Faculté des Sciences et Technologies

Rendu

Travail Encadré de Recherche

2 mai 2023

1 Résumé et motivations

1.1 Résumé

La TDA : topological data analysis (ou traduit littéralement : analyse de données topologique) est un domaine très récent dans la data science. Les premières recherches datent des années 2000 et consiste à déduire des informations à partir de la forme de nos données/de notre nuage de points. Pour cela, nous ferons appel à de nombreuses notions topologiques, informatiques mais aussi statistiques.

Il existe deux grandes approches pour illustrer la TDA :
L'algorithme de mappage : (ou mapper algorithm) où l'on se base sur l'idée que le nuage de points peut se décomposer en un graphe avec une structure bien particulière. Toutes les propriétés/notions que nous connaissons sur les graphes nous permettront une grande source d'interprétation.
L'homologie persistante : qui est plus difficile théoriquement mais bien plus visuelle et pratique. Lors de cette approche, nous parlerons notamment de simplexes. En bref, cette approche consiste à faire grossir un disque de rayon r autour des points du nuage jusqu'à ce que le nuage de points forme un graphe complet.
Pendant cette procédure, on enregistre les naissances et les morts des groupes d'homologies intervenant, appelé nombre de Betti.
Nous serons donc en mesure de savoir si deux ensembles de données sont équivalents quant à leurs formes qu'ils occupent dans l'espace.

1.2 Motivations

Peu de sujet de TER proposé m'inspirait. Je cherchais à retrouver des notions que j'avais apprécié lors de la L2 en fonctions de plusieurs variables. Nous avions introduit de la topologie et cette branche m'avait plu. Cependant, étant donné des connaissances très limités dans ce domaine : en faire de ce sujet un TER était très délicat de par l'étendue et la complexité que représente la topologie.
Grâce à Monsieur Sabbah, j'ai pu trouver un solide mélange entre mes envies et le lien avec la licence MIASHS (Sciences des données - Fouilles de graphes et réseaux sociaux), d'où ce sujet de TER.

Table des matières

2 Contexte algébrique

2.1 Cadre topologique

Nous considérerons toujours des nuages de points dans $\mathbb{R}^2$. L'étude peut s'appuyer sur $\mathbb{R}^p$, il sera néanmoins plus lisible de considérer le plan pour nos graphes et différentes représentations graphiques.

Definition 2.1 (Simplexe) *On définit par simplexe la généralisation de la notion de triangle dans une dimension quelconque. On définira donc par n-simplexe le simplexe à n dimensions.*

Exemple 1 *Un 0-simplexe est un point, un 1-simplexe est un segment, un 2-simplexe un triangle, un 3-simplexe un tétraèdre etc.*

Même si nous nous restreignons à un nuage de points dans le plan, il sera possible de considérer des simplexes à plus de 2 dimensions (d'où la distinction de p et n dans certains cas, qui ne nous concerneront pas ici).
Toute face d'un n-simplexe est un (n-1)-simplexe. [1]

Exemple 2 *Une face d'un tétraèdre est un triangle.*

Une union finie de simplexes est appelé complexe simplicial [1] [2], on aura l'occasion d'en voir la représentation dans le cadre de l'homologie persistante.

Definition 2.2 (Homéomorphisme) *Deux objets (ou plus précisement variétés dans le cadre topologique) sont dits homéomorphes si l'on peut passer de l'un à l'autre sans déchirures ni recollements. Un exemple très connu est celui de la boule et du cube : ils sont tous les deux homéomorphes.*
Plus formellement, si E et F sont des ensembles munis d'une topologie, on dira que $f : E \to F$ est un homéomorphisme si et seulement si f est continue, bijective et sa réciproque continue.

Cette notion est directement reliée à notre problématique de départ : nous souhaitons savoir si un nuage de points peut être caractérisé par l'espace qu'il occupe. Si deux nuages de points occupent deux espaces homéomorphes, peut-on dire que ces nuages transcrivent les mêmes informations ?

Pour terminer cette section, nous considérerons :

Definition 2.3 (nombres de Betti, informelle) *Les nombres de Betti sont les entiers naturels qui forment une suite où le i-ème terme de la suite représente le nombre de surfaces i-dimensionnelles indépendantes. On notera H_i le i-ème terme de la suite.*

Dans notre étude, les seuls termes qui nous intéresseront seront H_0 : le nombre de composantes connexes du nuage de points (en combien de morceau puis-je tenir ce nuage ?) et H_1 le nombre de 'trous' qui apparaissent dans un nuage de points.
La figure 1 ci-dessous présente 1 composante connexe et 1 cavité (celle formée par le tétraèdre) ainsi : $H_0 = 1$, $H_1 = 0$ et $H_2 = 1$.
La question : 'comment passe-t-on d'un nuage de points à un complexe simplical ?' sera répondue dans lorsque nous discuterons de l'homologie persistante.

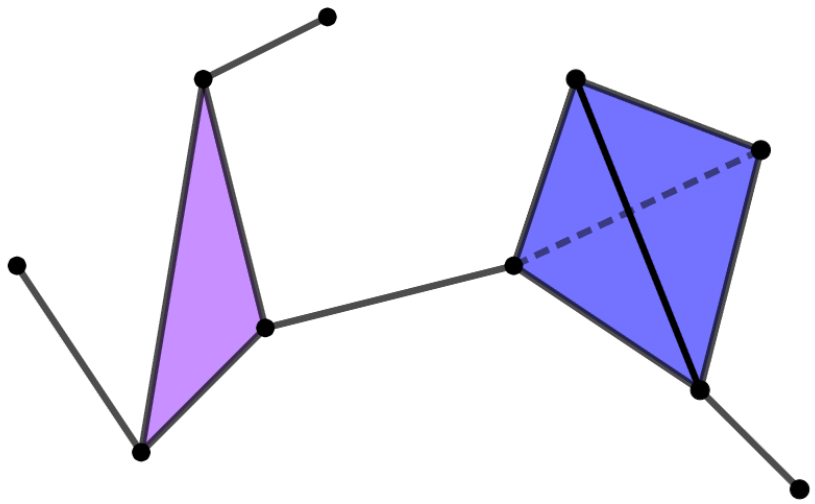


FIGURE 1 – Exemple d'un complexe simplicial

2.2 Un peu de théorie des graphes

Peu importe la vision que l'on portera sur le sujet de la TDA, la notion de graphe sera toujours de près ou de loin présente, c'est pourquoi il est nécessaire à mon sens de reposer le cadre.

Definition 2.4 (Graphe) *Un graphe se définit par un ensemble de points (sommets ou noeuds) auxquels peuvent se relier des arêtes. Plus formellement, on pose un graphe G : $G(V, E)$ avec V l'ensemble des sommets le composant, et E l'ensemble des arêtes formées par le lien entre deux sommets v_i et v_j de V (avec $i \neq j$ ici et $i,j \in \{1,...,n\}$ où $n = Card(V)$).*

Nos graphes seront toujours non-dirigé et auront toujours un nombre de sommets fini. Voici quelques propositions qu'il est notable de définir pour la suite :

Proposition 1 (Graphe connexe) *Un graphe G : $G(V, E)$ est connexe si :*
$\forall v_i, v_j \in V$, il existe au moins un chemin (i.e un parcours d'arêtes) entre v_i et v_j.
Autrement dit, il est possible de tenir le graphe en un seul morceau. (lien avec les composantes connexes et les nombres de Betti explicités plus haut (2.3))

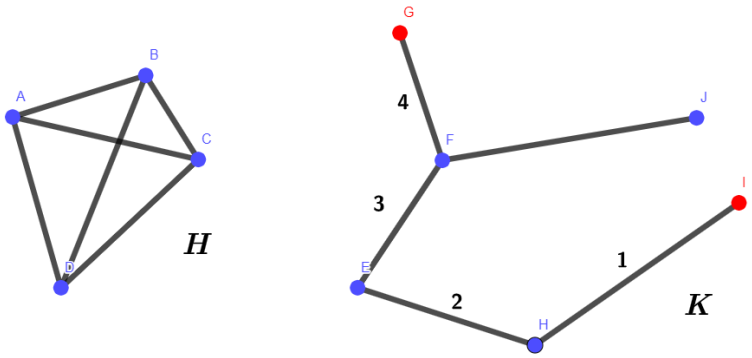
Proposition 2 (Graphe complet) *Un graphe G : $G(V, E)$ est complet si :*
$\forall v_i, v_j \in V$, il existe au moins une arête entre v_i et v_j.
Autrement dit, tous les sommets sont tous reliés entre eux.

Par conséquent, dans un graphe complet, chaque sommet est relié par $(n-1)$ arêtes.

Proposition 3 (Excentricité d'un sommet) *Soit un graphe G : $G(V, E)$,*
$\forall v_i \in V$ au moins relié par une arête, il est possible de définir son excentricité en calculant la distance de ce v_i à tous autres sommets v_j et d'en prendre la plus élevée.

Dans notre étude de l'algorithme de mappage, nous utiliserons une autre manière pour déterminer l'excentricité d'un sommet, mais l'idée restera la même.

La figure 2 ci-dessous reprend toutes les notions vues ici :



$$G = (\{A; B; D; E; F; G; H; I; J\}, \{(A, B); (A, D); ...; (F, J)\})$$

FIGURE 2 – Graphe G

Le sous graphe H est un graphe complet : tout sommet de H est relié à tout autre sommet de H. Le graphe G n'est pas connexe, il existe deux composantes connexes : le sous graphe H et le sous graphe K.
L'excentricité du sommet I est 4, en effet, le sommet qui est le plus éloigné de I est le sommet G et il faut parcourir 4 arêtes pour l'atteindre.

3 Algorithme de mappage

Maintenant que nous avons défini les principaux supports dont nous aurons besoin, passons à la présentation de la première approche, l'algorithme de mappage.

3.1 Introduction

Cet algorithme est applicable partout tant qu'il existe des données exploitables. Il permet de distinguer des communautés (clustering) de plus ou moins fortes densités dans un jeu de données tout en le retranscrivant dans un espace plus petit grâce à une fonction de filtre choisie en amont par l'utilisateur. L'algorithme de mappage se veut très flexible dû au nombre d'hyperparamètres qu'il le compose, chaque recherche peut avoir sa propre méthodologie même s'il en existe des usuelles.
Il est utilisé dans beaucoup de domaines notamment médicaux puisqu'il s'agit d'un outil performant dans la reconnaissance de structures complexes tels que les cellules. Pour prendre un exemple, l'algorithme est capable de détecter les sous-types de cancer du sein dont certains patient.e.s sont atteints et de distinguer les rechutes. [3]

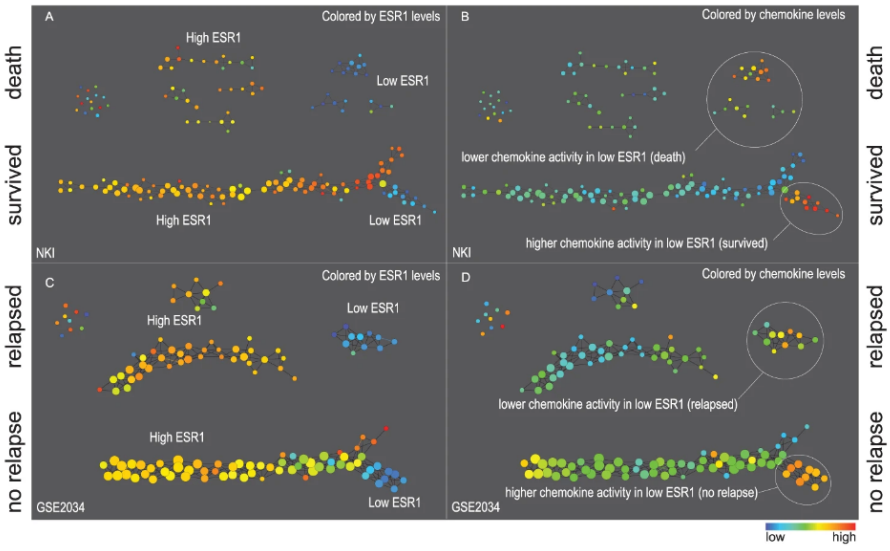


FIGURE 3 – Algorithme de mappage appliqué au cancer du sein

3.2 Méthode

La méthode en théorie est relativement simple [2] : considérons un nuage de points dans $\mathbb{R}^2$. Nous définissons une fonction de filtre f qui aura pour but de projeter les points du nuage de manière unidimensionnelle (ou en tout cas de réduire la dimension de départ). Cette fonction doit évidemment être choisie avec soin et requiert parfois d'en savoir beaucoup sur nos données et sur les résultats que nous souhaitons mettre en avant, nous en reparlerons.
Une fois projetés sur une droite, les points se verront être "agrégés" par des clusters dont la taille sera choisie au préalable. Chaque cluster correspond à une classe et lorsque deux clusters forment une intersection, les points à l'intérieur de cette dernière formeront comme un 'lien' entre ces deux classes. Tous les points doivent être pris en compte.
Une fois cette étape intermédiaire effectuée, nous rebasculons dans notre dimension de départ ($\mathbb{R}^2$ ici) où chaque classe peut être vue comme des sommets d'un graphe qui seraient reliés par des arêtes formées des points issus des intersections de clusters. Pour être plus précis, le résultat graphique est un complexe simplicial, que l'on simplifiera en un graphe.

3.3 Un premier exemple : l'analyse en composantes principales

Supposons donc un nuage de points dans $\mathbb{R}^2$ dont la seule chose que l'on connaisse soit les coordonnées des points et leurs classes initiales (ces dernières n'ont pas besoin d'être connues, elle ne serviront ici que pour la visualisation). Notre but est d'effectuer l'algorithme de mappage dessus et d'en tirer de l'information.
Puisque nous connaissons ses coordonnées, regardons à quoi il ressemble :

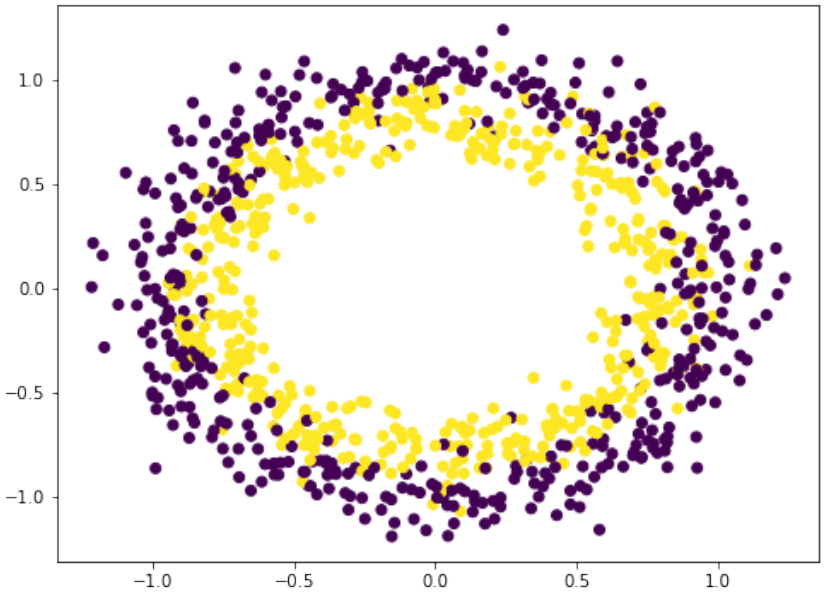


FIGURE 4 – Nuage de points

Pour que la méthode et les résultats soient plus flagrants, j'ai décidé de prendre un nuage de points formant un cercle, il est bien entendu possible de prendre n'importe quel autre nuage de points raisonnablement grand pour pouvoir effectuer l'algorithme.

Pour cette première application, notre choisissons une fonction f qui reflétera le procédé de l'analyse en composantes principales. Sans trop rentrer dans les détails, survolons ce concept :
L'analyse en composantes principales est un des nombreux outils permettant de réduire la dimension de données toute en retranscrivant de manière la plus fidèle possible, les informations portées par ces données.
Pour ce faire, dans notre cas en deux dimensions, l'ACP va représenter nos données sur un seul axe (la droite verticale représentée en figure 5) en retenant le plus possible la dispersion de nos données, c'est-à-dire la variance la plus grande possible. Autrement dit, on ne souhaite garder que les informations principales de notre point de départ : les données qui caractérisent - stéréotypent notre jeu de données.
Avant la procédure, nous normalisons nos données, et grâce à la librairie *sklearn* (pour un algorithme plus fonctionnel et optimal), voici ce à quoi nous aboutissons :

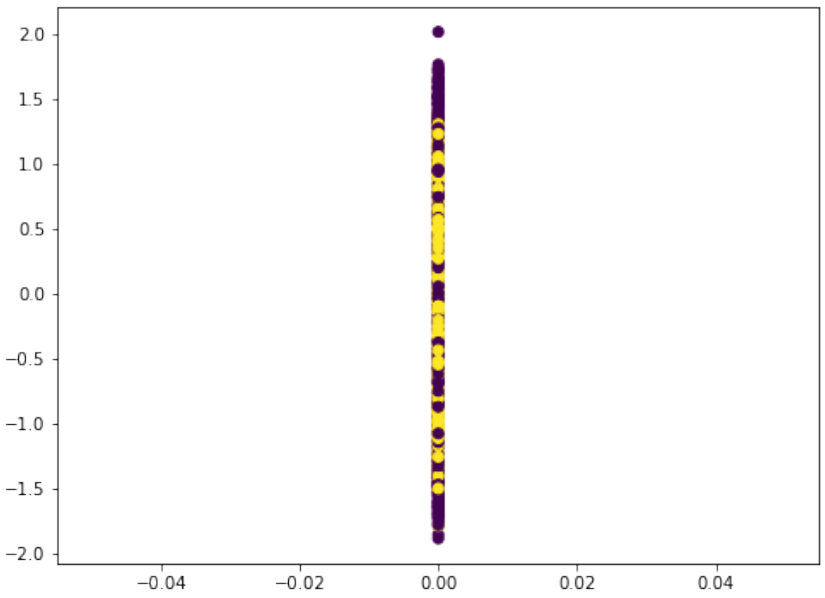


FIGURE 5 – Projection verticale par l'ACP

Nous observons une dispersion plutôt homogène partout sur la droite. On retrouve une légère concentration de points jaunes au centre de la projection, ce qui n'est pas étonnant au vu de la figure 4. Nos données s'étalent sur l'intervalle $[-2, 2]$, données centrées par *sklearn*. Une illustration ci-dessous permet de mieux visualiser ce qu'il se passe sur un nuage de points elliptique.

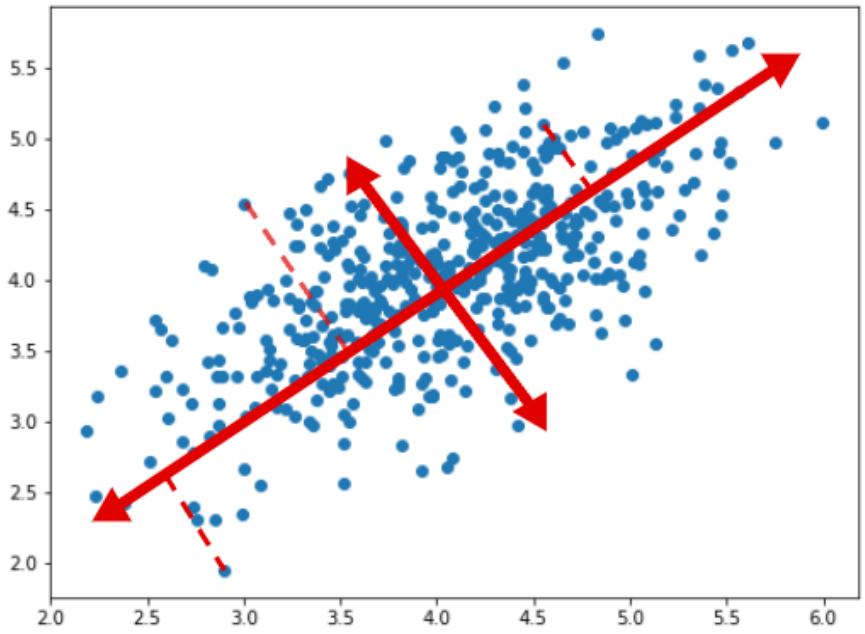


FIGURE 6 – Illustration fictive de l'ACP

Ce que l'on veut à présent, c'est pouvoir clusteriser ces points pour en déterminer un graphe. [2]
Supposons que nous souhaitons distinguer 3 classes (en ne prenant plus en compte les classes jaunes et mauves du dessus) pour savoir comment semble se répartir les points selon l'ACP.
Pour ce faire, nous ordonnons et découpons nos données en deux, et ce, par exemple, de manière homogène. Nous pourrions le faire également de manière non homogène pour faire ressortir les communautés les plus extrêmes.
Suite à ce cela, nous fixons $\varepsilon > 0$, le débordement que l'on s'autorise sur ce découpage pour faire naître des intersections entre les deux classes. Dans notre cas, nous avons vu que les données s'étendent de -2 à 2. Par conséquent, ε sera raisonnablement petit pour créer des intersections pertinentes.

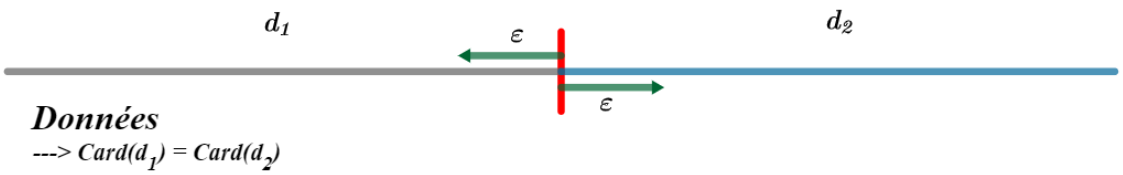


FIGURE 7 – Schéma de découpe de nos données

Pour des raisons de lisiblité, j'ai préféré faire le shéma de manière horizontale.

Le graphique ci-dessous résume toute cette discussion, nous avons fixé $\varepsilon = 0.3$. Nous avons donc la classe grise et bleue qui sont uniformément réparties, et la classe verte qui émane de l'intersection des clusters. C'est cette classe verte qui va permettre le 'pont' entre nos deux classes et former les arêtes de notre graphe final.

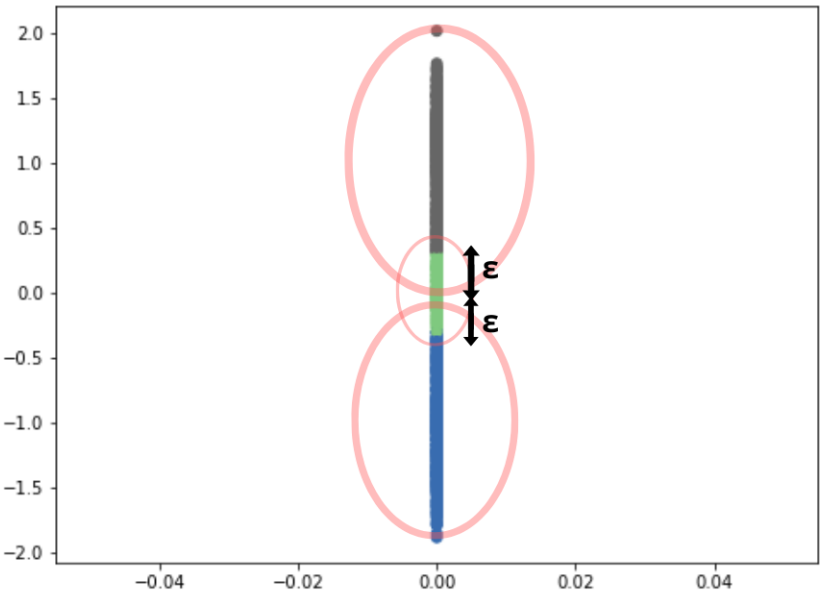


FIGURE 8 – Introduction des clusters et apparitions des classes pour l'ACP

Il est tout à fait possible de reproduire cette étape pour diviser notre jeu de données en n classes. Bien évidemment, cela requiert de plus en plus de données au fur et à mesure que n grandit et il sera probablement nécessaire d'aviser ε de manière plus précise afin d'avoir des découpages pertinents.

Maintenant que nous connaissons pour chaque point, sa classe, nous pouvons rebasculer en dimension 2 et observer ce que cela nous donne :

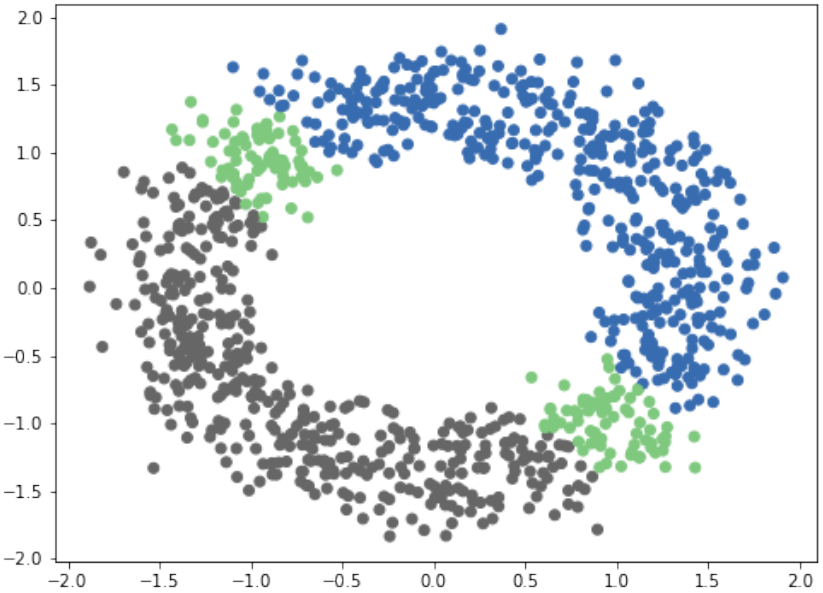


FIGURE 9 – Représentation de nos classes sur le nuage de points

Enfin, l'étape finale, consiste à transcrire ce graphique en un graphe récoltant les nouvelles informations que f nous a fourni [2] :

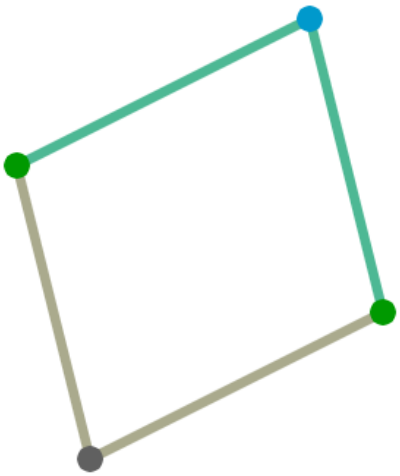


FIGURE 10 – Complexe simplicial issu de l'algorithme de mappage avec ACP

3.4 Un second exemple : l'excentricité

Pour cette seconde application, nous définissons $f : \mathbb{R}^2 \longrightarrow \mathbb{R}$ qui renvoie l'excentricité de chaque point de notre nuage. Les distances seront déterminées par la distance euclidienne que sépare deux points, l'excentricité d'un point sera donc la norme usuelle la plus élevée que l'on peut trouver parmi ses voisins les plus lointains.

Definition 3.1 (f) *Soit $\mathbb{X}$ notre jeu comportant n données, $\forall x = (x_1, x_2) \in \mathbb{X}$, on a :*

$$f(x) = \max_{y \in \mathbb{X} \setminus \{x\}} \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$$

Ainsi, pour tout point de notre nuage, est associé son excentricité, qui est un indicateur de l'éloignement de ce point par rapport aux autres. Nous sommes passé ici implicitement de données à deux dimensions à des données à une dimension, nous pouvons donc projeter ces nouvelles données sur une droite. Pour cela, nous ordonnons nos résultats en gardant en mémoire à quel point correspond chaque excentricité, nous obtenons (les données ont été normalisées pendant le processus) :

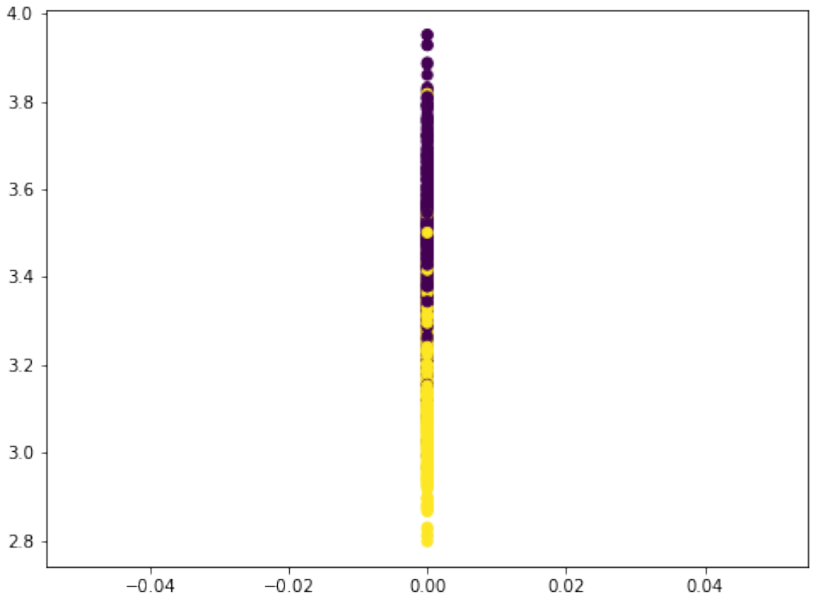


FIGURE 11 – Projection verticale par l'excentricité

On observe déjà ce à quoi on s'attendait, les points les plus excentrés (en mauve) ont une plus grande excentricité et remplissent la partie haute de la projection car ils recouvrent 'le bord' du cercle de points.

Pour définir nos clusters, une méthode satisfaisante est de découper nos données d'excentricité (qui sont ordonnées dans l'ordre croissant pour rappel) en 2 et dont on s'accorde un léger débordement ε. Si l'on répertorie dans E, un ensemble, toutes nos excentricités $(e_1, ..., e_n)$ renvoyées par f plus haut, et $e^{(i)}$ les données récupérés par les i débordements d'ε, nos intervalles ressembleront à ceci : (il s'agit du même raisonnement vu en (3.3))

$$E_1 = \left\{e_1, ..., e_{\lfloor n/2 \rfloor}\right\} \setminus E_3 \qquad E_2 = \left\{e_{\lfloor n/2 \rfloor}, ..., e_n\right\} \setminus E_3$$

$$E_3 = \left\{E_1 \cup \underbrace{e^{(1)}}_{+\varepsilon}\right\} \cup \left\{E_2 \cup \underbrace{e^{(2)}}_{-\varepsilon}\right\} \; avec \; E_1 \cup E_2 \cup E_3 = E$$

Ici, nous souhaitons ε relativement petit et proportionnel pour ne pas perdre trop d'information : n'obtenir qu'une seule classe car l'ensemble est entièrement compris dans l'intersection des clusters n'a aucun sens. Voici graphiquement avec nos données, et pour $\varepsilon = 0.1$ ce que cela donne :

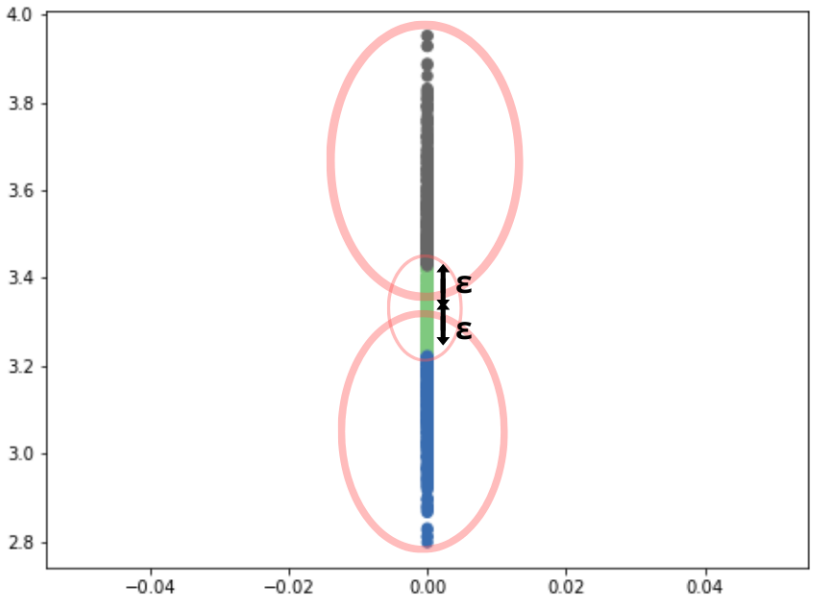


FIGURE 12 – Introduction des clusters et apparitions des classes pour l'excentricité

Les clusters gris (E_1) et bleus (E_2) ont le même nombre de données à un ou deux points près (erreur dû au ε) puique nous avons découpé nos données de manière uniforme. Nous obtenons nos deux classes, qui sont liées par les points représentés en vert (E_3). Si nous rebasculons en dimension 2 avec notre nuage de point initial comme le veut la méthode, nous obtenons la figure 13 ci-dessous.

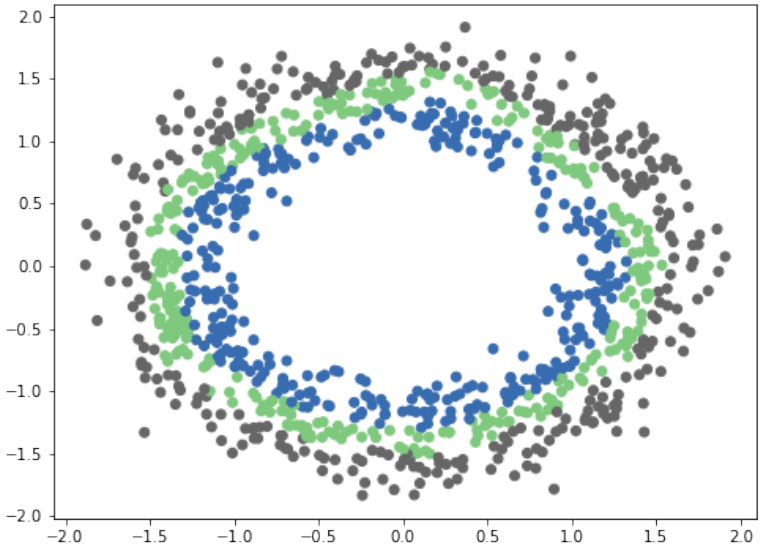


FIGURE 13 – Représentation de nos classes sur le nuage de points

Pour ce qui est de la représentation en un complexe simplial (i.e graphe ici pour simplifier), si l'on s'appuie sur l'interprétation de l'excentricité : on peut conclure que le graphe serait un graphe en forme de toile ou de flocon, ce qui serait radicalement différent du graphe issu de l'ACP. (interprétation personnelle).

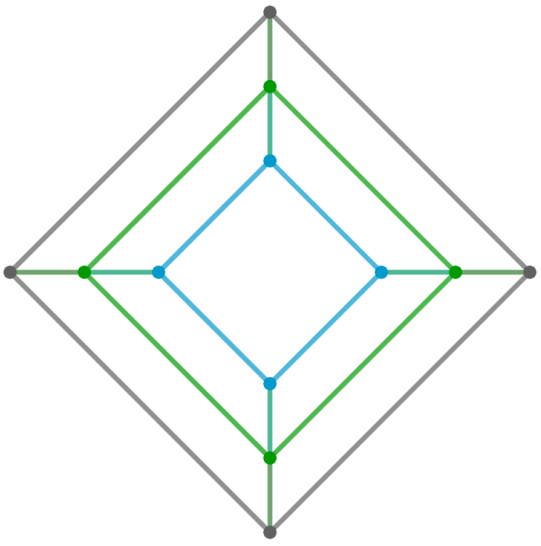


FIGURE 14 – Complexe simplicial issu de l'algorithme de mappage avec excentricité

Il est pertinent de remarquer que nous ne nous sommes pas penché sur l'établissement des clusters : nous avons réparti nos données de manières uniformes à chaque fois, puis avons débordé d'un certain ε. Cependant, juste avec la fonction de filtre et ce débordement, on aboutit à des graphes transcripteurs d'informations totalement différents.

3.5 Résultats et interprétations

Bien sûr, l'exemple ici se restreint à peu de points, cependant, si l'on prend des données plus vastes et moins 'triviale' qu'un nuage de points formant un cercle, le graphe aurait peut-être bien plus d'informations à transcrire.
Nous constatons une nette différence dans les deux approches déjà lors de la projection verticale. Il y a beaucoup de points concentrés autour de la médiane pour l'excentricité, de ce fait, lorsque l'intersection se fait au milieu de la projection (donc pour 3 classes), nous aurons tendance à avoir plus de points dans cette intersection, ce qui tend à avoir plus d'arêtes dans le graphe. L'ACP, elle, réduit bien plus fortement cette accumulation (point de vue et procédé complètement différent), ce qui lui permet d'être un peu plus invariante à ce genre de limites.
Nous remarquons également que les échelles et les bornes n'ont rien en commun, c'est pour cela que pour chaque fonction de filtre, il est important de définir (à tâtons si nécessaire) un ε porteur d'informations.

Afin d'arriver à des conclusions satisfaisantes, il existe beaucoup de paramètres dépendants à prendre en compte et à optimiser comme nous en avions parlé plus haut [2] (3.1).
La fonction de filtre doit être cohérente avec le jeu de données et y apporter un réel intérêt : elle est le support de ce que l'on cherche à montrer.
L'agrégation en classes de nos données est à débattre également. Il est possible de ne vouloir distinguer qu'un nombre très limité de groupes afin de condenser l'information que l'on peut en ressortir ou au contraire de la faire tendre proche du nombre de données pour évaluer les atomes du modèle.
Enfin, le choix d'ε est crucial : l'algorithme est très sensible à ses variations et peut changer du tout au tout. Pour reprendre l'exemple de l'excentricité, un ε deux fois plus grand avec 3 classes n'a plus de réel intérêt : tout semble mélangé et on ne distingue plus aussi nettement les groupes, c'est comme si nous avions perdu l'essentiel de l'information.

Pour conclure sur cette première partie, voici un exemple de ce à quoi ont abouti des chercheurs américains lorsqu'ils ont voulu implémenter l'algorithme sur des données issues de figures 3D. Leurs buts étaient de comparer la forme des nuages de points consistuants les modèles ci-dessous (avant et après mouvements) [4].

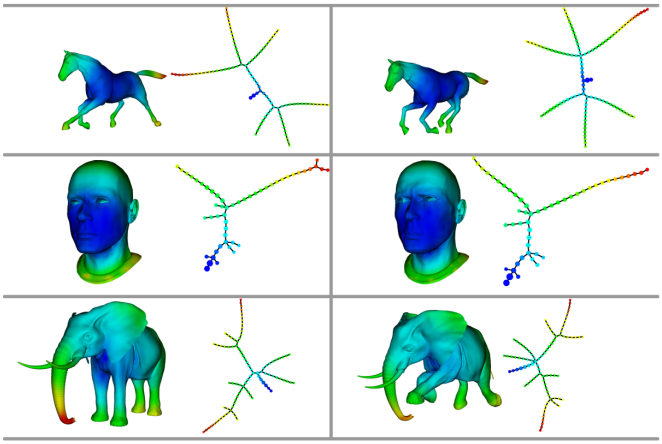


FIGURE 15 – Application de la fonction d'excentricité à des modèles 3D

4 Homologie persistante

4.1 Introduction

Nous considérerons que chaque point de notre jeu de données sera le centre d'un disque, de rayon r variable [5]. En fonction de ce r, notre nuage de points va évoluer en reliant ses points entre eux par des arêtes lorsque qu'une condition particulière est remplie. Au fur et à mesure des ces reliures, la structure du nuage va changer pour y faire apparaître des n-simplexes, formant des surfaces, des cavités etc. Nous verrons comment des arêtes peuvent se former.
Notre objectif est de pouvoir déterminer si deux jeux de données sont équivalents par la forme qu'ils occupent dans l'espace. Nous avons introduit plus haut le nombre de Betti (2.3), c'est par cette notion que cette approche se conclut. En effet, deux objets sont dits topologiquement équivalent si ils partagent la même suite de nombres de Betti. [2]
Cette approche consiste en fait à répondre à la question : "A quoi ressemble notre jeu de données ?" N'ayant besoin que de peu d'informations sur les données, elle est très simple d'utilisation et a par exemple aider des chercheurs de l'Université de Stanford à mieux visualiser les flux d'informations dans le système nerveux de certains primates. [6]

4.2 Méthode

La méthode consiste à passer en revue tous les états possibles que notre nuage de points est capable de passer et de les enregister. Pour ce faire, nous avons besoin d'outils. Notre analyse va se baser sur 3 graphiques complémentaires [7] : la représentation de notre nuage de points, le code-barre de persistance et le diagramme de persistance.
Le premier permettra de visualiser notre objet lors de la variation de r, le rayon des disques. Il nous sera utile pour visualiser et analyser l'évolution de notre nuage.
Le deuxième jouera un rôle d'enregistreur : au fur et à mesure de l'agrandissement de r, ce graphique nous retranscrit les différentes suites de Betti par lesquelles passent notre jeu de données. Chaque entier de ces suites est symbolisé par une demi-droite (verte ou rouge selon le groupe d'homologie), d'où cette visualisation en code-barre.
Enfin, le dernier graphique est un prolongement de ce dernier diagramme [8]. Il détermine pour chaque nombre de Betti sa durée de vie en un point (sa naissance en abscisse, sa mort en ordonnée). Ce graphique est scindé par la première bissectrice qui représente la délimitation entre espace réalisable et impossible. En effet, aucun trous ou autre cavités ne peuvent mourir avant même d'être né, donc être en dessous de cette droite.
Nous tracerons une arête entre un ou plusieurs points lorsque les disques formés par r s'intersectent, formant alors des n-simplexes (2.1). C'est ainsi que les complexes simpliciaux se formeront. [5]

4.3 Application sur l'alphabet

Pour une application simple et "concrète" de l'homologie persistante, j'ai décidé d'effectuer l'algorithme sur les 26 lettres que composent l'alphabet latin. L'idée est de savoir si, des lettres, par la forme qu'elles occupent dans l'espace, sont topologiquement équivalentes entre elles.
Pour ce faire, j'ai décidé de représenter les lettres dans leurs formes capitales sur une surface d'environ $2cm^2$. Chaque point composant ces lettres fait partie d'une extrêmité d'un segment ou d'une courbure.
Pour illustrer, voici comment le **R** est représenté :

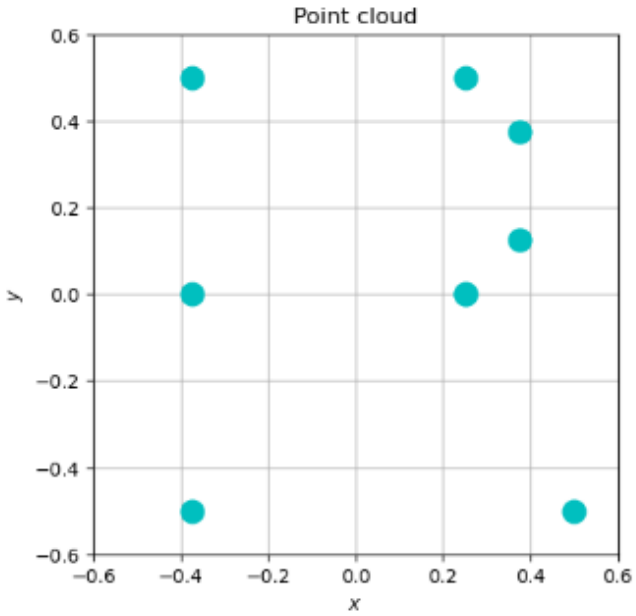


FIGURE 16 – Lettre R

Evidemment, chaque lettre a son nombre propre de points et il serait possible d'effectuer la même procédure sur une écriture en lettre attachées.
A partir de là, nous avons tous les éléments en main pour commencer le travail. Relatif à ce que nous avons dit plus haut, consacrons à chaque point une boule (ici un disque en dimension 2) qui grossit progressivement jusqu'à ce que les intersections de ces boules contiennent notre nuage de point. Pendant ce grossissement, nous constaterons les variations de composantes connexes et l'apparition de trous (plus formellement de courbes fermées).

Faisons une première démonstration avec la lettre **R**, nous poserons r, le rayon des disques. [7]
Commençons par $r = 0.08$:

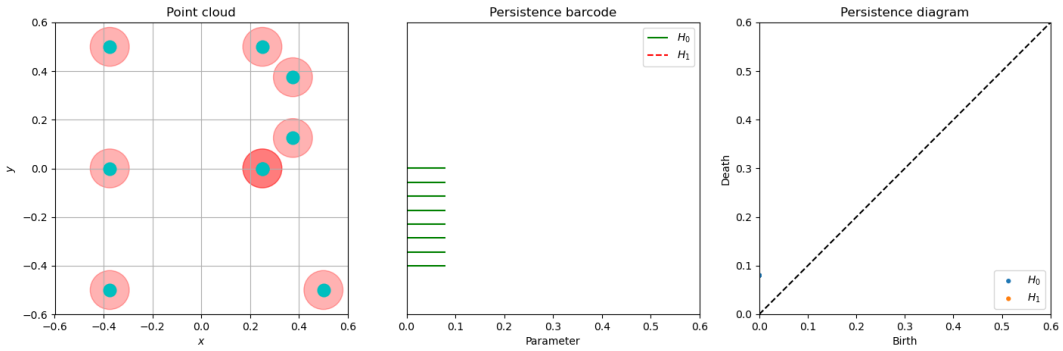


FIGURE 17 – Rayon égal à 0.08 cm

Pour un rayon au plus 2 fois inférieur à la plus petite distance entre deux points, il n'y a logiquement aucune intersection. De manière purement graphique, nous pouvons dire que les premières intersections se feront de manière simultanées entre les points $\{(0.25, 0); (0.375, 0.125)\}$ et $\{(0.25, 0.5); (0.375, 0.375)\}$. En effet, leurs distances s'élèvent à $\frac{1}{4\sqrt{2}}$ cm, il faut donc atteindre $r = \frac{1}{8\sqrt{2}}$ cm pour avoir les premières arêtes.
Pour le moment, nous restons donc avec notre nombre de points initial, soit 8 composantes connexes et aucun trou.

Pour $r = 0.16$:

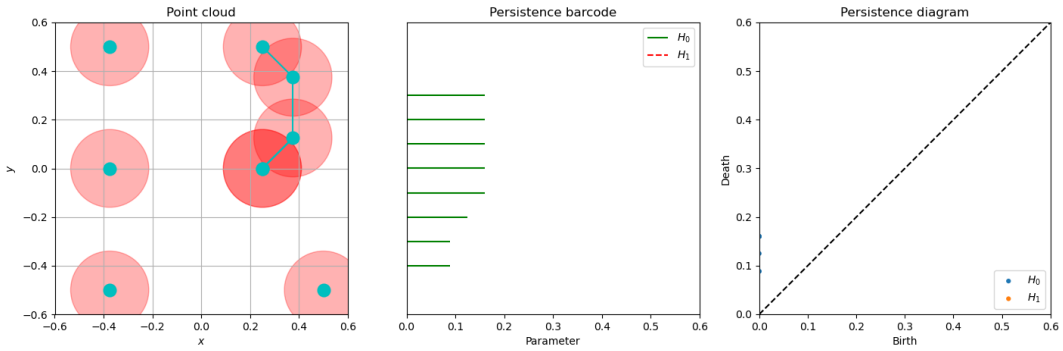


FIGURE 18 – Rayon égal à 0.16 cm

A ce stade, 4 points se sont liés, ce qui nous réduit notre nombre de composantes connexes de 3. Comme on peut le distinguer sur le graphique en code-barre, au delà de $r = 0.125$ ($3^{\text{ème}}$ ligne verte en partant du bas), nous n'avons plus que 5 composantes connexes, ce qui confirme bien nos constatations.

Pour $r = 0.35$:

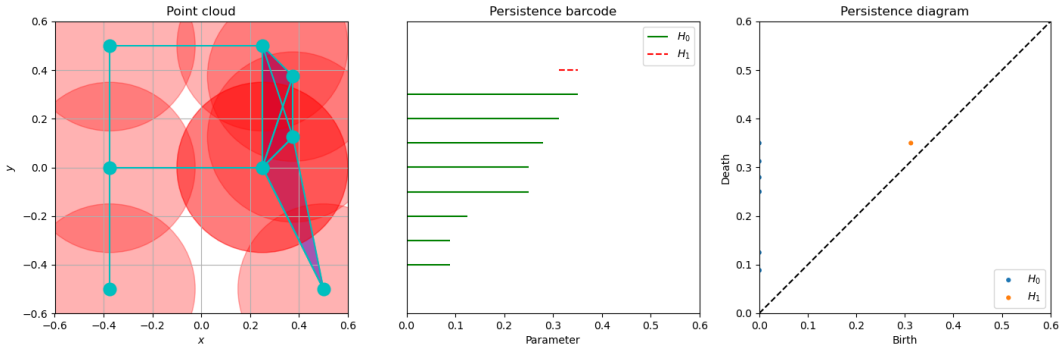


FIGURE 19 – Rayon égal à 0.35 cm

Pour ce rayon, plus aucun point n'est isolé, ce qui veut dire que nous ne détenons plus qu'une seule composante connexe, et ce jusqu'à la fin du grossissement du rayon.
A partir de $r = 0.3125$, un trou s'est formé (représenté par la boucle du **R**) : la durée de vie de ce trou est interprétée par la ligne discontinue rouge sur le diagramme à code barre et par le point orange sur le diagramme de persistance.
Une fois que le trou n'existera plus du fait de l'arrivée d'une nouvelle arête qui viendrait le transformer en une cavité, cette ligne rouge s'arrêtera, et nous serons en mesure de calculer sa durée de vie en fonction de r.
Remarque : La zone reliée située au dessus du triangle est apparue de manière instantanée, formant un tétraèdre, qui fait apparaître une cavité (H_2) et non un trou (H_1). Si on avait pris en compte les cavités, nous aurions encore eu une ligne différente dans le code barre et dans le diagramme de persistance.
Pour $r = 0.42$ et au delà :

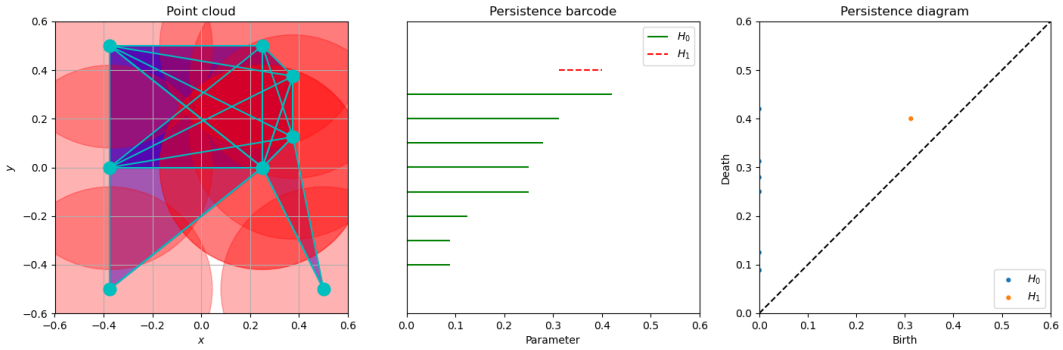
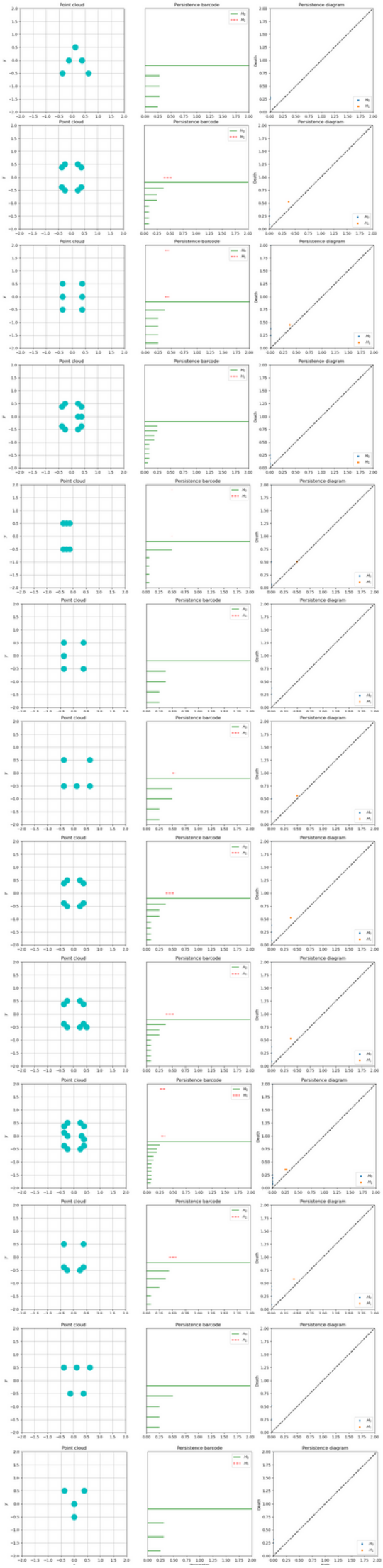


FIGURE 20 – Rayon égal à 0.42 cm

Aucun autre événement n'interviendra, le trou s'est dissipé en $r = \frac{\sqrt{41}}{16} \approx 0.4$ cm, lorsque les points inférieurs et supérieurs gauches ont été reliés à la boucle. Il n'y aura plus d'évolutions jusqu'à ce que les arêtes s'entrelassent pour former un graphe complet et arrêter la procédure.


Point cloud
Persistence barcode
Persistence diagram
Death

4.4 Résultats et interprétations

Ci-dessus, nous avons itéré 26 fois ce que nous avons effectué sur la lettre **R** [7]. Pour plus de visibilité, le choix de ne pas montrer les rayons des disques et les arêtes est volontaire.
Nous pouvons constater à première vue que la quasi-totalité des lettres contenant une boucle dans l'écriture manuscrite retranscrit un trou d'après l'homologie persistante. En effet, seul **A** en est l'exception.
Il existe des lettres qui forment un ou plusieurs H_1 de manière "innattendue" comme le **S** ou encore le **H**. Mais est-ce vraiment innattendu quand on sait comment ses trous apparaissent ? En réalité, pas vraiment. Ce qu'il se passe est relativement logique.
Pour prendre l'exemple de la lettre **H**, une fois que r est assez grand, les arêtes vont simultanément former deux rectangles dans la partie inférieure et supérieure de la lettre, d'où ces pointillés rouges qui ont exactement la même durée de vie. Il s'agit exactement des mêmes événements pour **S**.
D'autres au contraire et loin d'être une minorité, ne relèvent que des composantes connexes en terme d'informations. On ne peut cependant pas dire qu'ils occupent une même forme dans l'espace. Pour le prouver, il faudrait pousser l'analyse plus loin (analyse au delà de H_0 et de H_1 comme nous l'avons explicité dans l'onglet Méthode (4.2)). En effet, le **G** n'a par exemple rien à voir avec la plupart des autres lettres, l'expension de r créée deux tétraèdres (formant deux H_2) qui ne sont pas retranscrit par la simulation.

Avec ces hypothèses et cette méthode, qu'en conclut-on ?
On remarque que, l'aspect 'physique'/la ressemblance des lettres n'est pas toujours un critère pertinent pour déterminer si elles sont topologiquement équivalentes d'après l'algorithme. Pour s'en convaincre, il suffit de comparer le **C**, le **R**, le **M** et le **F**, tous se différencient dans leurs écritures, ils nous donnent pourtant les mêmes résultats.
Il est assez intuitif de penser que la rotation des lettres n'influence pas les résultats que nous avons, donc même si deux lettres ne se ressemblent pas de la façon dont on a l'habitude des les voir, leurs rotations, elles, peuvent être plus flagrantes. Par exemple, avec nos représentations, si l'on effetue une rotation à -90° de **D**, on retrouve (presque) le **U**. C'est donc pour cela que nous avons les mêmes résultats. Cet argument est encore plus fort sur le couple **N**, **Z** !
Relatif au paragraphe d'avant, on en déduit aussi qu'il ne suffit pas d'avoir une boucle dans une lettre pour que cette dernière ait un H_1 strictement positif, le **M** en est un exemple.

5 Conclusion

Nous avons exploré deux approches de la TDA : l'algorithme de mappage qui est un algorithme de clustering très modulable et personnalisable permettant la détection de communauté au sein d'un jeu de données. Et l'homologie persistante, qui, par le biais de notions topologiques, renvoie les caractéristiques de nos données dans l'espace.
Lors de ce TER, il aurait pu être intéressant de se pencher au delà de $\mathbb{R}^2$ pour notamment y interpréter les cavités H_2 dans le cadre de l'homologie persistante. Cela aurait nécessité d'étudier d'autres jeux de données, puisque considérer les lettres de l'alphabet en 3D n'ajouterait qu'une profondeur qui leurs seraient commune et donc dénuée de nouvelles informations. Il existe également de nombreuses autres manières de considérer les simplexes, ce qui rend l'algorithme plus ou moins coûteux.
La TDA est un domaine de recherche récent et très prometteur, applicable à toutes les échelles et peu gourmand en information, il fournit de nouvelles perspectives fiables dans la compréhension de structures complexes et ce même dans des espaces de grandes dimensions.

Références

[1] A. Mottet, "Homologie persistante et application en sécurité informatique," 2011.

[2] F. Chazal and B. Michel, "An introduction to topological data analysis : fundamental and practical aspects for data scientists," *Frontiers in artificial intelligence*, vol. 4, p. 667963, 2021.

[3] P. Y. Lum, G. Singh, A. Lehman, T. Ishkanov, M. Vejdemo-Johansson, M. Alagappan, J. Carlsson, and G. Carlsson, "Extracting insights from the shape of complex data using topology," *Scientific reports*, vol. 3, no. 1, pp. 1–8, 2013.

[4] G. Singh, F. Mémoli, G. E. Carlsson, *et al.*, "Topological methods for the analysis of high dimensional data sets and 3d object recognition.," *PBG@ Eurographics*, vol. 2, pp. 091–100, 2007.

[5] S. Talebi, "Topological data analysis introduction," 2022.

[6] G. Singh, F. Memoli, T. Ishkhanov, G. Sapiro, G. Carlsson, and D. L. Ringach, "Topological analysis of population activity in visual cortex," *Journal of vision*, vol. 8, no. 8, pp. 11–11, 2008.

[7] S. Ghafouri, "Persistent homology filtration + python code," 2022.

[8] H. Adams, "What is the difference between persistence barcodes and persistence diagrams ?," 2021.

[9] M. Wright, "Introduction to persistent homology," 2021.

[10] H. Adams, "Applied topology 8 : An introduction to persistent homology," 2021.

6 Annexes

Nous n'avons pas évoqué la possibilité de vouloir distinguer plus de 2 classes lors de l'algorithme de mappage, mais cela est tout à fait possible. Voici les résultats que l'on peut retrouver pour respectivement l'ACP et l'excentricité avec par exemple $k = 4$:

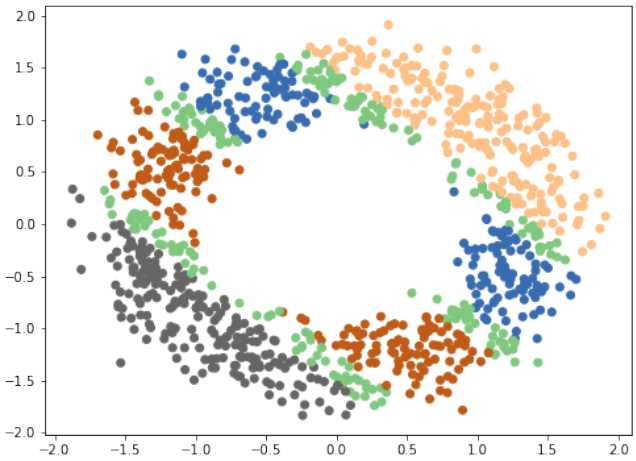


FIGURE 21 – ACP pour k = 4 classes

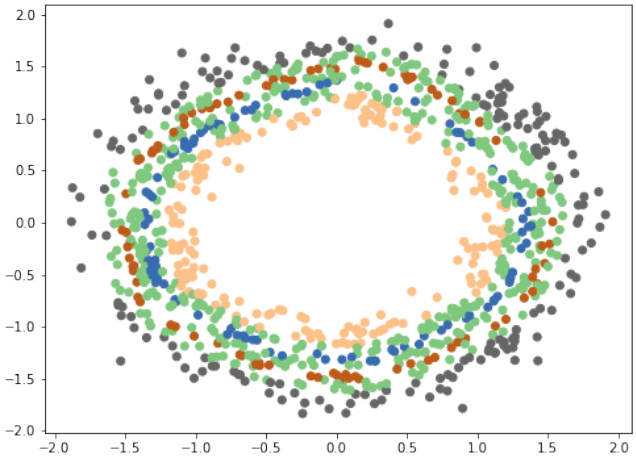


FIGURE 22 – Excentricité pour k = 4 classes

Nous observons que la distinction en communautés est plus claire du côté de l'ACP que de l'excentricité quand k grandit. En effet, ici nous sommes sur des données sphériques, "l'épaisseur" du cercle formé par les données est determinant ici. Nous concluons néanmoins que les deux méthodes aboutissent une fois de plus à de représentations radicalement différentes.

Et sur des jeux de données elliptique alors ? A quoi peut-on s'attendre ?
Il est notable de souligner que l'ACP se pratique généralement lorsque un jeu de données contenant des données corrélées (positivement ou négativement). Autrement dit, quand le nuage de point forme une ellipse plus ou moins étendue selon la matrice de variance-covariance. Voici ce à quoi nous aboutissons, dans le même ordre que précédemment et pour $k = 2$:

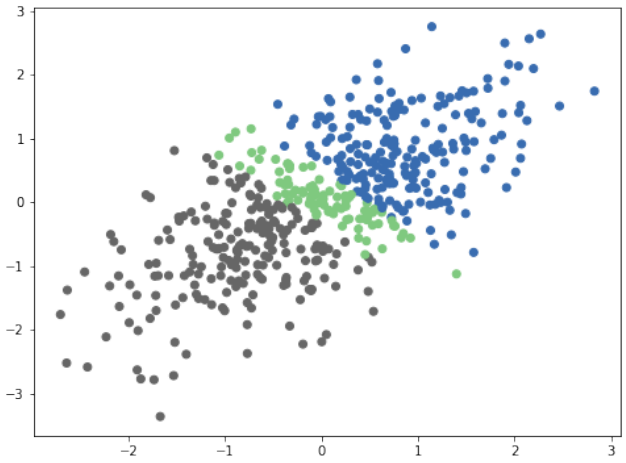


FIGURE 23 – Application de l'ACP

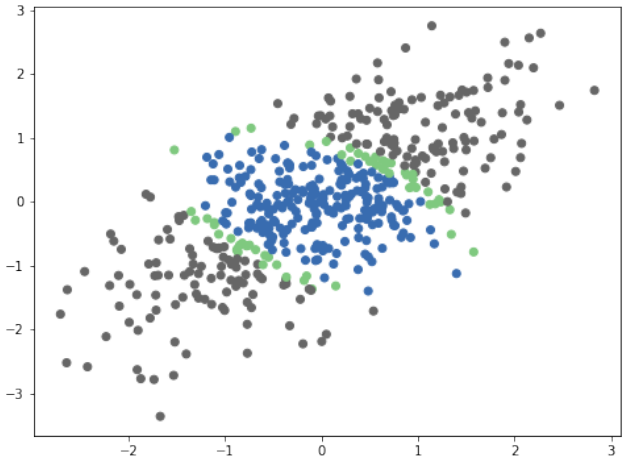


FIGURE 24 – Application de l'excentricité

En terme d'algorithme de mappage, il y a peu d'intérêt ici d'interpréter les graphes résulants puisqu'ils seraient sensiblement les mêmes. Il est cependant intéressant de noter que l'excentricité représente par une même communauté, des points qui ne sont pourtant pas spatialement proches, ce qu'on ne pouvait pas voir sur des données sphériques.

A quoi pourrait ressembler les diagrammes relatifs à l'homologie persistante sur un vrai jeu de données ?
C'est ce à quoi nous répondons ici, en prenant un jeu de 100 données :

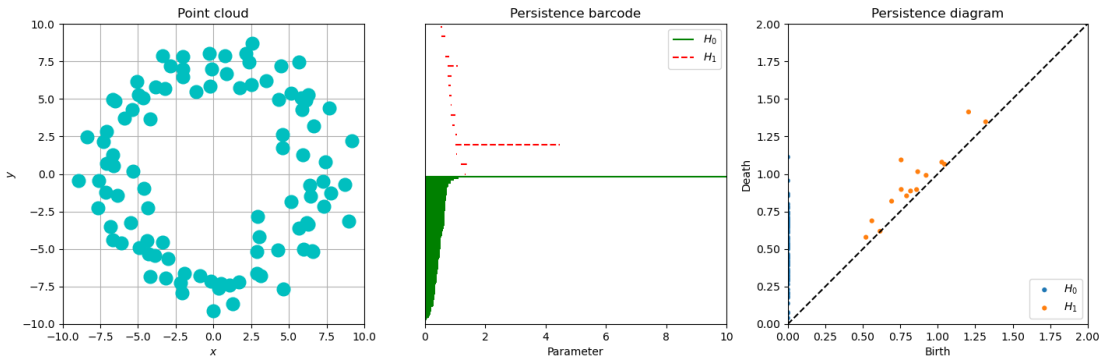


FIGURE 25 – Homologie persistante sur un jeu de 100 données

Le temps de calcul est déjà relativement élevé pour 100 points, on observe une décroissance assez rapide du nombre de composantes connexes, dû à la proximité des différents points. De plus, il existe bien plus de courbures fermées notamment une qui vit bien plus longtemps que les autres, il s'agit du vide situé au milieu de l'ensemble des points, ce dernier met du temps à complétement se remplir et donc à mourir.