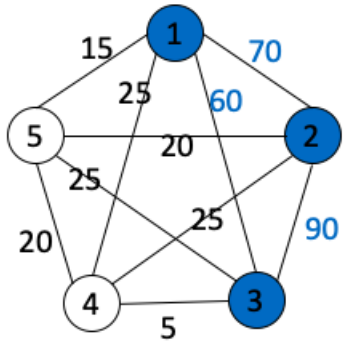


Figure 10: $\{1, 2, 3\}$ est dominant

Le sous-ensemble S de sommets $\{1, 2, 3\}$ est dominant, ce qui s'explique intuitivement en observant que les poids des arêtes "internes" à cet ensemble (60,70 et 90) sont plus grands que ceux des arêtes entre sommets internes et externes de S (qui sont compris entre 5 et 25).

4 Détermination des ensembles dominants par optimisation quadratique

4.1 Vecteur caractéristique pondéré

On dit qu'un ensemble non vide $S \subseteq V$ admet un *vecteur caractéristique pondéré* noté $\mathbf{x}^S$ si $W(S) \neq 0$. Le vecteur $\mathbf{x}^S$ est alors défini par :

$$x_i^S = \begin{cases} \frac{w_S(i)}{W(S)} & \text{si } i \in S \\ 0 & \text{sinon.} \end{cases}$$

Exemple
On reprend le graphe de la figure 1 avec $S = \{1, 2, 3\}, 4 \notin S, w_S(1) = 10$, $w_S(2) = 16$ et $w_S(3) = 18$.
On a alors :

$$\mathbf{x}^S = \begin{pmatrix} \frac{10}{44} \\ \frac{16}{44} \\ \frac{18}{44} \\ 0 \end{pmatrix}$$

Le poids total d'un sous-ensemble de sommets W(S) dépend ainsi de la topologie du graphe et des poids attribués aux arêtes, et peut très bien être nul. Par conséquent, un sous-ensemble de sommets n'admet pas toujours de vecteur caractéristique. Par exemple, tout sous-ensemble d'au moins deux sommets mutuellement non adjacents n'admet pas de vecteur caractéristique.

4.2 Point KKT

Rappelons que le théoreme de Kuhn et Tucker (ou Karush, Kuhn et Tucker, KKT) donne des conditions nécessaires de minimisation du problème suivant [6] :

$$\min_{x \in K} f(x) \tag{4}$$

lorsque la contrainte K est de la forme :

$$K = \{x \in \mathbb{R}^n : \mathbf{G}(x) \leq 0, \mathbf{H}(x) = 0\}$$

où

$$\mathbf{G} : \mathbb{R}^n \to \mathbb{R}^m$$
$$x \to \mathbf{G}(x) = (g_1(x), ..., g_m(x))^T \quad \text{avec les } g_i(x) : \mathbb{R}^n \to \mathbb{R}$$

et :

$$\mathbf{H} : \mathbb{R}^n \to \mathbb{R}^p$$
$$x \to \mathbf{H}(x) = (h_1(x), ..., h_p(x))^T \quad \text{avec les } h_i(x) : \mathbb{R}^n \to \mathbb{R}$$

Les conditions nécessaires pour qu'un point $\mathbf{x}^* \in K$ soit un point minimum local strict du problème (4) sont qu'il existe $(u_1, \ldots, u_p)$ et $(\lambda_1, \ldots, \lambda_m)$ tel que :

$$u_i \geq 0$$
$$u_i g_i(\mathbf{x}^*) = 0$$
$$\nabla f(\mathbf{x}^*) + \sum_{i=1}^{m} \lambda_i \nabla h_i(\mathbf{x}^*) + \sum_{i=1}^{p} u_i \nabla g_i(\mathbf{x}^*) = 0$$

Considérons à présent le problème posé en section 1 qui a pour but de déterminer un ensemble dominant :

$$\min_{\mathbf{x}\in\Delta} -\frac{1}{2}\mathbf{x}^T A\mathbf{x} \tag{1}$$

$$\text{ici } K = \Delta = \{\mathbf{x} \in \mathbb{R}^n : G(\mathbf{x}) \leq 0, h(\mathbf{x}) = 0\}$$

$$\text{et } f(\mathbf{x}) = -\frac{1}{2}\mathbf{x}^T A\mathbf{x}$$

avec

$$\mathbf{x} = (x_1, ..., x_n)^T$$

où :

$$G(\mathbf{x}) = (g_1(\mathbf{x}), ..., g_m(\mathbf{x}))^T, \text{ avec } g_i(\mathbf{x}) = -x_i$$

et

$$h(\mathbf{x}) = -1 + \sum_{i=1}^{n} x_i$$

et on a :

$$\nabla f(\mathbf{x}) = -A\mathbf{x}$$

$$\nabla g_i(\mathbf{x}) = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ -1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

où la ième composante du vecteur est égale à -1 avec uniquement des 0 ailleurs.

$$\nabla h(\mathbf{x}) = \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}$$

Les contraintes sont affines, donc qualifiées pour tout $\mathbf{x} \in \Delta$.
Soit $\mathbf{x} \in \Delta$ tel que $-f(\mathbf{x})$ soit un minimum local sur Δ. Alors d'après le théorème KKT, il existe des multiplicateurs de Lagrange $u_1, ..., u_n \in \mathbb{R}^+$ et $\lambda \in \mathbb{R}$ tel que :

$$\nabla f(\mathbf{x}) + \sum_{i=1}^{n} u_i \nabla g_i(\mathbf{x}) + \lambda \nabla h(\mathbf{x}) = 0$$

$\iff$
pour tout $i \in [1, n]$

$$-(A\mathbf{x})_i - u_i + \lambda = 0$$

$\iff$

$$(A\mathbf{x})_i + u_i - \lambda = 0 \tag{5}$$

et on sait d'après les conditions KKT que $x_i u_i = 0$, donc $i \in \sigma(\mathbf{x}) \Rightarrow u_i = 0$. Comme $u_i \in \mathbb{R}^+$, les conditions KKT peuvent alors se réécrire de la façon suivante pour tout $i \in [1, n]$:

$$(A\mathbf{x})_i \begin{cases} = \lambda & \text{si } i \in \sigma(\mathbf{x}) \\ \leq \lambda & \text{sinon} \end{cases} \tag{6}$$

ce qui implique $\exists \theta = -\lambda \in \mathbb{R}$ tel que

$$B_\sigma \begin{pmatrix} \theta \\ x_{i_1} \\ \vdots \\ x_{i_m} \end{pmatrix} = \begin{pmatrix} 0 & \mathbf{e}^T \\ \mathbf{e} & A_\sigma \end{pmatrix} \begin{pmatrix} \theta \\ x_{i_1} \\ \vdots \\ x_{i_m} \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

avec $\sigma = \sigma(\mathbf{x}) = \{i_1, ..., i_m\}, i_1 < ... < i_m$. On rappelle que $\sigma(\mathbf{x})$ correspond au support de $\mathbf{x}$, i.e. les valeurs d'indice de coordonnées non nulles de $\mathbf{x}$

Définition
Si $\mathbf{x} \in \Delta$ satisfait les conditions KKT, alors le vecteur $\mathbf{x}$ est appelé *point KKT*.

Lemme 2
Soit $\mathbf{x} \in \Delta$, $\sigma = \sigma(\mathbf{x})$, tel que $\mathbf{x}$ admet un vecteur caractéristique pondéré $\mathbf{x}^\sigma$. Le point $\mathbf{x}$ est un point KKT si et seulement si $\mathbf{x} = \mathbf{x}^\sigma$ et $(A\mathbf{x})_j \leq (A\mathbf{x})_i$, pour tout $i \in \sigma$ et $j \notin \sigma$. De plus dans ce cas, on a :

$$\frac{w_{\sigma \cup j}(j)}{W(\sigma)} = (A\mathbf{x})_j - (A\mathbf{x})_i = -u_j \tag{7}$$

Preuve : Soit $\mathbf{x} \in \Delta$ vérifiant les conditions KKT du problème (1), i.e. le problème défini par :

$$\min_{\mathbf{x} \in \Delta} -\frac{1}{2} \mathbf{x}^T A \mathbf{x}$$

Ce qui entraine que :

$$B_\sigma \begin{pmatrix} \theta \\ x_{i_1} \\ \vdots \\ x_{i_m} \end{pmatrix} = \begin{pmatrix} 0 & \mathbf{e}^T \\ \mathbf{e} & A_\sigma \end{pmatrix} \begin{pmatrix} \theta \\ x_{i_1} \\ \vdots \\ x_{i_m} \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

Ce qui est équivalent au système linéaire suivant :

$$\begin{cases} x_{i_1} + ... + x_{i_m} = 1 \\ a_{i_1,i_1} x_{i_1} + ... + a_{i_1,i_m} x_{i_m} = 0 \\ \vdots \\ a_{i_m,i_1} x_{i_1} + ... + a_{i_m,i_m} x_{i_m} = 0 \end{cases}$$

Comme $\mathbf{x}$ admet un vecteur caractéristique pondéré $\mathbf{x}^\sigma$, alors $W(\sigma) \neq 0$ par définition des vecteurs caractéristiques pondérés.
On a d'après le Lemme 1 et que $|\sigma| = m$: $W(\sigma) = (-1)^m \det(B_\sigma)$. Donc $W(\sigma) \neq 0$ d'où $\det(B_\sigma) \neq 0$. On a alors d'après la règle de Cramer :

$\forall h, 1 \leq h \leq m$:

$$\begin{aligned} x_{i_h} &= \frac{\det({}^hB_\sigma)}{\det(B_\sigma)} \\ &= \frac{(-1)^m w_\sigma(i_h)}{(-1)^m W(\sigma)} \qquad \text{d'après le Lemme 1} \\ &= \frac{w_\sigma(i_h)}{W(\sigma)} \end{aligned}$$

Donc $\mathbf{x} = \mathbf{x}^\sigma$. Et d'après (5) :

$$(A\mathbf{x})_j - (A\mathbf{x})_i = \lambda - u_j - \lambda + u_j$$

Et comme $i \in \sigma$, alors $u_i = 0$. On a donc :

$$\begin{aligned} (A\mathbf{x})_j - (A\mathbf{x})_i &= -u_j \\ (A\mathbf{x})_j - (A\mathbf{x})_i &\leq 0 \qquad \text{car } u_j \geq 0 \end{aligned}$$

Réciproquement, si $\mathbf{x} = \mathbf{x}^\sigma$ et $(A\mathbf{x})_j \leq (A\mathbf{x})_i$ pour tout $j \notin \sigma$ et $i \in \sigma$:

$$\begin{aligned} \frac{w_{\sigma\cup j}(j)}{W(\sigma)} &= \frac{\sum_{h\in\sigma}(a_{jh} - a_{ih})w_\sigma(h)}{W(S)} \qquad \text{d'après (3)} \\ &= \sum_{h\in\sigma} a_{jh}\frac{w_{\sigma(h)}}{W(S)} - \sum_{h\in\sigma} a_{ih}\frac{w_{\sigma(h)}}{W(S)} \\ &= \sum_{h\in\sigma} a_{jh}x_h^\sigma - \sum_{h\in\sigma} a_{ih}x_h^\sigma \\ &= (A\mathbf{x}^\sigma)_j - (A\mathbf{x}^\sigma)_i \\ &= (A\mathbf{x})_j - (A\mathbf{x})_i \qquad \text{car } \mathbf{x} = \mathbf{x}^\sigma \end{aligned}$$

$$\Rightarrow (A\mathbf{x})_i - (A\mathbf{x})_j + \frac{w_{\sigma\cup j}(j)}{W(\sigma)} = 0$$

Donc $\exists\lambda \in \mathbb{R}$ tel que $(A\mathbf{x})_i - \lambda = 0$
avec $\lambda = (A\mathbf{x})_j - \frac{w_{\sigma\cup j}(j)}{W(\sigma)}$ et $u_i = 0$.
et on a $\frac{w_{\sigma\cup j}(j)}{W(\sigma)} = (A\mathbf{x})_j - (A\mathbf{x})_i = -u_j$, avec $u_j \geq 0$ car $(A\mathbf{x})_j \leq (A\mathbf{x})_i$

Donc on a bien pour $j \notin \sigma$:

$$\begin{aligned} (A\mathbf{x})_j - \lambda + u_j &= (A\mathbf{x})_j + \frac{w_{\sigma\cup j}(j)}{W(\sigma)} - (A\mathbf{x})_j + (A\mathbf{x})_i - (A\mathbf{x})_j \\ &= \frac{w_{\sigma\cup j}(j)}{W(\sigma)} - (A\mathbf{x})_j + (A\mathbf{x})_i \\ &= 0 \end{aligned}$$

Proposition 1
Soit $\mathbf{x} \in \Delta$ un vecteur dont le support $\sigma = \sigma(\mathbf{x})$ a un poids $W(\sigma) > 0$, et qui admet dès lors un vecteur caractéristique $\mathbf{x}^\sigma$. Alors $\mathbf{x}$ est un point KKT si et seulement si les conditions suivantes sont vérifiées :

1. $\mathbf{x} = \mathbf{x}^\sigma$
2. $w_{\sigma \cup \{j\}} \leq 0, \forall j \notin \sigma$

Preuve :
D'après le lemme 2, le vecteur $\mathbf{x}$ satisfait les conditions KKT si et seulement si $\mathbf{x} = \mathbf{x}^\sigma$ et $(A\mathbf{x})_j \leq (A\mathbf{x})_i$ pour tout $i \in \sigma$ et $j \notin \sigma$. Or comme $W(\sigma) > 0$,$(A\mathbf{x})_j \leq (A\mathbf{x})_i \Leftrightarrow w_{\sigma \cup \{j\}} \leq 0$ d'après la propriété (7) du Lemme 2.

4.3 Test de la matrice hessienne bordée

Soit un problème d'optimisation de la forme :

$$\begin{aligned} &\min\ f(\mathbf{x}) \\ &\text{sous la contrainte } \mathbf{H}(\mathbf{x}) = 0 \end{aligned} \tag{5}$$

où $\mathbf{x} \in \mathbb{R}^n$ avec $\mathbf{H}(\mathbf{x}) = (h_1(\mathbf{x}), ..., h_m(\mathbf{x}))^T$.

Soit le Lagrangien $\Lambda(x, \lambda) = f(\mathbf{x}) - \sum_{i=1}^{m} \lambda_i h(\mathbf{x})$, on définit la matrice hessienne bordée comme suit [7] :

$$H = \begin{pmatrix} 0 & \dots & 0 & \frac{\partial h_1}{\partial x_1} & \dots & \frac{\partial h_1}{\partial x_n} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & 0 & \frac{\partial h_m}{\partial x_1} & \dots & \frac{\partial h_m}{\partial x_n} \\ \frac{\partial h_1}{\partial x_1} & \dots & \frac{\partial h_m}{\partial x_1} & \frac{\partial^2 \Lambda}{\partial x_1^2} & \dots & \frac{\partial^2 \Lambda}{\partial x_n x_1} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \frac{\partial h_1}{\partial x_n} & \dots & \frac{\partial h_m}{\partial x_n} & \frac{\partial^2 \Lambda}{\partial x_1 x_n} & \dots & \frac{\partial^2 \Lambda}{\partial x_n^2} \end{pmatrix}$$

ou de façon simplifiée :

$$H = \begin{pmatrix} 0 & \nabla \mathbf{H}^T \\ \nabla \mathbf{H} & \nabla^2 \Lambda \end{pmatrix}$$

Le *test de la matrice bordée* est le suivant :

On calcule les $n - m$ mineurs principaux de H. La matrice $\nabla^2 \Lambda$ est définie positive au point $\mathbf{x}$ sur $R_\mathbf{x} = \{y \in \mathbb{R}^n : \nabla \mathbf{H}(\mathbf{x}) y = 0\}$ si et seulement si tous ses mineurs sont du signe de $(-1)^m$ (On rappelle que m correspond au nombre de contraintes d'égalité).

Lemme 3

Soit $Q \subseteq V, |Q| = m$ avec $Q = \{i_1, \ldots, i_m\}$ et $i_1 < ... < i_m$. La matrice A_Q est définie négative sur le sous-espace $M = \{\mathbf{y} \in \mathbb{R}^m : \sum_{i=1}^m y_i = 0\}$ si et seulement si, pour tout ensemble non vide $T \subseteq Q$, on a $W(T) > 0$

Preuve :
On reprend le problème d'optimisation sous contrainte 1 en modifiant Δ pour ne garder que les contraintes d'égalité :

$$\min_{h(\mathbf{x})=0} \Big(-\frac{1}{2}\mathbf{x}^T A_Q \Big)$$

avec $h(\mathbf{x}) = \mathbf{e}^T\mathbf{x} - 1$. On note $f(\mathbf{x}) = -\frac{1}{2}\mathbf{x}^T A_Q \mathbf{x}$.

$$\nabla h = (1, \ldots, 1)^T; \nabla f(\mathbf{x}) = -A_Q\mathbf{x} \text{ et } \nabla^2 f(\mathbf{x}) = -A_Q$$

Soit H_Q la matrice hessienne bordée liée à la matrice A_Q. On a alors $H_Q = \begin{pmatrix} 0 & \mathbf{e}^T \\ \mathbf{e} & -A_Q \end{pmatrix}$.

On note $(\mathbf{e}, A_Q^1, \ldots, A_Q^m)\backslash A_Q^j$ la matrice $\big(\mathbf{e}, A_Q^1, \ldots, A_Q^m\big)$ où la colonne A_Q^j a été retirée. La quantité $(-1)^{1+j} \det[(\mathbf{e}, A_Q^1, \ldots, A_Q^m)\backslash A_Q^j]$ est donc égale à un mineur de B_Q.

On a d'après la formule de Laplace :

$$\begin{aligned}
\det(H_Q) &= \sum_{j=1}^{m} (-1)^{1+j} \det[(\mathbf{e}, -A_Q^1, \ldots, -A_Q^m)\backslash A_Q^j] \\
&= (-1)^{m-1} \sum_{j=1}^{m} (-1)^{1+j} \det[(\mathbf{e}, A_Q^1, \ldots, A_Q^m)\backslash A_Q^j] \\
&= (-1)^{m-1} \det(B_Q)
\end{aligned}$$

D'après le test de la matrice bordée, il vient que la matrice $-AQ$ est symétrique définie positive si et seulement si les $n-1$ mineurs de la matrice H sont du signe de $(-1)^1$ donc négatifs.
Chaque mineur est de la forme suivante :

$$H_{Q_1} = \begin{pmatrix} 0 & \mathbf{e}^T \\ \mathbf{e} & -A_Q \end{pmatrix}$$

$$H_{Q_2} = \begin{pmatrix} 0 & \mathbf{e}^T \\ \mathbf{e} & -A_{Q\backslash\{i_m\}} \end{pmatrix}$$

$$\vdots$$

$$H_{Q_{n-1}} = \begin{pmatrix} 0 & \mathbf{e}^T \\ \mathbf{e} & -A_{Q\backslash\{i_m,\ldots,i_3\}} \end{pmatrix}$$

Soit $T = \{i_1, ...i_z\} \subseteq Q$, avec $\{i_1 < ... < i_z\}$. On a :

$$\begin{aligned}\text{signe}(\det(H_T)) &= \text{signe}((-1)^{z-1}\det(B_T)) \\ &= \text{signe}((-1)W(T)) \qquad \text{d'après le Lemme 1}\end{aligned}$$

Donc $\text{signe}(\det(H_T)) = -1 \Leftrightarrow W(T) > 0$.

On a donc prouvé que si $Q = \{i_1, \ldots, i_m\}$ avec $i_1 < ... < i_m$, alors A_Q est définie négative sur M, entraine que $W(T) > 0$ pour tout T de la forme $[i_1, ...i_z] \cap Q$.

Montrons maintenant que la condition doit être respectée quel que soit le sous-ensemble T pris.

Soit $\tilde{A}_n$ la matrice correspondant à la matrice A à laquelle n permutations symétriques ont été effectuées. Pour rappel, une permutation symétrique consiste par exemple à échanger les lignes 2 et 3 dans la matrice puis les colonnes 2 et 3.
Soit P la matrice de permutation des lignes i et j. On note $(e_1, ..., e_m)$ la base canonique de l'ensemble $\mathbb{R}^m$. On a alors $\forall i, j$:

$$P = \begin{pmatrix} e_1 & \ldots & \underset{\text{ième colonne}}{e_j} & \ldots & \underset{\text{jème colonne}}{e_i} & \ldots & e_m \end{pmatrix}$$

On a immédiatement que $P^2 = I$. Donc par définition de $\tilde{A}_n$, on a :

$$\tilde{A}_n = P_n \ldots P_1 A P_1 \ldots P_n$$

où $P_1, \ldots, P_n$ sont des permutations choisies de façon appropriée.
$\tilde{A}_n$ est symétrique car A est symétrique, et pour tout $\mathbf{x} \in M, P_1 \ldots P_n \mathbf{x} \in M$
Si A est symétrique définie positive sur M, alors $\forall \mathbf{x} \in M$:

$$\mathbf{x}^T \tilde{A}_n \mathbf{x} = \mathbf{x}^T P_n \ldots P_1 A P_1 \ldots P_n \mathbf{x}$$

et comme, $P_1 \ldots P_n \mathbf{x} \in M$ et que A est symétrique définie positive sur M :
$\mathbf{x}^T P_n \ldots P_1 A P_1 \ldots P_n \mathbf{x} > 0$.

Donc quel que soit le nombre de permutations symétriques, $\tilde{A}_n$ est définie positive sur M si et seulement si A est définie positive sur M.

Donc, $\forall T \subseteq Q$ non vide, on peut trouver un ensemble de permutations symétriques qui définit une matrice $\tilde{A}_n$ tel que :

$$A \text{ est symétrique définie positive} \Leftrightarrow \tilde{A}_n \text{ est symétrique définie positive} \Rightarrow W(T) > 0$$

4.4 Condition d'optimalité de second ordre

Soit un problème d'optimisation de la forme :

$$\min f(\mathbf{x})$$
$$\mathbf{H}(\mathbf{x}) = 0$$
$$\mathbf{G}(\mathbf{x}) \leq 0$$

où $\mathbf{x} \in \mathbb{R}^n$ avec $\mathbf{H}(\mathbf{x}) = (h_1(\mathbf{x}), ..., h_m(\mathbf{x}))^T$
et $\mathbf{G}(\mathbf{x}) = (g_1(\mathbf{x}), ..., g_p(\mathbf{x}))^T$
Un point $\mathbf{x}^*$ est un minimum strict du problème s'il existe $(\lambda_1, ..., \lambda_m) \in \mathbb{R}^m$ et $(u_1, ..., u_p) \in \mathbb{R}^p$ tel que les conditions KKT soient satisfaites [6], i.e.

$$u_i \geq 0$$
$$u_i g_i(\mathbf{x}^*) = 0$$
$$\nabla f(\mathbf{x}^*) + \sum_{i=1}^{m} \lambda_i \nabla h_i(\mathbf{x}^*) + \sum_{i=1}^{p} u_i \nabla g_i(\mathbf{x}^*) = 0$$

et que la matrice *hessienne de Lagrange* définie par :

$$\mathbf{L}(\mathbf{x}^*) = \nabla^2 f(\mathbf{x}^*) + \sum_{i=1}^{m} \lambda_i \nabla^2 h_i(\mathbf{x}^*) + \sum_{i=1}^{p} u_i \nabla^2 g_i(\mathbf{x}^*)$$

est définie positive sur :

$$P = \{y \in \mathbb{R}^n : \nabla \mathbf{H}(\mathbf{x}^*) y = 0, \nabla g_j(\mathbf{x}^*) y = 0, \forall j \in J\}$$

où, $J = \{j : g_j(\mathbf{x}^*) = 0, u_j > 0\}$

4.5 Résultat Principal

Théorème 1

1. Si S est un ensemble dominant, alors son vecteur caractéristique pondéré $\mathbf{x}^S$ est une solution strict du problème de minimisation (1) : $\min_{\mathbf{x} \in \Delta} -\frac{1}{2}\mathbf{x}^T A \mathbf{x}$.

2. de façon réciproque, si $\mathbf{x}^*$ est une solution strict du problème (1), alors son support $\sigma = \sigma(\mathbf{x})$ est un ensemble dominant si $w_{\sigma \cup \{j\}} \neq 0, \forall j \notin \sigma$

Preuve 1 :
Soit S un ensemble dominant avec $|S| = m$. Alors d'après la proposition 1, son vecteur caractéristique pondéré $\mathbf{x}^S$ vérifie les conditions KKT.

Soit $f(x) = -\frac{1}{2}\mathbf{x}^T A \mathbf{x}$ la fonction à minimiser de (1). La matrice hessienne de Lagrange du problème (1) est donné par :

$$\mathbf{L}(\mathbf{x}) = \nabla^2(-\frac{1}{2}\mathbf{x}^T A\mathbf{x}) + \lambda\nabla^2 h(\mathbf{x}) + \sum_{i=1}^{n} u_i \nabla^2 g_i(\mathbf{x})$$
$$= -A$$

car $\nabla^2 h(\mathbf{x}) = 0$ et $\nabla^2 g_i(\mathbf{x}) = 0$

D'après la condition d'optimalité de second ordre, $\mathbf{x}^S$ est un minimum strict du problème 1, si la matrice $-A$ est symétrique défnie positive sur

$$P = \{\mathbf{y} \in \mathbb{R}^n : \sum_i y_i = 0, y_j = 0, \forall j \notin S\}$$

car $\forall i \in S, u_i = 0$
Or comme $\forall j \notin S, y_j = 0$, on a l'équivalence suivante :

$$-A \text{ est définie positive sur } P \Leftrightarrow -A_S \text{ est définie positive sur } M$$

Et comme $\forall T$ non vide $\subseteq S, W(T) > 0$ par définition des ensemble dominants, on a alors d'après le Lemme 3 que la matrice $-A_S$ est définie positive sur $M = \{\mathbf{y} \in \mathbb{R}^m \sum_{i=1}^m y_i = 0\}$. Donc d'après la seconde condition d'optimalité, $f(\mathbf{x}^S)$ est un minimum strict du problème (1).

Preuve 2 :

Préliminaire :
Si $\mathbf{x}^*$ est un minimum local strict du problème (1), avec $\sigma = \sigma(\mathbf{x}^*)$ son support, alors la matrice A_σ est définie négative sur M.

Preuve : Montrons d'abord que si $\mathbf{x}^*$ est un minimum local strict de (1), alors $\mathbf{x}_\sigma$ est un minimum local strict de $-\frac{1}{2}\mathbf{x}^T A_\sigma \mathbf{x}$ (avec $\mathbf{x}_\sigma$ correspondant au vecteur auquel on a gardé uniquement les indices présent dans σ).
En effet, comme $-\frac{1}{2}\mathbf{x}_\sigma^T A_\sigma \mathbf{x}_\sigma = -\frac{1}{2}\mathbf{x}^{*T} A\mathbf{x}^*$, supposons qu'il existe un vecteur $\mathbf{y}$ tel que $-\frac{1}{2}\mathbf{y}^T A_\sigma \mathbf{y} < -\frac{1}{2}\mathbf{x}_\sigma^T A_\sigma \mathbf{x}_\sigma$. Si on note $\mathbf{z}$ le vecteur $\mathbf{y}$ auquel on a rajouté des 0 aux indices qui ne sont pas dans σ pour qu'il soit de la bonne dimension, on a alors : $-\frac{1}{2}\mathbf{z}^T A\mathbf{z} < -\frac{1}{2}\mathbf{x}^{*T} A\mathbf{x}^*$, ce qui contredit la proposition initiale.

Comme $\nabla^2(-\frac{1}{2}\mathbf{x}_\sigma^T A_\sigma \mathbf{x}_\sigma) = -A_\sigma$ et que
$\nabla^2\mathbf{h} = 0$
$\nabla^2\mathbf{g} = 0$
Alors la Hessienne $\mathbf{L}(\mathbf{x}_\sigma^*) = -A_\sigma$
Comme $\mathbf{x}_\sigma^*$ est un minimum local strict, il vient donc d'après le seconde condition d'optimalité que la matrice $-A_\sigma$ est définie positive sur $P = \{y \in \mathbb{R}^n : \nabla\mathbf{h}(\mathbf{x}*)y = 0, \nabla g_j(\mathbf{x}*)y = 0, \forall j \in J\}$
or, $\forall i \in \sigma, u_i = 0$, donc $J = \varnothing$
et comme $\nabla h = (1, \ldots, 1)^T$, on a donc que A_σ est définie négative sur M si $\mathbf{x}^*$ est un minimum local strict du problème (1)

Soit $\mathbf{x}^*$ une solution locale strict de (1), alors d'après la proposition précédente, A_σ est déifinie négative sur M. Donc d'après le test de la matrice bordée, on a $\forall T$ non vide $\subseteq \sigma, W(T) > 0$.
Comme $\mathbf{x}^*$ vérifie les équations KKT, on a $\mathbf{x}^* = \mathbf{x}^\sigma$.
De plus, comme $\mathbf{x}^* \in \Delta$ et que $W(\sigma) > 0$, alors $w_S(i) > 0$ d'après la définition des vecteurs caractéristiques pondérés.
Enfin, on a d'après la proposition 1 que $w_{\sigma\cup\{j\}} \leq 0, \forall j \notin \sigma$. Et comme $w_{\sigma\cup\{j\}} \neq 0, \forall j \notin \sigma$ par hypothèse, il vient que $w_{\sigma\cup\{j\}} < 0, \forall j \notin \sigma$.
Toutes les conditions sont respectés, il advient dès lors que σ est un ensemble dominant.

Note : La condition $w_{\sigma\cup\{j\}} \neq 0$ pour tout $j \notin \sigma$ est une technicité due à la présence de solutions "fallacieuses" dans (1), à savoir des solutions dont le support n'admet pas de vecteur caractéristique pondéré.
S'il existe $j \notin \sigma$ tel que $w_{\sigma\cup\{j\}} = 0$, alors le résultat précédent ne permet pas de conclure sur l'existence d'un ensemble dominant pour un vecteur $\mathbf{x}^*$ qui minimise (1). Cela, correspond cependant à une situation non générique et donc, dans la suite, il en sera fait abstraction.

5 Trouver un équilibre

5.1 Réplication Dynamique

Afin de pouvoir trouver un ensemble dominant du graphe pondéré, il est nécessaire de résoudre le problème d'optimisation quadratique. On va employer une technique d'optimisation continue afin de pouvoir estimer la solution de ce problème.

Considérons la matrice M, à coefficient positifs de taille $n \times n$ et solution du système dynamique suivant :

$$\dot{x_i}(t) = x_i(t)[(Mx(t))_i - x(t)^T Mx(t)] \tag{8}$$

La notation avec un point désigne la dérivée par rapport à la variable t. De plus, la version discrète de ce système peut être écrit sous la forme :

$$\dot{x_i}(t+1) = x_i(t)\frac{(Mx(t))_i}{x(t)^T Mx(t)} \tag{9}$$

(8) et (9) sont appelées les équations de réplications dynamiques. De plus, les points stationnaires de (8) et (9), i.e $\dot{x}_i(t) = 0$ pour (1) et $x_{i+1}(t) = x_i(t)$ pour (2) coincident et satisfont :

$$x_i(t)[(Mx(t))_i - x(t)^T Mx(t)] = 0$$

En pratique on calcule la distance $\|x_{i+1}(t) - x_i(t)\|$ et on arrête d'itérer quand celle-ci devient inférieure à un certain seuil ϵ, par exemple $\epsilon = 10^{-7}$.

L'idée générale est de calculer une solution stationnaire x_T à partir d'un point proche de x_T que l'on appelle $x_0 = x(t_0)$ de coordonnées $\forall i\ x_i(t_0) = x_{i^0} = \frac{1}{n}$ par exemple, et qui va converger de manière asymptotiquement stable, c'est-à-dire $\lim_{t\to\infty} x(t)$= x_T

De manière évidente, un point quelconque pris dans Δ sera invariant dans Δ selon la dynamique des équations, qui sont souvent utilisées en théorie des jeux et en biologie évolutionnaire.

$$\sum_i x_i(t+1) = \sum_i x_i(t)\frac{(Mx(t))_i}{x(t)^T Mx(t)} = \frac{x(t)^T Mx(t)}{x(t)^T Mx(t)} = 1$$

Théorème 2

Si M est une matrice symétrique positive et $x(t)$ est une trajectoire du système dynamique (8) et (9) alors la fonction $x(t)^T Mx(t)$ est strictment croissante quand t croît, c'est-à-dire que $x(t+1)^T Mx(t+1) > x(t)^T Mx(t)$.

De plus la trajectoire $x(t)$ converge vers un point stationnaire x_T. Finalement on a que : $x \in \Delta$ est asymptotiquement stable sous (8) et (9) $\iff$ x est un maximum local strict de $x(t)^T Mx(t)$ sur Δ.

Exemple

Les équations de réplication dynamiques sont une heuristique naturelle pour trouver les ensembles dominants dans un graphe pondéré. En choisissant $M = A$ une matrice d'adjacence et en partant d'une position initiale x_0 dans Δ, on itère un très grand nombre de fois jusqu'à atteindre le point asymptotique dans Δ qui correspond au maximum local de $x(t)^T Mx(t)$ et par conséquant à l'ensemble dominant du graphe. Une fois qu'un ensemble dominant est trouvé, on l'enlève de l'ensemble des sommets du graphe et on recommence jusqu'à trouver le nouvel ensemble dominant et ainsi de suite. Chaque ensemble dominant correspond alors à un cluster.

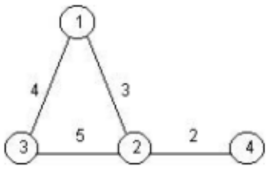
Il est possible qu'au bout d'un certain nombre d'itérations, aucune coordonnée d'un vecteur caractéristique ne soit exactement nulle mais on peut faire une approximation numérique :

$$\delta\tilde{(x)} = \{i | x_i > \theta * max(x)\}\, \theta \in [0,1]$$

Ci-dessous un exemple implémenté sur Matlab et calculé après seulement 8 itérations :

$\widetilde{i \in S}$

Exemple
Soit le graphe suivant:



qui admet pour matrice d'adjacence : $\begin{pmatrix} 0 & 3 & 4 & 0 \\ 3 & 0 & 5 & 2 \\ 4 & 5 & 0 & 0 \\ 0 & 2 & 0 & 0 \end{pmatrix}$

(a) Graphe et Matrice d'adjacence associée

```
X =

    0.2257
    0.3648
    0.4095
    0.0000
```

(b) Calcul du vecteur caractéristique sur Matlab après 8 itérations

5.2 Réplication Dynamique Exponentielle

Une variante des équations de réplication dynamique consiste à calculer une trajectoire exponentielle qui converge plus rapidement et est plus adaptée aux grands graphes, i.e. ici quand la matrice M est de grande dimension. [1]

$$x_i(t+1) = x_i(t)\frac{e^{(\kappa * Mx(t))_i}}{x(t)^T Mx(t)} \tag{10}$$

κ représente l'accélération de la dynamique exponentielle et qui a généralement pour valeur soit 1 ou 2.

5.3 Algorithme d'Infection-Immunisation

L'algorithme d'infection immunisation repose sur le principe de la **théorie des jeux évolutionnaire**. Avant d'expliquer la théorie, introduisons quelques notions nécessaires à sa compréhension :

- On appelle **stratégie mixte** le vecteur distribution de probabilité $x = (x_1, ..., x_n)^T$. Le vecteur x appartient au simplexe standard de dimension n :

  $\{\Delta = x \in \mathbb{R}^n : \sum_i x_i = 1, x_i \geq 0, i = 1, ..., n\}$

- On appelle A la **matrice d'utilité** $n \times n$ et de coefficient a_{ij}, $i, j \in \{1, ..., n\}$
- On appelle Oq l'**ensemble des stratégies pures** $\{1, ..., n\}$
- On appelle **support de** x, $\sigma(x) = \{i \in O : x_i \geq 0\}$
- On appelle e^i la $i^{ème}$ colonne de la matrice identité de dimension n
- On appelle le **gain attendu** d'un joueur qui joue la stratégie pure i contre une stratégie mixte x : $\pi(e^i|x) = (Ax)_i = \sum_j a_{ij}x_j$

- Le gain attendu en adoptant une stratégie mixte y contre une autre stratégie mixte $x \in \Delta$ est $\pi(y|x) = y^T Ax$
- On note aussi comme notations raccourcies : $\pi(x) = \pi(x|x) = x^T Ax, \pi(y-x|z) = \pi(y|z) - \pi(x|z), \pi(y-x) = \pi(y-x|y) - \pi(y-x|x)$
- On appelle x **équilibre de Nash** ou **stratégie de Nash** si $\forall$ y $\in \Delta$ $\pi(y-x|x) \leqslant 0$ et implicitement, toutes les stratégies en dehors de $\sigma(x)$ ont un gain inférieur ou égale à $\pi(x)$
- On dit que x est une **stratégie évolutionairement stable (Evolutionary Stable Strategy ESS)** $\iff$ x est un équilibre de Nash et que $\forall y \in \Delta \setminus \{x\}$ $\pi(y-x|x) = 0, \pi(y-x|x) < 0$

Théorème 3

Soient $S \subseteq V$, S un ensemble dominant, si A est une matrice d'affinité alors son vecteur caractéristique x^S est un **ESS** dans un jeu à 2 joueurs vis à vis de la matrice de A. Réciproquement, si x est un **ESS** dans un jeu à 2 joueurs, alors $S = \sigma(x)$ est un ensemble dominant vis à vis de la matrice A.

Preuve : Soit $x^S \in \Delta$ un ESS pour A (matrice d'affinité), alors $\forall y \in \Delta \setminus \{x\}$ on a bien ${x^s}^T Ax^s > y^T Ax^s$ car $\pi(y-x^s|x^s) = 0, \pi(y-x^s|x^s) < 0$ et donc ${x^s}^T Ax^s$ est maximal sur Δ. Il s'agit donc bien d'un ensemble dominant.

5.3.1 Dynamique évolutionnaire

Soit x la stratégie en place (que l'on peut voir aussi comme une population) et y la **stratégie concurrente** (que l'on peut voir comme une **population** mutante qui va venir envahir x), on définit $z = (1-\epsilon)x + \epsilon y$ avec $\epsilon \in \{0, 1\}$ et la fonction du score de y versus x comme étant :

$$h_x(y, \epsilon) = \pi(y-x|z) = \epsilon\pi(y-x) + \pi(y-x|x) \tag{11}$$

On définit alors $b_x(y)$ la **barrière d'invasion** de $x \in \Delta$ contre la **stratégie mutante** y comme étant ϵ_y, la plus grande population de y et de telle sorte que $\forall\ 0 < \epsilon < \epsilon_y$, le gain de x soit plus grand ou égal à celui de y. Formellement cela s'écrit :

$$b_x(y) = inf(\{\epsilon \in (0,1) : h_x(y, \epsilon) > 0\} \bigcup \{1\}) \tag{12}$$

$\forall$ x,$y \in \Delta$ on dit que x est **immunisé** contre y $\iff$ $b_x(y) > 0$. Notons que si $\pi(y-x|x) < 0$ ou que $\pi(y-x|x) = 0$ et $\pi(y-x) \leqslant 0$ alors x est immunisé contre y. Si $\pi(y-x|x) > 0$ alors y est **infectieux** pour x. On appelle l'**ensemble des stratégies infectieuses** pour x :

$\Upsilon(x) = \{y \in \Delta : \pi(y-x|x) > 0\}$

Soit $y \in \Upsilon(x)$, cela implique que $b_x(y) = 0$, si l'on admet une invasion ϵ d'une population y et tant que la fonction score de x contre y est positive, alors à un moment celle-ci va devenir négative [8] et on définit :

$$\delta_x(y) = inf(\{\epsilon \in (0,1) : h_x(y,\epsilon) < 0\} \bigcup \{1\}) \tag{13}$$

On note que si y est infectieux pour x, alors $\delta_x(y) > 0$, tandis que si x est immunisé contre y alors $\delta_x(y) = 0$. Un autre moyen de trouver ce minimum est de chercher la pente minimale car la fonction de score est une fonction affine. [8] On peut donc aussi définir :

$$\delta_x(y) = min[\frac{\pi(y-x|x)}{\pi(y-x)}, 1] \tag{14}$$

si $\pi(y-x) < 0$, 1 sinon.

Proposition 2

Soit $y \in \Upsilon(x)$, $\delta = \delta_x(y)$ et $z = (1-\delta)x + \delta y$, alors $y \notin \Upsilon(z)$. [8]

5.3.2 Méthodologie de l'Algorithme

La proposition précédente est très importante car le principe de l'algorithme est basé dessus. Si x est un **équilibre de Nash** alors $\Upsilon(x) = \varnothing$. L'idée générale et donc de partir d'une population y infectieuse pour x et d'effectuer des itérations en mettant à jour y à chaque itération tel que décrit dans la **Proposition 2**. On va réitérer jusqu'à ce que l'on ne trouve plus de stratégie infectieuse. [8]

En formalisant ce processus cela donne :

$$x^{(t+1)} = \delta_{S(x^t)}(x^t)[S(x^t) - x^t] + x^t \tag{15}$$

où $S : \Delta \to \Delta$ est une fonction qui retourne soit une stratégie infectieuse pour x ou bien x. En utilisant ce système dynamique, on espère atteindre une population x qui ne pourra pas être infectée par d'autres stratégies. Si c'est le cas alors x est un équilibre de Nash.

Théorème 4

Soit x une stratégie appartenant à Δ. Les 3 propositions sont équivalentes :

1. $\Upsilon(x) = \varnothing$. Il n'y a pas de stratégie infectieuse pour x.
2. x est une stratégie de Nash
3. x est un point fixe de la dynamique (15)

Preuve : Si x est une stratégie de Nash, alors $\forall\ y \in \Delta$, $\pi(y-x|x) \leqslant 0$. Donc $\Upsilon(x) = \varnothing$, $\delta = 0$ ce qui implique que $S(x) = x$. Le système dynamique (15) va renvoyer x et il s'agira d'un point stationnaire.

Théorème 5 Soit $\{x^{(t)}\}_{t\geqslant 0}$ une trajectoire de (15), on a $\forall$ t :

$$\pi(x^{(t+1)}) \geqslant \pi(x^{(t)})$$

Donc d'après ce théorème, en calculant par itération les trajectoires du système dynamique (15), on augmente le gain jusqu'à ce que l'on atteigne un équilibre de Nash. [8]

Proposition 3

On propose la fonction de sélection d'une stratégie particulière (utilisée dans l'algorithme) :

$$S_{Pure}(x) = \begin{cases} e^i \text{ si } \pi(e^i - x|x) > 0 \text{ et i} = \mathrm{M}(x) \\ \overline{e^i} \text{ si } \pi(e^i - x|x) < 0 \text{ et i} = \mathrm{M}(x) \\ x \text{ sinon} \end{cases}$$

$$\text{Avec M(x)} = argmin_{i=1,..,n}|\pi(e^i - x|x)|$$

$$\overline{e^i} = \tfrac{x_i}{x_i - 1}(e^i - x) + x$$

Soit la population $x \in \Delta$, il existe une stratégie infectieuse pour x, i.e $\Upsilon(x) \neq \varnothing \iff S_{Pure}(x) \in \Upsilon(x)$

Preuve : Soit y $\in \Upsilon(x)$. Alors $0 < \pi(y - x|x) = \sum_i y_i \pi(e^i - x|x)$. Il existe donc bien une stratégie pure i qui est infectieuse pour x et $e^i \in \Upsilon(x)$ pour un $i \in \{1, ..., n\}$.

Théorème 6

Un état x, avec la stratégie $S_{Pure}(x)$ et une matrice de gain symétrique A, est asymptotiquement stable pour la dynamique (S) $\iff$ x est un ESS. [8]

Théorème 7

Soit $x(t)$ la trajectoire de (15) et sa transformation linéaire $Ax(t)$, les 2 sont calculables en temps linéaire. L'algorithme calculera à chaque itération la transformation linéaire $Ax(t)$.

Preuve : Posons $x = x(t)$. Si $S_{Pure}(x) = e^i$, alors $\delta_{e^i}(x)$ est calculable en temps linéaire n car $\pi(e^i - x|x) = (Ax)_i - \pi(x)$ et $\pi(e^i - x) = a_{ii} - 2Ax + \pi(x)$. De plus :

$Ax^{(t+1)} = \delta_{e^i}(x)[A_i - Ax] + Ax$

A_i représente la $i^{ème}$ colonne de A

Si $S_{Pure}(x) = \overline{e^i}$,

$Ax^{(t+1)} = (\frac{x_i}{x_i - 1})\delta_{\overline{e^i}(x)}[A_i - Ax] + Ax$

$\pi(\overline{e^i} - x|x) = (\frac{x_i}{x_i - 1})\pi(e^i - x|x)$ et $\pi(\overline{e^i} - x) = (\frac{x_i}{x_i - 1})^2\pi(e^i - x)$ sont aussi calculables en temps linéaire n. [8]

6 Implémentation

6.0.1 Préliminaire : exemples de papiers ayant recours à la méthode des ensembles dominants

La méthode des ensembles dominants a été utilisé dans plusieurs domaines radicalement differents :

- **Regroupement de protéines** : Dans [10], la méthode des ensembles dominants a été utilisée pour étudier l'interaction des protéines pendant la plasticité neuronale.
- **Détection de groupes conversationnels** : Dans [11] [12], la méthode des ensembles dominants a été utilisée pour détecter automatiquement des groupes conversationnels en temps réel dans des séquences vidéo.
- **Connectomique cérébrale** : Dans [13] [14] la méthode des ensembles dominants a été utilisée pour étudier les connexions neuronales dans l'hémisphère avec pour objectifs de regrouper automatiquement les fibres nerveuses ayant la même forme. De plus, une procédure d'appariement permet de récupérer les structures communes à travers les sujets différents.

Nous avons utilisé pour l'implémentation qui suit la librairie DSlib codé en matlab dont le détail peut-être trouvé dans le papier suivant : [15].

6.1 Etude de cas : images

6.1.1 Transformation d'une image en graphe

Soit G un graphe non orienté, dont les sommets sont les mn pixels d'une image. Chaque sommet i est définie par un niveau d'intensité B(i), ainsi que par sa position dont les coordonnées sont notées X(i).

On définit le poids $w_{i,j}$ de l'arrête entre les pixels i et j par la formule suivante :

$$w_{i,j} = \begin{cases} e^{\frac{-\|B(i)-B(j)\|_2^2}{\sigma^2}} & \text{si } \|X(i)-X(j)\| < r \\ 0 & \text{sinon.} \end{cases}$$

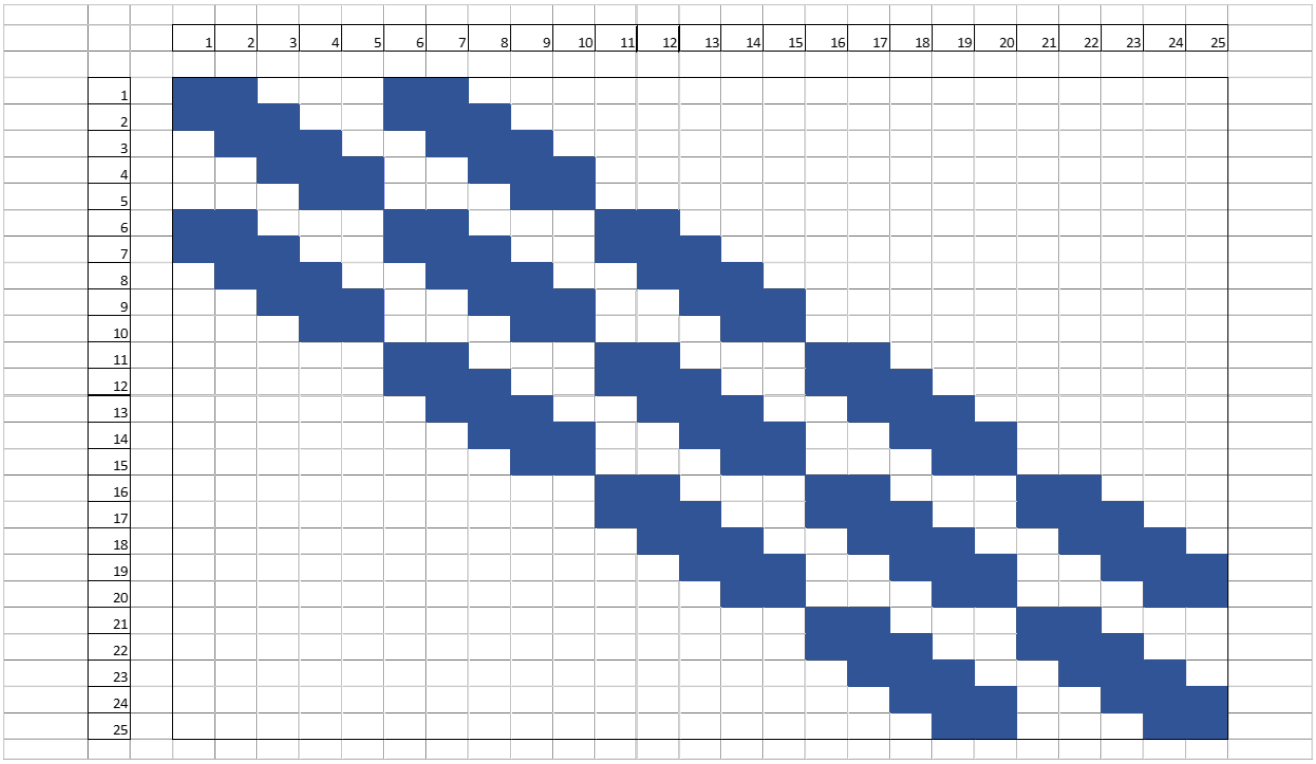
Où l'intensité du pixel i est donnée par $B(i) := \begin{pmatrix} v \\ v.s.sin(h) \\ v.s.cos(h) \end{pmatrix}$ avec (h, s, v) le code HSV du pixel.

Après avoir défini $w_{i,j}$ ainsi, il est clair que deux pixels trop éloignés ne seront pas connectés. En effet si la distance entre les pixels i et j est trop grande (dans notre cas supérieure à r), alors $w_{i,j} = 0$. Si deux pixels i et j "proches" ont une intensité similaire alors $w_{i,j}$ sera plus important, c'est à dire que le poids de l'arête reliant i et j sera important, la connexion entre ces deux pixels est donc bien forte. De la même manière, plus les pixels i et j seront éloignés en contraste plus $\|B(i) - B(j)\|_2^2$ sera grand et donc le poids de la connexion entre i et j sera faible.

Regardons l'image suivante:

Le logo de Dauphine est ici représenté par 25 pixels, le cercle rouge représente l'ensemble des pixels i tels que $\|X(13) - X(i)\| < r$. C'est à dire que pour tous les pixels j qui ne sont pas dans ce cercle, $w_{13,j} = 0$.

Cette image a donc pour matrice d'adjacence la matrice A que nous avons représenté ci dessous de manière un peu particulière. En effet les coefficients non nuls sont mis en évidence en bleu:



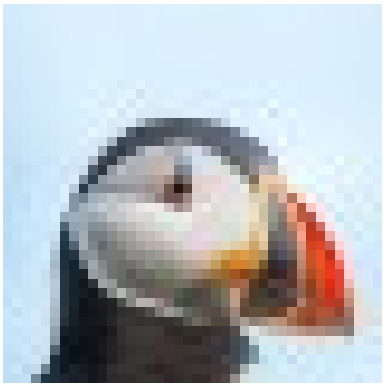
On voit bien que dans notre exemple, A est une matrice creuse. Cela peut s'expliquer par le choix de r qui est petit. En effet pour i fixé, plus r est petit, plus $\{j / \|X(j) - X(i)\| < r\}$ est petit et donc plus le nombre de pixels j tels que $w_{i,j} = 0$ est grand, c'est à dire que le nombre de coefficients nuls de A est important.

Une telle matrice d'adjacence est très intéressante pour le code puisqu'elle peut être implémentée comme une matrice creuse et permet un gain de stockage et de rapidité dans les calculs.

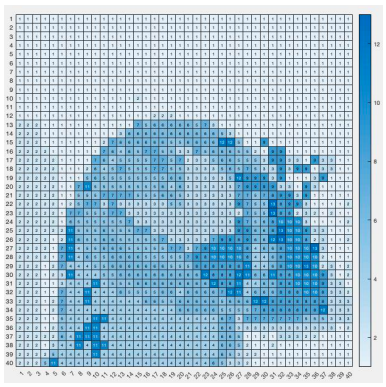
Dans le papier en revanche, un rayon r n'est pas définie, ce qui fait que chaque point est relié à tous les autres. Cela génère un cout de calcul important, qui pourrait gagné à être déiminué en rajoutant un rayon. Par exemple, une image 100x100 contient 10 000 pixels. calculer toutes les distances euclidiennes deux à deux (pdist avec matlab) revient à faire $\binom{10000}{2} = 49\ 995\ 000$ opérations.

6.1.2 Application à des données sans bruits

Nous avons appliqué tout d'abord appliqué la méthode sur une image de 40 pixels et comparé avec des algorithmes tel que kmeans ou ncut (la théorie derrière ces méthodes peut-être trouvée en annexe) :



(c) Image originale



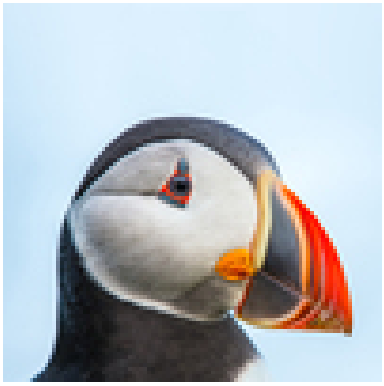
(d) Dominant set

(e) k-means

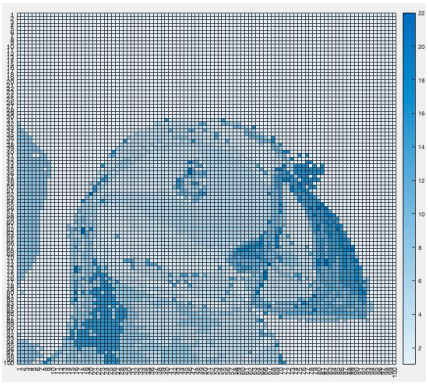
(f) Ncut

Figure 11: Teste sur une image de 40 pixels

On observe ici que l'algorithme fournit de bons résultats. Les quelques points au-dessus du bec du pingouin sont parmis les 10 % des pixels restant, assignés au cluster le plus proche.



(a) Image originale



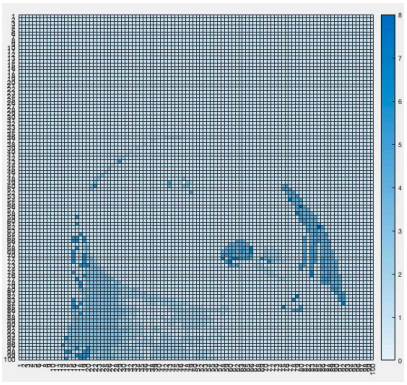
(b) Dominant set

(c) k-means

(d) Ncut

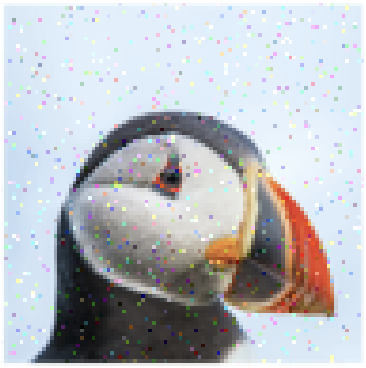
Figure 12: Teste sur une image de 100 pixels

On remarque que le rendu l'algorithme DS est plus précis dans la définition de petits cluster, notamment pour le cluster de l'oeil du pingouin que les autres méthodes testées.
La méthode d'itération par les équations de réplications dynamique fournit d'ailleurs de biens meilleurs résultats que la méthode d'infection immunisation, dont le résultat peut-être trouvé ci-dessous :

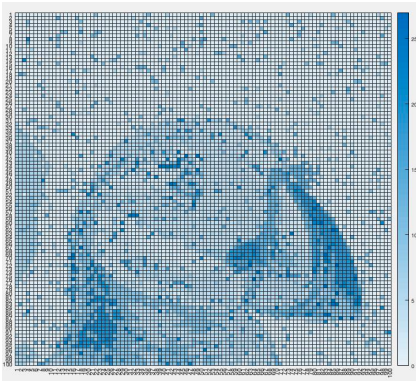


(a) DS avec infection Imunisation

6.1.3 Données bruitées

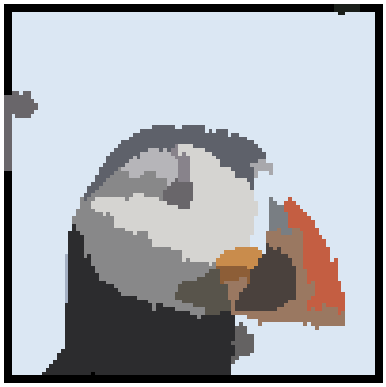


(b) Image originale



(c) Dominant set

(d) k-means



(e) Ncut

Figure 13: Teste sur une image de 40 pixels

On remarque que l'algorithme Dominant Set détecte les pixels bruités comme étant des clusters à part entière.

6.2 Etude de cas sur 2 jeux de données

Dans cette partie, nous allons appliquer les algorithmes de clustering sur des ensembles de données bien connus et accessibles au grand public : les jeu de données **"Jain"** et **"Iris"** qui sont des jeux de données biologiques. Jain comporte un échantillon de 373 individus qui ont 2 variables et qui forment 2 clusters. Iris comporte un échantillon de 150 individus qui ont 4 variables et forment 3 clusters.

Pour évaluer la performance d'un algorithme de clustering sur un échantillon, on propose d'utiliser la **F-Mesure** sur un ensemble de données à 2 clusters (Jain) et sur un à 3 clusters (Iris).

6.2.1 F-Mesure ou F-Score

Si l'on connaît les classes initiales, on peut évaluer la précision et le rappel de l'algorithme de clustering. Prenons un exemple simple avec 2 classes : A et B. Appelons **"VP"** les vrais positifs, **"VN"** les vrais négatifs, **"FP"** les faux positifs et **"FN"** les faux négatifs.

Classes	A initiaux	B initiaux
A prédits	1	3
B prédits	2	2

Ici pour la classe A, $VP = 1, FP = 3, VN = 2, FN = 2$, et on définit :

- Précision = $\frac{VP}{VP+FP}$
- Rappel = $\frac{VP}{VP+VN}$
- F-Mesure ou F-Score = $\frac{2*Précision*Rappel}{Précision+Rappel}$

On peut aussi étendre ce principe à plusieurs classes et elles auront chacune une précision, un rappel et un F-Score. On somme alors sur les lignes pour déterminer le dénominateur de la précision d'une classe et sur les colonnes pour déterminer le dénominateur du rappel d'une classe, le numérateur étant toujours le nombre de vrais positifs d'une classe.

Classes	Chats initiaux	Poissons initiaux	Poulets initiaux
Chats prédits	4	6	3
Poissons prédits	1	2	0
Poulets prédits	1	2	6

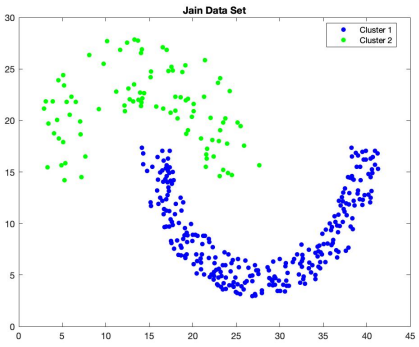
Ce qui donne alors le tableau suivant :

Classes	Précision	Rappel	F-Score
Chats	30.8%	66.7%	42.1%
Poissons	66.7%	20%	30.8%
Poulets	66.7%	66.7%	66.7%

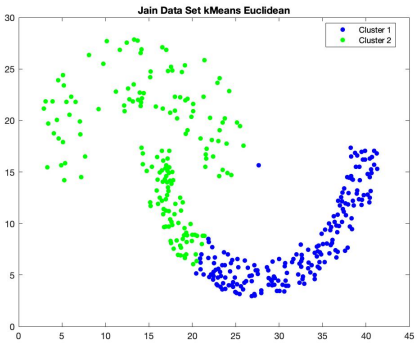
Le F-Score final ou **Macro F-Score** sera alors la moyenne des 3 F-Score de chaque classe ici égal à 46.5%.

6.2.2 Test des performances de l'algorithme des k-Means

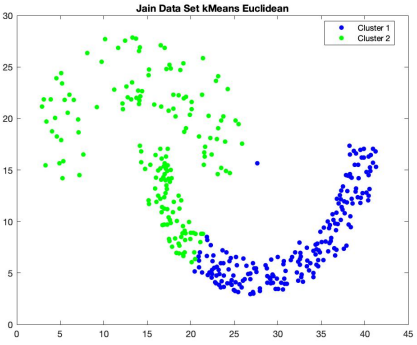
Les paramètres de l'algorithme kMeans sont le nombre d'itération ainsi que le type de distance utilisé, comme la distance euclidienne **"Euclidean"** ou L_1 **"CityBlock"**. Les formules des différentes types de distance utilisées par MATLAB est accessible dans sa documentation.



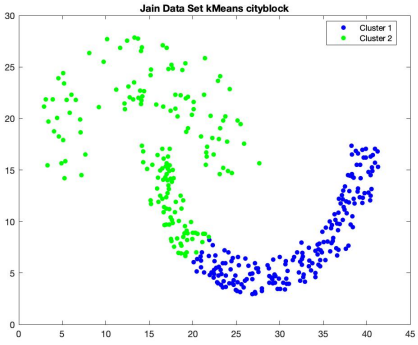
(a) Jain Data Set



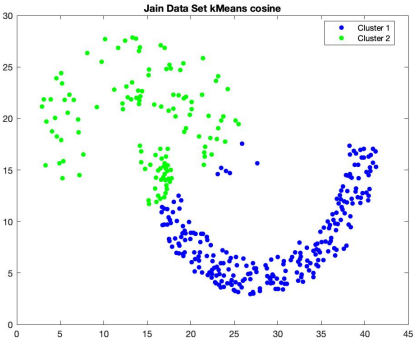
(b) kMeans Distance Euclidean sur Jain



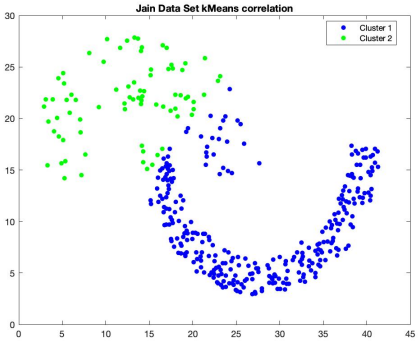
(c) kMeans Distance Square Euclidean sur Jain



(d) kMeans Distance CityBlock sur Jain



(e) kMeans Distance Cosine sur Jain



(f) kMeans Distance Correlation sur Jain

Distance	F-Score
Euclidean	0.8137
SquareEuclidean	0.8288
CityBlock	0.8333
Correlation	0.9455
Cosine	0.9195

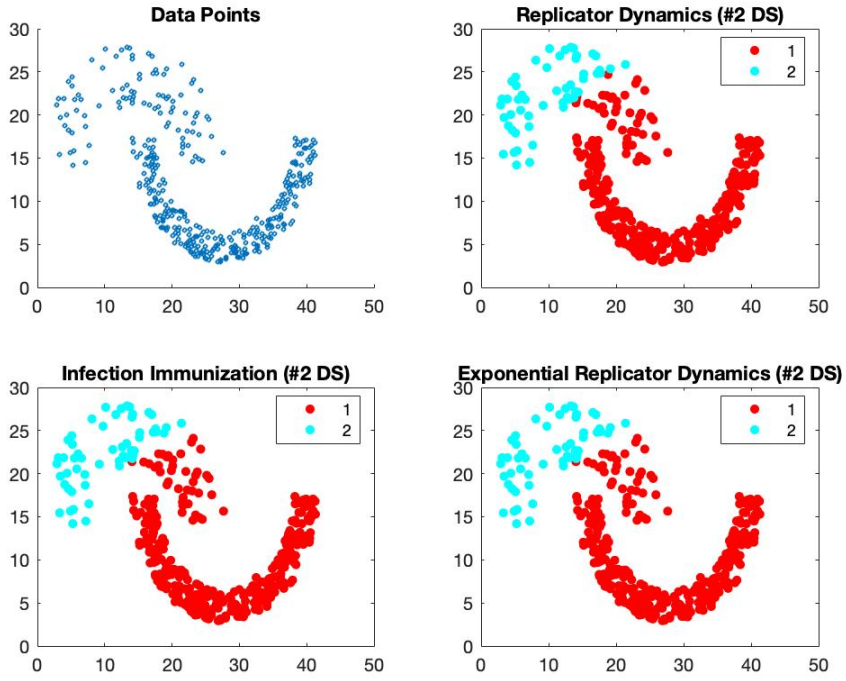
On utilise aussi l'algorithme des kMeans sur le jeu de données **Iris** à quatre dimensions avec 3 classes : "viriginica", "setosa" et "versicolor".

Distance	F-Score
Euclidean	0.8137
SquareEuclidean	0.8288
CityBlock	0.8333
Correlation	0.9455
Cosine	0.9195

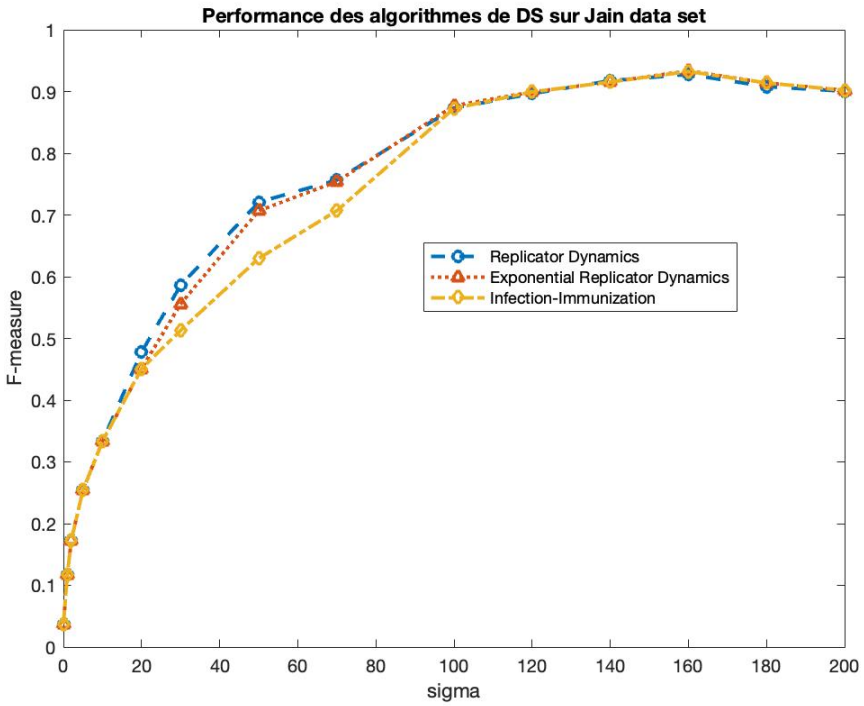
Tout comme pour le jeu de données **Jain**, l'algorithme des kMeans a de meilleures performances avec un F-Score plus élevé.

6.2.3 Test des performances de l'algorithme de l'ensemble dominant

L'algorithme de l'ensemble dominant utilise la matrice d'affinité comme paramètre d'entrée. Avec l'affinité couramment utilisée $exp(\frac{-d(x,y)}{\sigma})$, le seul paramètre est donc en fait σ. En pratique, $\sigma = \lambda * moyenne(D)$ où D est la matrice distance nécessaire pour calculer la matrice d'affinité et λ une constante. Le σ optimal, c'est-à-dire celui qui maximise la performance de l'algorithme peut varier d'un jeu de données à un autre, il est donc nécessaire de le trouver empiriquement pour chaque jeu. La performance ou F-mesure n'est pas robuste au paramètre σ, mais il existe des méthodes d'égalisation d'histogramme que l'on peut effectuer sur la matrice d'adjacence pour rendre l'algorithme robuste au paramètre σ.[9] Dans notre cas nous avons plutôt cherché à chaque fois la valeur du σ optimal.

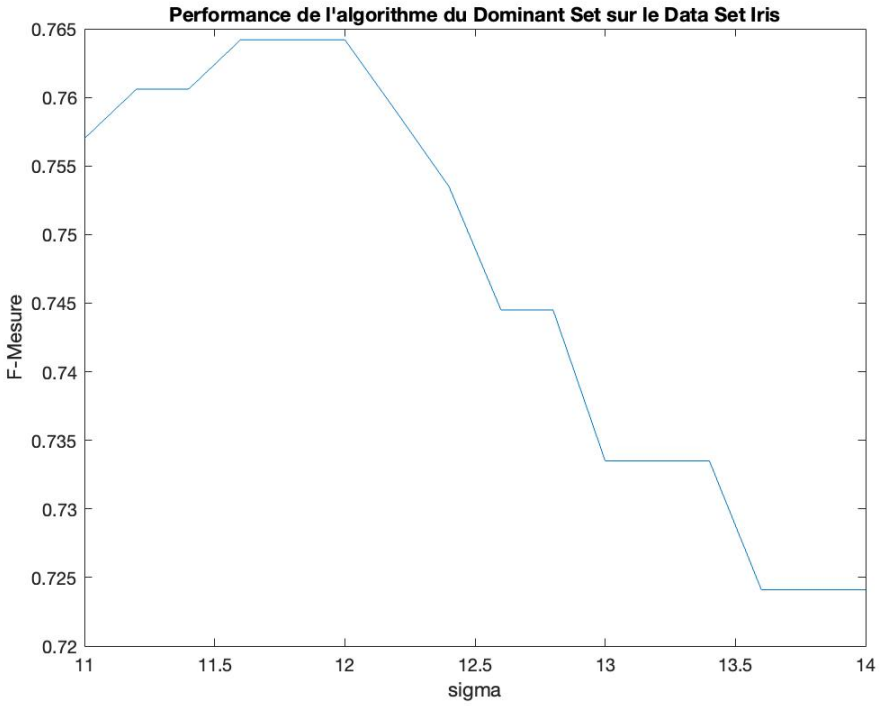


(g) Dominant Set sigma=160 sur Jain Data Set



(h) F-Measure de l'Algorithme Dominant Set sur Jain Data Set

Sur le jeu de données Jain, le F-Score maximal observé est de 0.9324 donc il est moins performant que l'algorithme des kMeans avec la distance corrélation.



(i) F-Score maximal de 0.76 pour sigma = 12*mean(D)

L'algorithme de l'ensemble dominant est aussi moins performant pour la prédiction des classes du jeu de données Iris que l'algorithme des kMeans.

6.2.4 Conclusion

Les performances de l'algorithme de l'ensemble dominant ne semblent pas surclasser de très loin celle d'autres algorithmes. Pour le clustering sur Jain et Iris l'algorithme des kMeans fait mieux. Pour la segmentation d'image l'algorithme des ensembles dominants fait mieux que ncut ou kMeans. Cependant notre étude de cas n'est pas assez vaste, on aurait pu expérimenter l'algorithme sur plus de jeux de données dont certains avec un nombre de cluster supérieurs ou d'autres images plus grandes. Enfin la F-Mesure n'est pas la mesure absolue de la performance d'un algorithme de clustering, il existe d'autre moyens comme l'index Jaccard ou l'indice de Rand.

7 Annexe

7.1 Représentation matricielle d'images

Un ordinateur n'est pas capable de lire directement une image brute mais il lit des matrices qui représentent cette image. Les images numériques sont effectivement décomposées en pixels (pictures elements) tous attribués d'une couleur. Plus le nombre de pixels sera élevé, plus la qualité de l'image sera bonne.

L'image de gauche contient plus de pixels que celle de droite et le logo est donc de meilleur qualité.

Il existe plusieurs modèles de codage d'une couleur, les modèles RGB (Red Green Blue) et HSV (High Saturation Value) en sont des exemples.

Le modèle RGB reconstitue une couleur par synthèse additive à partir des trois couleurs primaires: le rouge, le vert et le bleu. Cette méthode associe à chaque pixel un triplet dont la première composante est le pourcentage de rouge, la seconde de vert et la troisème de bleu (ou un entier compris entre 0 et 255).

Le modèle HSV associe également à chaque pixel un triplet mais reconstitue les couleurs à partir de la teinte, la saturation et la valeur. La valeur de la teinte est celle de l'angle associé à la coueleur du pixel sur le cercle chromatique. La saturation est un pourcentage qui correspond à l'intensité de la couleur étudiée. La composante correspondant à la valeur est aussi un pourcentage mais qui reflète la "brillance" de la couleur, plus ce pourcentage sera petit, plus le pixel sera sombre.

Pour lire une image, la plupart des langages informatiques transforment l'image en une matrice de triplets, chaque coefficient de cette matrice correspond aux coordonnées RGB du pixel associé à ce coefficient. Ces langages proposent souvent des fonctions qui permettent d'obtenir les coordonnées HSV d'une image à partir de ses coordonnées RGB.

En Python, nous représenterons les images commme des matrices tridimensionnelles de taille m x n ou m est le nombre de pixels par colonne de l'image, et n le nombre de pixels par ligne.

7.2 Méthodes de segmentation d'images

La segmentation d'images est un proccedé qui consiste à rassembler des pixels "proches" entre eux, à trouver ainsi les lignes naturelles d'une image pour la partitionner le long de celles-ci. Bien que nous n'ayons aucun problème, en tant qu'humain, à séparer les différents objets d'une image, il faut beaucoup de travail à un ordinateur pour partitionner une image et la recherche d'algorithmes performants pour la segmentation d'images est d'actualité.

Les méthodes de segmentation d'images sont réparties dans quatre domaines:

- la segmentation fondée sur les régions: l'algorithme manipule directement les régions qu'il va modifier;
- la segmentation fondée sur les contours: en s'appuyant sur le fait que la différence entre deux régions est marquée, l'algorithme va chercher les frontières des régions;
- la segmentation fondée sur la classification des pixels: l'algorithme regarde les relations entre chaque pixel avec le reste de l'image afin de les classifier;
- la segmentation fondée sur les trois segmentations précedentes: l'algorithme segmente automatiquement l'image à partir des autres méthodes.

7.3 K-means

On va chercher dans l'ensemble des observation $(x_1, ..., x_n)$ qui peuvent être les pixels d'une image à identifier des clusters $(S_1, ..., S_k)$ à l'aide de k centroïdes, $k \leq n$. L'objectif est de minimiser la somme des carrés des distances intra-classe : $argmin_S \sum_{i=1}^{k} \sum_{x \in S_i} ||x - u_i||^2$, u_i étant le pixel moyen de la classe S_k. L'algorithme fonctionne donc ainsi:

i) On choisit le nombre de clusters K

ii) On choisit aléatoirement K centroïdes parmi les points

iii) On itère jusqu'à ce que les centroïdes ne changent plus

iv) A chaque itération on affecte les observations à un cluster

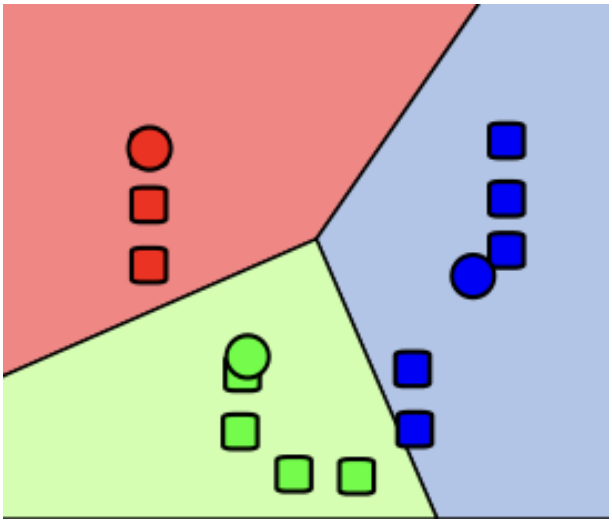
$$\mathrm{S}_i^{(t)} = \left\{ \mathbf{x}_j : \left\| \mathbf{x}_j - \mathbf{m}_i^{(t)} \right\| \leq \left\| \mathbf{x}_j - \mathbf{m}_{i^*}^{(t)} \right\| \ \forall \ i^* = 1, \ldots, k \right\}$$

On remet aussi à jour la moyenne des clusters

$$\mathbf{m}_i^{(t+1)} = \frac{1}{|S_i^{(t)}|} \sum_{\mathbf{x}_j \in S_i^{(t)}} \mathbf{x}_j$$

On peut appliquer cet algorithme à des nuages de points ou à des images. Une image est une matrice de dimension MxNx3 (RGB). On reshape l'image en matrice M*Nx3 (RGB) et on applique la méthode KMeans sur une intensité de couleur (axis = 1 en Python par exemple).

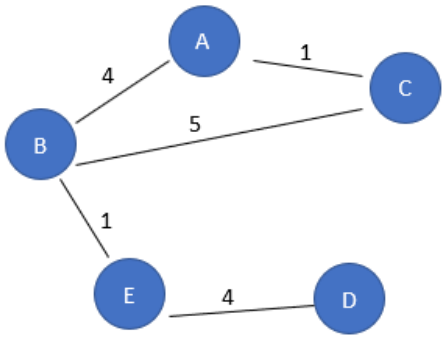
Figure 14: Output : Exemple de 3 clusters avec nuage de points



7.4 Une autre méthode de segmentation d'image: l'algorithme de Shi et Malik

7.4.1 Matrice de degrés

La matrice de degrés du graphe G est la matrice diagonale D de taille N x N où les coefficients diagonaux sont donnés par $d_{i,i} = \sum_{j=1}^{N} w_{i,j}$



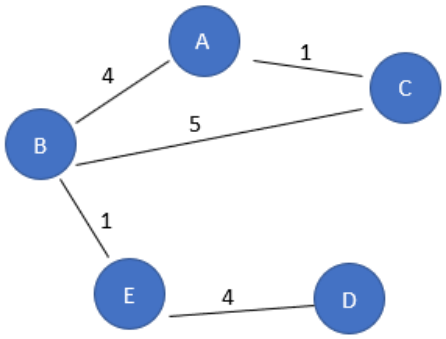
La matrice de degrés D du graphe donné en exemple est:

$$D = \begin{pmatrix} 4+1+0+0 & 0 & 0 & 0 & 0 \\ 0 & 4+5+0+1 & 0 & 0 & 0 \\ 0 & 0 & 1+5+0+0 & 0 & 0 \\ 0 & 0 & 0 & 0+0+0+4 & 0 \\ 0 & 0 & 0 & 0 & 0+1+0+4 \end{pmatrix} = \begin{pmatrix} 5 & 0 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 & 0 \\ 0 & 0 & 6 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 0 & 5 \end{pmatrix}$$

Le coefficient $d_{i,i}$ est appelé degré du sommet i, il correspond à la somme des poids des arêtes qui partent du sommet i.

7.4.2 Matrice laplacienne

La matrice laplacienne de G est la matrice L de taille N x N donnée par $L = D - A$ où A est la matrice d'adjacence du graphe et D sa matrice de degrés.



Dans cet exemple, la matrice laplacienne est:

$$L = \begin{pmatrix} 5 & 0 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 & 0 \\ 0 & 0 & 6 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 0 & 5 \end{pmatrix} - \begin{pmatrix} 0 & 4 & 1 & 0 & 0 \\ 4 & 0 & 5 & 0 & 1 \\ 1 & 5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 4 \\ 0 & 1 & 0 & 4 & 0 \end{pmatrix} = \begin{pmatrix} 5 & -4 & -1 & 0 & 0 \\ -4 & 10 & -5 & 0 & -1 \\ -1 & -5 & 6 & 0 & 0 \\ 0 & 0 & 0 & 4 & -4 \\ 0 & -1 & 0 & -4 & 5 \end{pmatrix}$$

7.4.3 Coupe d'un graphe

On dit qu'on effectue une coupe sur G lorsqu'on séepare en deux sous-graphes $G1 = (V_1, E_1)$ et $G2 = (V_2, E_2)$ les sommets de G de telle manière que :

- $V_1 \cup V_2 = V$
- $V_1 \cap V_2 = \varnothing$

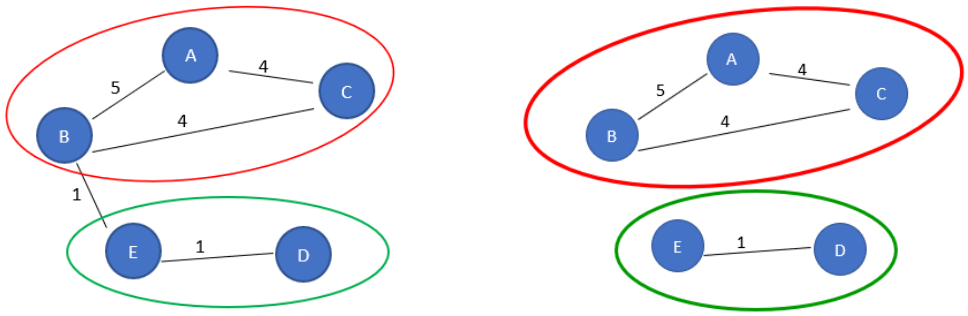
On appelle coupe l'ensemble des arêtes reliant un sommet de $V1$ à un sommet de V_2. Le poids de la coupe est alors donné par:

$$\text{cut}(V_1, V_2) = \sum_{i \in V_1, j \in V_2} w_{i,j}$$

Le poids d'une coupe permet de quantifier la différence entre deux sous-graphes, plus elle est petite,plus les deux sous-graphes obtenus sont distincts.

7.4.4 Segmentation du graphe

Pour distinguer les différents groupes homogènes d'un graphe G on peut utiliser une méthode de partitionnement. Cela consiste à diviser G en deux (ou plus) sous-graphes tout en regardant le poids de la coupe afin d'obtenir des sous-graphes cohérents. Une de ces méthodes est l'algorithme du min cut (de la coupe minimale), l'objectif est de partitionner G en deux sous-graphes V_1 et V_2 de telle manière que cut(V_1,V_2) soit minimisée. Cette méthode est cependant limitée car l'algorithme peut renvoyer des coupes avec des noeuds isolés, puisqu'il ne tient compte que du poids de la coupe et non de la taille des sous-graphes.



(a) graphe initial (b) graphe après la transformation

Dans cet exemple, la coupe sera l'arête (B, E) *et le poids de cette coupe sera de 1*

7.4.5 Méthode du "Normalized-cut"

Pour éviter le partitionnement d'un graphe en de trop petits sous-ensembles, Shi et Malik proposent une autre mesure du poids d'une coupe dite normalized cut (N-cut) et donnée par: La méthode de Shi et Malik consiste à segmenter une image en effectuant des "coupes" sur son graphe.Dans un souci d'optimisation, l'algorithme choisira d'effectuer la coupe dont le poids est minimal, on la choisira donc de la manière suivante:

- calculer la matrice de degrés D et la matrice laplacienne L du graphe;
- calculer les valeurs propres de la matrice $D^{-1/2}LD^{-1/2}$;
- calculer un vecteur propre v associé a la deuxième valeur propre la plus petite de $D^{-1/2}LD^{-1/2}$;
- extraire les indices des coefficients positifs de v, ces indices correspondent aux sommets de la première composante connexe;
- les indices des coefficients restants (qui sont les coefficients négatifs) correspondent aux sommets de la seconde composante connexe.

7.4.6 Autre

La matrice de degrés D associée au graphe de l'image est diagonale et pourra donc être implémentée sous la forme d'un tableau NumPy sous Python afin d'optimiser l'analyse.

Etapes de construction de A et D

- Initialisation: on définit d'abord A comme une matrice creuse de taille $mn \times mn$ et D comme un vecteur de longueur mn;
- Pour chaque sommet i, $i = 1, ..., nm - 1$:
  1. On extrait d'abord l'ensemble J_i défini par $\{\text{sommets } j / \|X(i) - X(j)\| < r\}$;
  2. Pour chaque sommet j de J_i, on calcule le poids $w_{i,j}$;
  3. On définit la $i^{ème}$ composante de D par $d_i := \sum_{j \in J_i} w_{i,j}$.

Cet algorithme est relativement simple, bien qu'extraire le voisinnage J_i de chaque pixel i puisse être laborieux mais ne requiert que de connaître le rayon r et la taille de l'image originale (m x n). Nous avons donc implémenter la fonction suivante qui pour chaque pixel i retourne son voisinnage J_i:

7.4.7 Remerciements

Nous tenons à remercier Monsieur Patrice Bertrand pour avoir supervisé ce mémoire et nous avoir apporté son aide durant l'année. Nous tenons également à remercier Gabrielle de Chastenet pour son apport sur la théorie des graphes et l'algorithme du N-cut.

References

[1] Marcello Pelillo, Massimiliano Pavan (2003), *A new graph-theoretic approach to clustering and segmentation*, DOI: 10.1109/CVPR.2003.1211348, IEEE Xplore.

[2] Marcello Pelillo, Massimiliano Pavan, *M.: Generalizing the Motzkin–Straus theorem to edge-weighted graphs, with applications to image segmentation*, International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition

[3] P. F. Felzenszwalb et D. P. Huttenlocher. *Efficient graph-based image segmentation.* International Journal of Computer Vision, 2004

[4] T. S. Motzkin et E. G. Straus. Maxima for graphs and a new proof of a theorem of Turan. Canad. J. Math., 17:533–540, 1965. p 486, 493

[5] A. K. Jain et R. C. Dubes. Algorithms for Clustering Data. Prentice Hall, Englewood Cliffs, NJ, 1988.

[6] D. G. Luenberger. Linear and Nonlinear Programming. Addison-Wesley, Reading, MA, 1984.

[7] Panos Y. Papalambros et Douglass J. Wilde, Principles of optimal design: Modeling and computation, Cambridge University Press, 2000 p185

[8] Samuel Rota Bulò, Immanuel M. Bomze et Marcello Pelillo (2010), *A Fast Population Game Dynamics for Dominant Sets and Other Quadratic Optimization Problems* , pages 275-285, Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR).

[9] Jian Hou, Weixue Liu (2017), *Parameter Independent Clustering Based on Dominant Sets and Cluster Merging*, https://doi.org/10.1016/j.ins.2017.04.006, Elservier, Information Sciences, Volume 405, Pages 1-17.

[10] Francesca Pennacchietti, Sebastiano Vascon, Thierry Nieus, Christian Rosillo, Sabyasachi Das, Shiva Tyagarajan, Alberto Diaspro, Alessio del Bue, Enrica Maria Petrini, Andrea Barberis, et Francesca Cella Zanacchi. *Nanoscale molecular reorganization of the inhibitory postsynaptic density is a determinant of gabaergic synaptic potentiation.* Journal of Neuroscience, 2017.

[11] Sebastiano Vascon, Eyasu Zemene Mequanint, Marco Cristani, Hayley Hung, Marcello Pelillo, et Vittorio Murino. *Detecting conversational groups in images and sequences: A robust game-theoretic approach.* Computer Vision and Image Understanding, 143:11–24, 2016.

[12] Sebastiano Vascon, Eyasu Zemene Mequanint, Marco Cristani, Hayley Hung, Marcello Pelillo, et Vittorio Murino. *A game-theoretic probabilistic approach for detecting conversational groups.* Asian Conference in Computer Vision, Lecture Notes in Computer Science, 2014.

[13] L. Dodero, S. Vascon, L. Giancardo, A Gozzi, D. Sona, et V. Murino. *Automatic white matter fiber clustering using dominant sets.*. Pattern Recognition in Neuroimaging (PRNI), 2013 International Workshop on, pages 216–219, June 2013.

[14] Luca Dodero, Sebastiano Vascon, Vittorio Murino, Angelo Bifone, Alessandro Gozzi, et Diego Sona. *Automated multi-subject fiber clustering of mouse brain using dominant sets.* Frontiers in Neuroinformatics, 8:87, 2015.

[15] Sebastiano Vascon, Vittorio Murino, Samuel Rota Bulo et Marcello Pelillo. *DSLib: An open source library for the dominant set clustering method.* 2020