Semaine d'Etude Mathématiques et Entreprises 2 : Analyse de grands volumes de données en grande dimension

Eric Dalissier, Charles Dapogny, Sofiane Hendili, Lise-Marie Imbert-Gérard, Thomas Pradeau

Semaine d'Etude Maths–Entreprises 2

28 Novembre - 2 Décembre 2011, Université Lyon I

Analyse de grands volumes de données en grande dimension

E. Dalissier[a] C. Dapogny[b]
S. Hendili[c] L.-M. Imbert-Gérard[b]
T. Pradeau[d]

[a] *Institut Camille Jordan, Insa de Lyon, 69621 Villeurbanne, France.*
[b] *Laboratoire Jacques-Louis Lions, UPMC, 75252 Paris, France.*
[c] *Institut de Mathématiques et de Modélisation de Montpellier, Université Montpellier 2, 34095 Montpellier, France.*
[d] *CERMICS - ENPC, 77455 Marne la Vallée, France.*

Sujet proposé par :

PICVIZ® LABS

Correspondants : P. Saadé (Picviz Labs)

Maths & Entreprises — cnrs dépasser les frontières — a m i e s

Numéro de publication : SEME002-2011-11-B

Le problème considéré ici provient de l'analyse de journaux d'événements informatiques. Ces journaux constituent des jeux de données en grande dimension, qui peuvent contenir de l'information sur d'éventuelles attaques contre un réseau. A l'heure actuelle la recherche d'événements extraordinaires s'applique mal à la sécurité des systèmes d'information. En effet, les points de vue sont souvent partiels, ils ne permettent pas de comprendre les phénomènes de grande échelle et la prise en compte de la globalité des informations est très complexe. De plus la visualisation de données est problématique en grande dimension.

Deux contraintes incontournables sont à prendre en compte. D'une part les systèmes d'informations contiennent une masse de données considérable en permanence croissante, leur temps de stockage est donc court car très coûteux et leur traitement doit donc nécessairement être rapide. D'autre part les éléments rares que l'on cherche à isoler ne sont pas modélisés car ils peuvent prendre sans cesse de nouvelles formes, l'expertise humaine devient alors irremplaçable pour les détecter. Le but de ce travail est de présenter quelques réflexions sur l'extraction d'information depuis des jeux de données en grande dimension. Pour donner une vision globale des informations il faut restructurer l'information pour lui conférer du sens et la rendre accessible à la compréhension humaine. En pratique l'objectif est d'illustrer certaines propriétés - qui restent à définir - des données pour permettre à un expert d'identifier des éléments rares. Il s'agit donc de donner un éclairage pertinent à une grande quantité d'informations.

Le point de vue adopté dans cette étude est géométrique. Un fichier de N lignes et d colonnes de données chiffrées est considéré comme un ensemble discret $E \subset \mathbb{R}^d$ de cardinal N. L'espace $\mathbb{R}^d$ est muni de sa métrique euclidienne. L'information géométrique pourrait permettre de repérer des grosses structures regroupant des points normaux auxquelles s'opposeraient des points isolés. Il reste alors à définir comment construire une information suffisamment globale en partant de l'information ponctuelle. Les problèmes de visualisation en grande dimension n'ont pas été abordés dans ce travail.

Dans une première partie nous proposons un algorithme pour créer un squelette structurant le nuage de points. Ce squelette est défini localement par des variétés de dimension 1 dans $\mathbf{R}^d$. Il représente localement le nuage comme une variété de dimension k, qui restera à définir. Quelques idées sur la détermination de la valeur de k seront exposées. Dans une seconde partie nous détaillerons la mise en œuvre numérique d'une partie de l'algorithme précédent, en dimension d. Nous illustrerons l'utilisation de la méthode en dimension 2 pour simplifier la visualisation. Dans une dernière partie nous présenterons une autre méthode de structuration du nuage de points, basée sur une idée de dichotomie.

1 Une première idée de structuration du nuage par recherche de ses 'lignes de force'

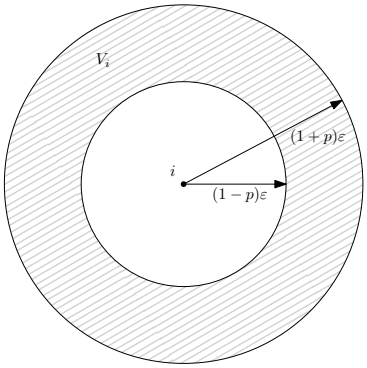
1.1 Quelques notations

Nous considérons un ensemble de N points $E \subset \mathbb{R}^d$, que l'on assimile à $[\![1, N]\!]$.

On munit $\mathbb{R}^d$ d'une métrique et on note d la distance associée. On se fixe également deux paramètres $(\epsilon, \theta) \in \mathbb{R}^2$, ainsi que $p \in [0, 1]$.

Définition 1.1 *Pour tout $(i, j) \in E^2$, on dira que i et j sont* voisins *si $(1-p)\epsilon \leq d(i, j) \leq (1+p)\epsilon$. On notera*

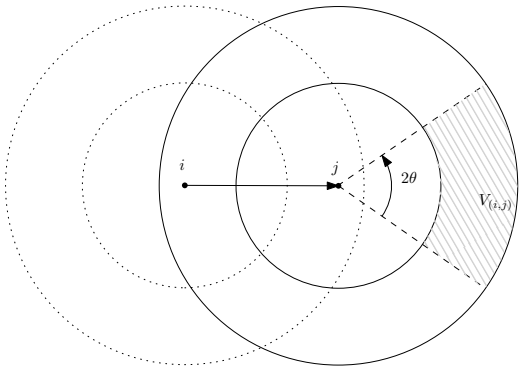
$$V_i = \{j \in E, (1-p)\epsilon \leq d(i, j) \leq (1+p)\epsilon\}$$



Définition 1.2 *Soit $(i,j) \in E^2$. Si i et j sont voisins, on définit l'*arête *orientée de i vers j, et on la note (i,j). On notera A l'ensemble des arêtes.*

Définition 1.3 *Pour une arête $(i,j) \in A$, on considère l'ensemble des points voisins de j,* dans la direction *de (i,j) comme les points $k \in V_j$ tels que l'angle entre les arêtes (i,j) et (j,k) soit au plus θ en valeur absolue.*

$$V_{(i,j)} = \left\{ k \in V_j, \frac{(i,j).(j,k)}{\|(i,j)\|\|(j,k)\|} \geq \cos\theta \right\}$$



Définition 1.4 *On appelle* ligne *une suite d'arêtes $(i_1,i_2),(i_2,i_3),\ldots,(i_{n-1},i_n)$, et on notera une telle ligne $(i_1 i_2 \ldots i_n)$. Par définition des arêtes, on a pour tout $k \in [\![2, n-1]\!], i_{k+1} \in V_{(i_{k-1},i_k)}$.*

On dira qu'une arête (i,j) est une extrémité *(par i) si $V_{(j,i)} = \emptyset$.*

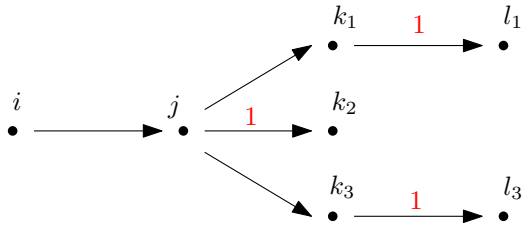
Enfin, on appelle ligne maximale *une ligne $(i_1 i_2 \ldots i_n)$ telle que (i_1,i_2) et (i_n,i_{n-1}) sont des extrémités, c'est-à-dire que $V_{(i_2,i_1)} = V_{(i_{n-1},i_n)} = \emptyset$.*

1.2 Présentation de l'algorithme de recherche de squelette

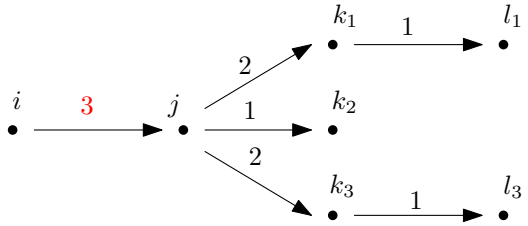
On va chercher à définir un *squelette* de la structure des données, comme les principales lignes maximales dans l'ensemble. Pour cela, on introduit la notion suivante :

Définition 1.5 *Pour chaque arête $(i,j) \in A$, on note le* poids *de l'arête $\pi_{i,j}$, qui est défini comme la longueur maximale d'une ligne pouvant être obtenue à partir de cette arête.*

Pour trouver les principales lignes maximales, on va dans un premier temps chercher toutes les lignes ayant une extrémité, en excluant donc les boucles. Pour chaque extrémité on stocke dans une liste non ordonnée les arêtes de l'arborescence obtenue à partir de cette extrémité, en veillant à ne pas prendre deux fois la même arête. Ainsi, dans cette liste, les extrémités hors celle de départ auront donc un poids égal à 1.



Ensuite, on remonte l'arborescence depuis les extrémités jusqu'à l'arête de départ, en mettant à jour les poids.



1.3 Eléments considérés

L'algorithme va considérer les éléments suivants :
- Une fonction c sur A indiquant si une arête a déjà été traitée.
- Une fonction π sur A indiquant le poids d'une arête.
- Une pile AV contenant les arêtes "à voir" : lorsque l'algorithme passe sur une arête, il met tous les voisins de cette arête dans la bonne direction au sommet de la pile.
- Une pile VU contenant les arêtes "vues" : dès que l'agorithme passe sur une arête, il la met au sommet de la pile.

1.4 Etapes de l'algorithme

Initialisation : Tout d'abord, on initialise les poids : $\pi_{i,j} = 0$
Boucle sur les sommets : Soit un sommet i_0.
Boucle sur les arêtes : Soit $(i_0, j_0) \in A$ tel que (i_0, j_0) est une extrémité par i_0.

1. **Réinitialisation** :
   (a) On fixe $c(i_0, j_0) = 1$ et $c(i, j) = 0$ pour toutes les autres arêtes.
   (b) On fixe $AV = \{(i_0, j_0)\}$ et $VU = \emptyset$
2. **Boucle sur la première pile** : Tant que $AV \neq \emptyset$:
   (a) On considère $(i, j) \in AV$, l'arête au sommet de la pile.
   (b) **Test** : Si $V_{(i,j)} = \emptyset$ ou si $\forall k \in V_{(i,j)}, c(j, k) = 1$, alors $\pi(i, j) = 1$.
   (c) **Boucle sur les voisins** : Pour tout $k \in V_{(i,j)}$, si $c(j, k) = 0$:
      i. On met (j, k) au sommet de AV,
      ii. On indique que l'arête est visitée : $c(j, k) \leftarrow 1$
   (d) On empile l'arête (i, j) au sommet de VU.

   A ce stade, on a donc dans VU les arêtes de l'arborescence obtenue à partir de (i_0, j_0), et les extrémités ont un poids $\pi = 1$. On vide maintenant la pile en actualisant les poids :
3. **Boucle sur la deuxième pile** : Tant que $VU \neq \emptyset$:
   (a) On considère $(i, j) \in VU$, l'arête au sommet de la pile.

(b) **Test** : Si $\pi(i,j) = 0$, On remet (i,j) à la fin de la pile VU.

(c) Sinon :

i. On cherche l'unique arête de VU qui finit par i, par exemple (k,i).

ii. On actualise le poids : $\pi(k,i) \leftarrow \max[\pi(k,i), \pi(i,j)+1]$.

iii. Enfin, on enlève (i,j) de VU.

Chaque ligne sera ainsi parcourue deux fois, une fois depuis chaque extrémité. Comme on garde le poids maximal, chaque arête aura bien un poids correspondant à la longueur de la plus grande ligne que l'on peut construire à partir de cette arête.

1.5 Régularisation du squelette et recouvrement d'entités de dimension supérieure

Une fois que le squelette a été créé et tracé, il est intéressant de pouvoir classer les différentes lignes maximales obtenues : en effet, on dispose seulement à ce stade d'une liste de segments reliés entre eux, qu'il est difficile de décrire par des fonctions mieux que continues. L'idée va être d'étudier (ou d'approcher) la régularité intermédiaire des segments, sachant que l'on ne sera jamais de classe $\mathcal{C}^1$. Quelques notions ont besoin d'être introduites au préalable :

(a). Quelques notions de régularité

Définition 1.6 *(régularité ponctuelle) Soit f, la fonction qui approchera le segment passant par le point x_0.*

On dit que $f \in \mathcal{C}^\alpha(x_0)$, si $\exists C > 0, \delta > 0, P$ avec $\deg(P) < \alpha$ tel que :

$$|x - x_0| \leq \delta \Rightarrow |f(x) - P(x - x_0)| \leq C|x - x_0|^\alpha \tag{1}$$

On peut donc essayer d'approcher les lignes du squelette par des fonctions de différentes classes avec l'encadrement minimal.

Définition 1.7 *On dit qu'une fonction f est uniformément höldérienne, $f \in \mathcal{C}^\alpha(\mathbb{R}^d)$, si*

$$\exists C > 0, \text{ tel que } \forall t, \exists P_t(u)\ ,\ |f(u) - P_t(u)| \leq C|t - u|^\alpha \tag{2}$$

À l'aide de cette notion de régularité ponctuelle, et de fonctions uniformément höldériennes, on peut classer les différents lignes du squelette. Seulement il peut être difficile voire impossible de calculer de telles fonctions surtout lorsque l'on doit traiter un grand nombre de données car il existe une infinité de fonctions possibles ainsi que de nombreux encadrements. On peut donc essayer d'utiliser un autre outil, la *dimension de Hausdorff*.

(b). La dimension de Hausdorff

Soit X un espace métrique. On note $\Omega(X,\epsilon)$ l' ensemble des parties ω de X de diamètre $\delta_\omega \leq \epsilon$.

Définition 1.8 *La mesure de Hausdorff d-dimensionnelle de X est définie par :*

$$m(X,d) = \lim_{\epsilon \to 0} \left(\inf_{\{\omega_i\}_{i \in I} \subset \Omega(X,\epsilon)} \sum_{i \in I} \delta_{\omega_i}^d \right) \tag{3}$$

Cette mesure peut être finie ou infinie suivant la valeur de d.

FIGURE 1 – Représentation d'un recouvrement de lignes par des boules de diamètre δ_ω

L'idée sera donc de trouver la mesure de Hausdorff associée aux lignes du squelette. Pour cela, il faut recouvrir les lignes par des ouverts, le plus simple étant des boules de taille très petites $\delta_\omega \leq \epsilon$ (figure 1). La difficulté et le temps de calcul vont être de déterminer cette taille δ_ω.

Une fois que l'on dispose de la mesure de Hausdorff associée à un espace de dimension d, on peut déterminer sa *dimension*.

Proposition-définition 1.1 *Il existe un réel d_c tel que :*
- $m(X, d) = 0$ *si* $d < d_c$,
- $m(X, d) = +\infty$ *si* $d > d_c$.

Ce nombre s'appelle la dimension de Hausdorff de l'espace X.

Le but sera de trouver pour chaque ligne du squelette cette valeur d_c de transition de la mesure de Hausdorff, qui définit sa dimension de Hausdorff.

Ce processus requiert un long temps de calcul, mais moins important que l'approche par des fonctions. On peut demander à l'utilisateur de donner une valeur de la dimension de Hausdorff dont il souhaite voir apparaître le squelette. Les temps de calculs seront alors acceptables.

Si la quantité de données n'est pas trop importante, on peut calculer assez rapidement la mesure de Hausdorff. En effet, dans notre méthode l'espace entre deux points est connu, on peut alors facilement l'approcher par des boules petites de taille δ_ω^d.

Ce type de méthode est très utilisé dans l'analyse multifractale ou dans les ondelettes.

2 Mise en oeuvre de l'algorithme de recherche de squelette dans un cas 2D

L'objectif est de considérer l'implémentation des premières étapes de l'algorithme décrit précédemment. Un programme a été rédigé sous Matlab afin d'exécuter les premières étapes de l'algorithme : il permet de parcourir et de tracer toutes les chaînes contenant deux arêtes.

Nous avons choisi à dessein de tester la mise en œuvre en dimension 2. Cela nous a permis de mieux comprendre les difficultés inhérentes au parcours de données en s'affranchissant de la complexité liée à la grande dimension de données traitées.

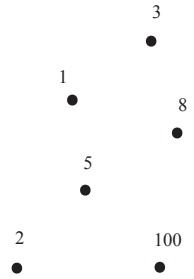
2.1 Principales étapes de l'implémentation

Cette description est donc illustrée sur un exemple en dimension 2, mais tout ce qui suit reste juste en dimension supérieure.

- **Déterminer l'ensemble des points i tels que $\mathbf{V_i} \neq \emptyset$**
  Afin d'optimiser le stockage de ces points, la méthodologie suivante a été adoptée : en parcourant dans un ordre croissant la numérotation des points on construit un tableau qui contient :

– dans la première colonne les numéros des points i tels que $V_i \neq \emptyset$,
– dans la deuxième colonne les numéros des points appartenant à chaque $V_i \neq \emptyset$.
La figure suivante illustre la structure de ce tableau.

i	j	c_1	c_2	d
1	3	0	0	$d_{1,2}$
1	5	0	0	$d_{1,5}$
1	8	0	0	$d_{1,8}$
2	5	0	0	$d_{2,5}$
2	100	0	0	$d_{2,100}$
⋮	⋮	⋮	⋮	⋮



Remarques :
- Le nombre de lignes de ce tableau correspond au nombre total d'arêtes (i, j) avec $i < j$.
- Les coefficients c_1 et c_2, stockés dans les colonnes 3 et 4 du tableau, interviendront dans la prochaine étape de l'algorithme. A ce stade du programme ils sont tous initialisés à zéro.
- La longueur $d_{i,j}$ de chaque arête (i,j) est stockée dans la dernière colonne du tableau.

– **Déterminer et identifier l'ensemble des arêtes (i,j) telles que $\mathbf{V_{i,j} \neq \emptyset}$**
Cette partie de l'algorithme est réalisée en deux étapes :
– déterminer pour chaque arête (i_k, j_k), les éléments de V_{i_k,j_k}.
– déterminer pour chaque arête (j_k, i_k), les éléments de V_{j_k,i_k}.
Le nombre d'éléments de V_{i_k,j_k} (resp. V_{j_k,i_k}) est stocké dans la troisième colonne (resp. quatrième colonne) de la ligne k.

L'exécution du code est détaillée dans l'exemple suivant (où l'on a pris $p = 0.3$) : on considère 8 points, numérotés de 1 à 8 dont la disposition est illustrée dans la figure 2.

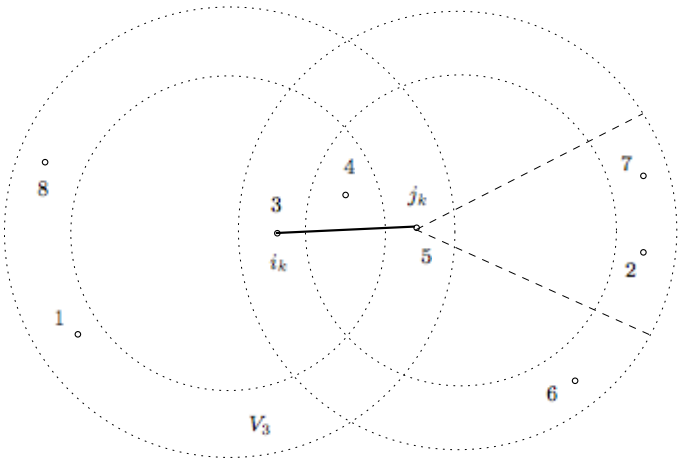


FIGURE 2 – Disposition des points sous l'hypothèse : d(3,5)=ε

La première étape de l'algorithme permet de générer le tableau suivant :

i	j	c_1	c_2	d
1	3	0	0	$d_{1,3}$
2	5	0	0	$d_{2,5}$
3	5	0	0	$d_{3,5}$
3	8	0	0	$d_{3,8}$
5	6	0	0	$d_{5,6}$
5	7	0	0	$d_{5,7}$

Ce tableau est noté tab1 dans le code et le nombre de ses lignes, qui correspond aux nombre d'arêtes, est noté k.

Remarque : le point numéro 4 n'apparaît pas dans le tableau étant donné qu'il ne possède pas de voisin, i.e. il n'existe pas de point P qui vérifie

$$0.7\varepsilon \leq d(4, P) \leq 1.3\varepsilon.$$

Pour la deuxième étape on effectue :

– Un parcours des arêtes orientées dans le sens croissant (numéro du point de départ $<$ numéro du point d'arrivée) en effectuant une boucle sur les numéros des arêtes. A chaque itération :

i) Le point de départ de l'arête est noté *ii* et son point d'arrivée *jarete*.
Exemple : à la troisième itération, l'arête considérée est l'arête (3,5). On aura donc ii=3 et $jarete$=5.
Dans la suite on se reportera toujours à cette itération lorsqu'on illustrera une étape par un exemple.

ii) Les numéros des arêtes qui ont pour point de départ *jarete* et un numéro de point d'arrivée supérieur à *jaret* (resp. inférieur à *jaret*) sont déterminés et stockés dans un tableau noté z1 (resp. z2).
Exemple : Les arêtes candidates, dont les numéros sont stockés dans les tableaux z1 et z2, sont respectivement les arêtes (5,6) et (5,7), et les arêtes (5,2) et (5,3). On en déduit le contenu des tableaux z1 et z2 :

$$z1 = [6, 7] \ ; \ z2 = [2, 3].$$

Remarque : L'arête (3,5) est sélectionnée mais sera éliminée automatiquement à la prochaine étape.

iii) Une boucle sur les éléments de z1 est réalisée ; à chaque itération ℓ :
a) Le numéro du point d'arrivée de l'arête numéro z1(ℓ) est déterminé et noté *jj*,
b) Un test est effectué pour vérifier si $jj \in V_{ii,jaret}$ en calculant

$$cos\left(\widehat{(ii, jaret), (jaret, jj)}\right) = \frac{< (ii, jaret) \ , \ (jaret, jj) >}{||(ii, jaret)|| \cdot ||(jaret, jj)||}$$

et en vérifiant si

$$cos\left(\widehat{(ii, jaret), (jaret, jj)}\right) > cos\theta$$

Si le test est vérifié alors la valeur de c1 de l'arête (ii,jaret) est incrémentée de 1.

Exemple : A la fin de cette boucle, la valeur de c1 de l'arête (3,5) est 1. Cette valeur indique que le point numéro 7 a été comptabilisé comme voisin, i.e $7 \in V_{3,5}$. Le point numéro $6 \notin V_{3,5}$, il est donc cohérent que le point 6 n'ait pas été comptabilisé. Le point numéro $2 \in V_{3,5}$ mais n'a pas encore été comptabilisé. Il le sera au cours de la prochaine boucle.

A la fin de cette boucle, le tableau tab1 contient les valeurs suivantes :

i	j	c_1	c_2	d
1	3	1	0	$d_{1,3}$
2	5	0	0	$d_{2,5}$
3	5	1	0	$d_{3,5}$
3	8	0	0	$d_{3,8}$
5	6	0	0	$d_{5,6}$
5	7	0	0	$d_{5,7}$

iv) Une boucle sur les éléments de z2 est réalisée ; à chaque itération ℓ :

a) Le numéro du point d'arrivée de l'arête numéro z2(ℓ), arête considérée dans le sens inverse i.e. numéro point départ $>$ numéro point d'arrivée, est déterminé et noté jj.

b) Le test iii) b) est effectué pour vérifier si $jj \in V_{ii,jaret}$. Si le test est vérifié alors la valeur de c1 de l'arête (ii,jaret) est incrémentée de 1.
Exemple : A la fin de cette boucle, la valeur de c1 est 2. La valeur de c1 a été incrémentée car le point numéro 2 a été retenu comme élément de $V_{3,5}$.

A la fin de cette boucle, le tableau tab1 contient les valeurs suivantes :

i	j	c_1	c_2	d
1	3	1	0	$d_{1,3}$
2	5	1	0	$d_{2,5}$
3	5	2	0	$d_{3,5}$
3	8	0	0	$d_{3,8}$
5	6	0	0	$d_{5,6}$
5	7	0	0	$d_{5,7}$

– Un parcours des arêtes orientées dans le sens décroissant. Chaque itération contient exactement les mêmes étapes que lors du parcours des arêtes orientées dans le sens croissant. A l'issue de ce parcours le tableau contient les valeurs suivantes :

i	j	c_1	c_2	d
1	3	1	0	$d_{1,3}$
2	5	1	0	$d_{2,5}$
3	5	2	2	$d_{3,5}$
3	8	0	1	$d_{3,8}$
5	6	0	0	$d_{5,6}$
5	7	0	1	$d_{5,7}$

Ce programme permet donc bien de parcourir toutes les suites de deux arêtes vérifiant le critère d'angle proposé. Pour les compter il suffit alors de sommer tous les coefficients $c1$ et $c2$ de tab1. Ces suites peuvent être tracées au fil du parcours. Dans l'état actuel du code, elles ne sont cependant pas stockées en mémoire.

Remarque : La mise en œuvre de la suite de l'algorithme décrit dans la première partie requiert un traitement attentif des informations manipulées. En particulier il faut créer une structure adaptée pour garder en mémoire les suites d'arêtes constituant les lignes structurantes. C'est la principale étape suivante de l'algorithme à mettre en œuvre.

2.2 Résultats numériques : l'importance des paramètres

Afin d'insister sur la pertinence de cette démarche, on souhaite illustrer le fonctionnement du code décrit au paragraphe précédent sur un exemple assez intuitif. On considère donc un ensemble de

$N = 500$ points équi-répartis dans un anneau $\mathcal{R}$ défini par :

$$\mathcal{R} = \left\{(x, y) \in \mathbb{R}^2 \setminus 0.8 \leq ||(x, y)|| \leq 1\right\}.$$

En effet les résultats permettront aisément de voir si l'on réussit à retrouver la structure par-

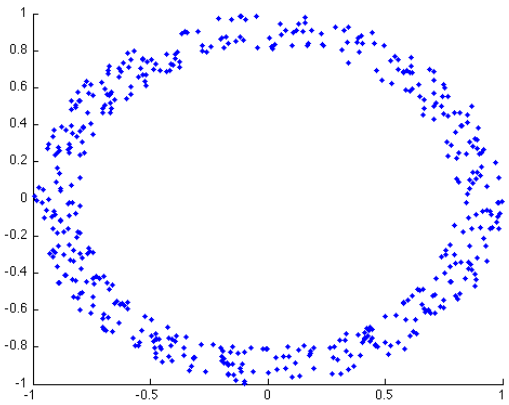


FIGURE 3 – Echantillon de 500 points équi-répartis dans $\mathcal{R}$.

ticulière de l'anneau. L'ensemble de points est représenté sur la figure 3. Le fichier de données correspondant a été créé avec Matlab.

Les figures 4 5 et 6 représentent toutes les arêtes sélectionnées par le programme, pour différents jeux de paramètres (ε, α).

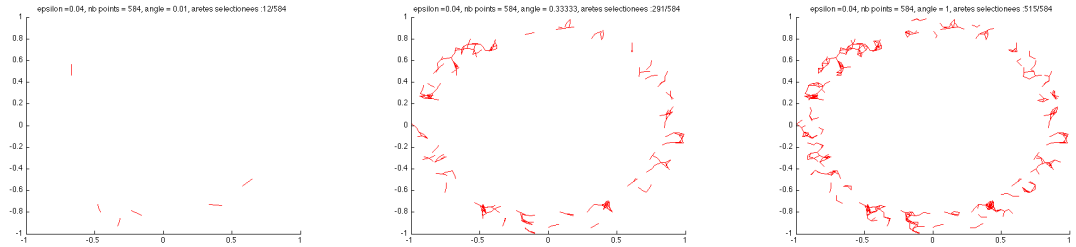


FIGURE 4 – Ensemble des arêtes sélectionnées avec $\varepsilon = 0.04$ et respectivement $\alpha = \pi/100$, $\alpha = \pi/3$ et $\alpha = \pi$.

On observe sur la figure 4 que si le paramètre ε est trop petit alors les suites de deux arêtes ne sont pas suffisantes pour retrouver la structure de l'anneau. On imagine que c'est le rapport entre ε et une taille caractéristique de la structure du nuage, ici la largeur de l'anneau, qui détermine la pertinence du résultat.

On observe sur la figure 5 que pour une plus grande valeur du paramètre ε, une trop petite valeur de α ne permet pas de distinguer la structure de l'anneau. On imagine que de manière générale, pour $\alpha = \pi/3$ les lignes correspondantes sont assez pertinentes pour décrire la structure du nuage de points.

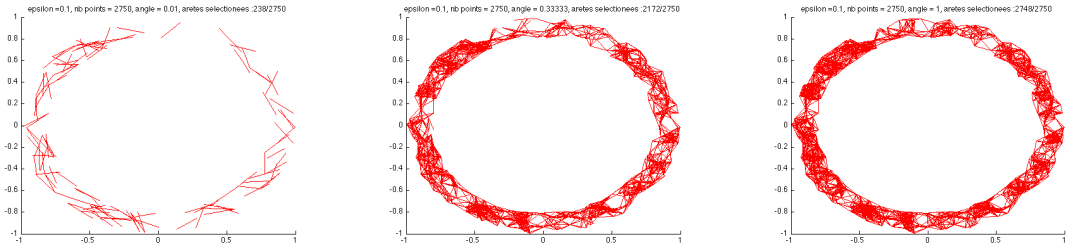


FIGURE 5 – Ensemble des arêtes sélectionnées avec $\varepsilon = 0.1$ et respectivement $\alpha = \pi/100$, $\alpha = \pi/3$ et $\alpha = \pi$.

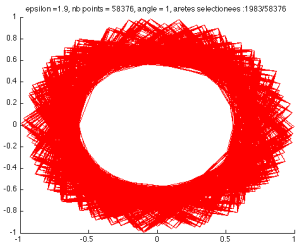


FIGURE 6 – Ensemble des arêtes sélectionnées avec $\varepsilon = 1.9$ et $\alpha = \pi$.

Enfin, on observe sur la figure 6 que pour une plus grande valeur de ε on distingue la structure de l'anneau sans pour autant en saisir la dimension réelle. Les cas $\alpha = \pi/100$ et $\alpha = \pi/3$ ne sont pas réprésentés car aucune suite d'arête n'est sélectionnée dans ce cas par le critère d'angle. A nouveau c'est le rapport entre ε et une taille caractéristique de la structure du nuage qui doit entrer en jeu.

En conclusion il semble nécessaire de penser à ajouter une phase d'optimisation des paramètres définissant les lignes du squelette pour saisir de façon pertinente la structure du nuage.

3 Une méthode par dichotomie

De manière assez différente des idées évoquées dans la partie précédente, on peut imaginer regrouper les différents points du nuage $E \subset \mathbb{R}^d$ en testant leur proximité par une subdivision adaptative de l'espace environnant. C'est une idée assez répandue en simulation numérique où, à l'issue d'un calcul, on est amené à subdiviser les cellules de la grille de calcul où l'on prédit (au sens où l'on dispose d'estimateurs d'erreur a priori, ou a posteriori) que des phénomènes nécessitant une précision accrue se déroulent localement (voir par exemple, mais non limitativement [1]).

A. Identification des zones de forte densité du nuage de points

Informellement, la méthode envisagée procède comme suit :

1. Inclure tous les points du nuage E dans un 'gros' hypercube (on dira indifféremment 'cube' dans la suite) $\mathcal{C}$ de $\mathbb{R}^d$. En pratique, $\mathcal{C}$ est construit à partir des coordonnées extrémales des

points du nuage ; par exemple :

$$\mathcal{C} = \left\{ x = (x_1, ..., x_d) \in \mathbb{R}^d \ / \ \forall i = 1, ..., d, \inf_{p=(p_1,...,p_d)\in E} p_i \leq x_i \leq \sup_{p=(p_1,...,p_d)\in E} p_i \right\}.$$

2. Couper ce cube $\mathcal{C}$ en deux dans chaque direction de $\mathbb{R}^d$. Le cube initial se trouve subdivisé en 2^d sous-cubes. Si le nuage de points était uniformément réparti dans l'espace, chacun de ces sous-cubes $\mathcal{C}_j, j = 1, .., 2^d$, aurait la même *densité* d_j de points :

$$d_j := \frac{\text{Nombre de points de } E \text{ appartenant à } \mathcal{C}_j}{\text{Nombre total de points de } E} \approx \frac{1}{2^d}.$$

3. On souhaite regrouper les points par paquets. Pour ce faire, on peut ne garder que les sous-cubes $\mathcal{C}_j$ qui ont une densité de points *nettement* supérieure à la densité qu'auraient tous ces sous-cubes si le nuage de points était uniformément réparti, ou au contraire, abandonner ceux qui ont une densité de points nettement inférieure à cette densité moyenne. On peut par exemple se donner un paramètre $\alpha >> 1$ et décider de ne garder parmi les sous-cubes $\mathcal{C}_j$ que ceux dont la densité est :

$$d_j > \frac{\alpha}{2^d}.$$

   À noter que les points écartés par cette méthode, qui apparaissent ainsi isolés des zones de forte concentration du nuage pourraient correspondre à l'idée que l'on se fait d' événements 'rares'.

4. Repartir du point (2) pour chaque sous-cube $\mathcal{C}_j$ conservé à l'étape précédente.

Ainsi, en fonction du nombre d'itérations de ce processus, on parvient à une structure de plus en plus fine, qui englobe les structures remarquables du nuage E. La figure 7 donne une illustration du déroulement de cet algorithme sur un exemple simple en dimension 2.

Une fois obtenu un tel recouvrement grossier des composantes dominantes du nuage de points E, se pose la question de reconnaître la nature des entités qui apparaissent : peut-on les voir comme des morceaux de droites, ou bien est-il plus judicieux de les voir comme des pans de surface,..., d'entités de codimension 1 ?

Pour répondre à cette question, une information clé semble être la connaissance des relations de voisinage entre cellules d'une subdivision d'un cube par dichotomie (voir figure 8).

B. Reconnaissance des relations de voisinage dans une grille à d dimensions

Soit un hypercube de dimension d (dans la suite, on supposera sans perte de généralité qu'il s'agit du cube unité), que l'on subdivise successivement n fois par dichotomie. On souhaite attribuer un numéro à chaque sous-cellule d'une manière logique, qui permette ensuite, si l'on considère deux cellules ou plus, de reconnaître par un simple test sur ces numéros si elles sont voisines (par une hyperface).

On propose pour ce faire de procéder comme suit (voir la figure 9 pour un exemple) : un sous-cube d'une subdivision à l'ordre n sera repéré par un d-uplet $(c_1, ..., c_d)$ d'entiers, chacun ayant une représentation binaire sur n bits. Plus précisément,

1. Au moment de réaliser la première subdivision, à chacun des 2^d sous-hypercubes obtenus est associé un d-uplet de bits - i.e. l'entier 0 ou 1 - $(c_1, ..., c_d)$. Chaque bit c_i vaut 0 si le sous-cube en question est compris dans le demi-cube ' à gauche' $\left\{x = (x_1, ..., x_d) \in \mathbb{R}^d \ / \ 0 \leq x_i \leq \frac{1}{2}\right\}$, 1 s'il est compris dans le demi-cube ' à droite' $\left\{x = (x_1, ..., x_d) \in \mathbb{R}^d \ / \ \frac{1}{2} \leq x_i \leq 1\right\}$.

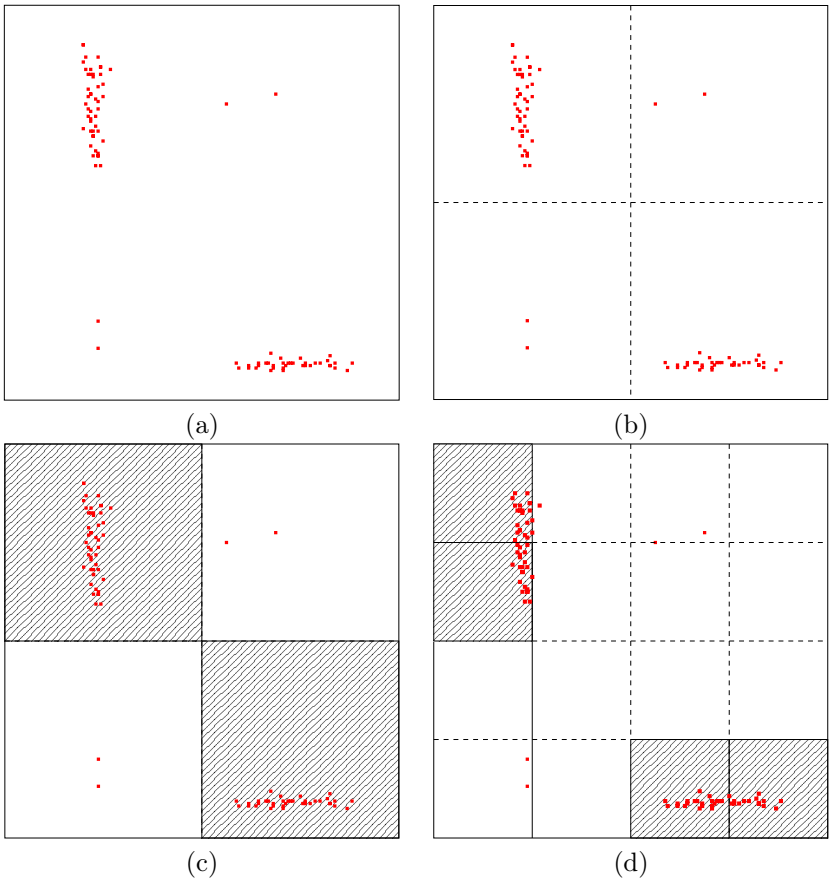


FIGURE 7 – Mise en œuvre de la méthode dans le cas d'un nuage de points de $\mathbb{R}^2$: (a) Le nuage de points E, placé dans un cube englobant, (b) subdivision du cube en 2^2 sous-cubes, (c) sélection des sous-cubes de forte densité et (d) itération du processus.

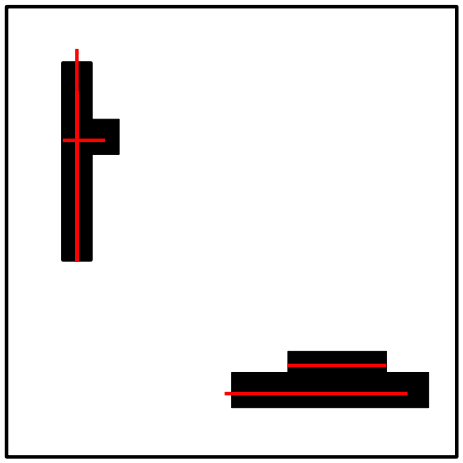


FIGURE 8 – Appariemment des 'amas' de points proches en structures d'ordre supérieur

2. Chacun de ces sous-cubes au niveau 1 est ensuite lui-même divisé en 2^d sous-cubes de niveau 2. Chacun de ces sous-sous-cubes est alors encodé par un d-uplet $(c_1, ..., c_d)$, où chaque c_i

est un entier qui tient sur deux binaires, $c_i = a_{i,1}a_{i,2}$ (il s'agit de l'écriture en représentation binaire : l'entier associé est $2^1\ a_{i,1} + 2^0\ a_{i,2}$), $a_{i,1}$ étant hérité de la subdivision de niveau 1, et $a_{i,2}$ rendant compte de la position de ce sous-sous-cube dans le sous-cube afférent, selon la règle de positionnement précédente.

3. ...

4. Au dernier niveau n de subdivision, chaque sous-cube de niveau n est repéré par un d-uplet $(c_1, ..., c_d)$, où chaque c_i est un entier qui tient sur n binaires, $c_i = a_{i,1}a_{i,2}...a_{i,n}$, $a_{i,j}$ valant 0 ou 1 selon la position de ce cube dans le sous-cube de niveau j.

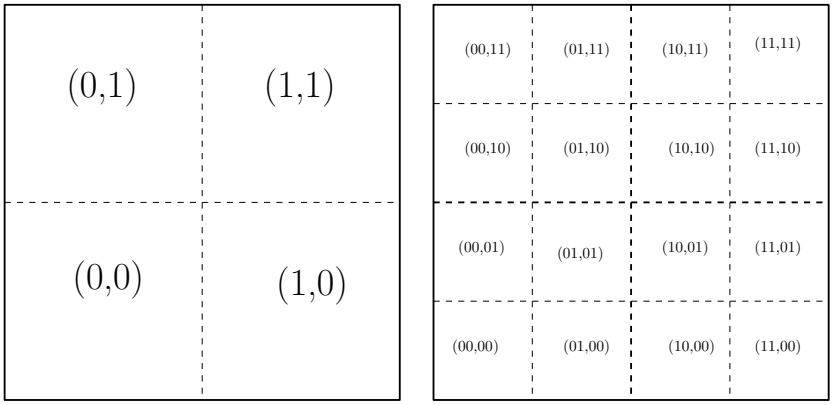


FIGURE 9 – Représentation binaire des deux premiers niveaux de subdivision du carré unité de $\mathbb{R}^2$.

Cette représentation binaire d'un sous-cube de niveau n étant obtenue, il semble alors possible de décider si deux sous-cubes de niveau n sont ou non adjacents par une hyperface :

Proposition 3.1 *Deux sous-cubes d'ordre n, représentés par respectivement* $(b_1, ..., b_d)$ *et* $(c_1, ..., c_d)$ *sont adjacents par une face (i.e. une entité de dimension* $d-1$*) si et seulement si les entiers associés aux représentations binaires* $b_1, ..., b_d$ *et* $c_1, ..., c_d$ *sont tels que*
- *Il existe un unique* $i = 1, ..., d$ *tel que* $b_i \neq c_i$*, et*
- *pour cet indice i,* $|b_i - c_i| = 1$.

Ce résultat résiste à tous les exemples que nous avons pu construire, bien que nous n'ayons pas eu le temps d'en dériver une preuve formelle.

Remarques et perspectives :

- Cette méthode semble réaliste pour traiter de nuages de points dans des espaces de dimension d 'raisonnable' : en effet, chaque étape de subdivision d'un cube fait immédiatement intervenir 2^d sous-cubes, ce qui peut très vite s'avérer prohibitif si d est très important. Dans ce dernier cas, on pourrait penser à adapter cette méthode pour ne subdiviser que dans certaines dimensions, afin de limiter le nombre d'entités manipulées ; il faut alors s'attendre à ce que la combinatoire - notamment pour ce qui est de reconnaître les entités (qui seront alors des pavés en dimension d en général) voisines l'une de l'autre - devienne plus complexe.

- Cette méthode trie les points selon qu'ils sont proches au sens de la distance euclidienne de $\mathbb{R}^d$ ou non. Ce critère peut être satisfaisant pour certaines applications, mais s'avérer inadapté pour d'autres. Il faut alors introduire une autre manière de mesurer des distances entre les

points du nuage, par exemple via la donnée d'une métrique sur $\mathbb{R}^d$, qui est intrinsèquement dépendante de la manière dont on a projeté (ou chiffré) les données considérées sur l'espace $\mathbb{R}^d$. Malheureusement, il semble difficile d'utiliser une autre métrique que la métrique euclidienne dans le contexte de cette méthode.

– On pourrait utiliser cette idée de dichotomie pour tester quelles sont les directions $x_1, ..., x_d$ qui sont véritablement significatives pour le nuage de points considéré : il est ainsi tout à fait possible de réduire la dimension du problème étudié, avant d'employer éventuellement une autre technique pour mener à bien l'étude du nuage considéré.

Remerciements

Nous tenons à exprimer notre gratitude aux organisateurs de cette seconde Semaine d'Étude Maths-Entreprises, pour nous avoir donné l'occasion de découvrir et de travailler sur des thématiques nouvelles, dans des conditions d'accueil absolument irréprochables, et dans une ambiance tout à fait conviviale. À tout point de vue, cet aperçu a été enrichissant, en grande partie grâce à l'encadrement et aux conseils prodigués par M. Philippe Saadé. Ces remerciements ne sauraient être complets s'ils ne s'adressaient pas aux personnes qui nous ont consacré du temps pour nous aiguillller et nous conseiller dans notre manière d'appréhender le sujet : un grand merci donc à MM. Simon Masnou, Bertrand Maury, Régis Monneau et Didier Auroux !

V_3

Références

[1] C. Min and F. Gibou, *A second order accurate level set method on non-graded adaptive cartesian grids*, J. Comput. Phys 225 (2007), pp. 300–321.