Homologie Persistante et application en sécurité informatique

Antoine Mottet

Rapport rédigé dans le cadre de l'Unité d'Enseignement
TIPE de l'Université Claude Bernard Lyon 1
et encadré par Philippe Malbos.

27 juin 2011

Résumé

Ce rapport présente une méthode d'analyse de problèmes relevant de la sécurité informatique basée sur l'utilisation de méthodes issues de la topologie algébrique. Dans un premier temps nous présentons l'homologie simpliciale et l'homologie persistante, et nous donnons des exemples illustrant ces notions. Nous appliquons dans un deuxième temps ces méthodes à un problème de sécurité informatique fourni par la société Picviz Labs.

Table des matières

1 Introduction

L'objectif de ce rapport est de présenter une théorie permettant d'étudier des nuages de points de taille quelconque et de dimension quelconque, dans le but de recueillir des informations *topologiques* à partir de la forme du nuage. Ceci permet de reproduire le phénomène durant lequel le cerveau interprète une image afin d'en extraire des informations. Par exemple, la théorie présentée dans ce rapport peut permettre de reconnaître un tore à partir du nuage de points dans l'espace de la figure 1.

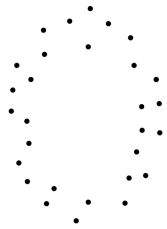


FIGURE 1 – Nuage de points ayant la forme d'un tore

Cette problématique s'inscrit dans plusieurs démarches différentes, ayant trait à plusieurs contextes scientifiques ou d'ingénierie variés.

Dans le domaine de la vision par ordinateur, par exemple, nous serons intéressés par l'extraction des informations à partir d'une image afin de reconnaitre les formes apparaissant sur cette image. Nous devrons donc être capable d'enlever le bruit de l'image, c'est-à-dire d'enlever toutes les données qui ne représente pas un contenu important de l'image, puis nous devrons être en mesure d'identifier les formes restantes.

Dans [1], H. Edelsbrunner présente également une application de l'homologie persistante dans le cas de la mesure de l'expression de gènes chez la souris. Dans ce cas, l'homologie persistante a permis d'identifier quelques gènes responsables du développement cyclique des vertèbres de la souris. Les résultats de cette étude se trouvent dans [2].

L'homologie simpliciale

L'*homologie* est un formalisme mathématique associant des groupes abéliens appelés *groupes d'homologie* à un espace topologique. Elle fait partie d'une discipline plus vaste appelée *topologie algébrique*, qui vise à associer plus généralement des *invariants algébriques* à un espace topologique. Ces invariants algébriques sont des objets associés de telle sorte que les invariants d'espaces homéomorphes sont isomorphes. Par conséquent la présence d'invariants non isomorphes pour deux espaces différents implique que ces espaces ne sont pas homéomorphes.

On déterminera par exemple que les trois premiers groupes d'homologie du tore $T^2 \subset \mathbb{R}^3$ sont respectivement isomorphes à $\mathbb{Z}$, $\mathbb{Z} \oplus \mathbb{Z}$ et $\mathbb{Z}$, et que tous les groupes d'homologie de la boule $B^3 \subset \mathbb{R}^3$ sont nuls à part le premier qui est isomorphe à $\mathbb{Z}$. Par conséquent, la boule et le tore ne sont pas homéomorphes, c'est-à-dire que l'on ne peut pas déformer un tore, sans le déchirer ou sans « coller », pour arriver à obtenir une boule et vice versa.

FIGURE 2 – Un tore et une boule

L'homologie persistante

L'homologie simpliciale ne permet pas de faire l'étude de structures discrètes, ou plutôt on ne peut pas l'utiliser si on désire obtenir des informations importantes de cette structure. En effet, l'homologie simpliciale nous permettrait seulement de déterminer si des nuages sont homéomorphes, mais il suffit de connaître le nombre de points des nuages pour le savoir. C'est pourquoi nous présenterons dans la section 3 l'*homologie persistante*, qui permet d'obtenir plus d'informations sur un nuage de points.

De plus, l'homologie simpliciale ne donne pas de moyen pour différencier des espaces homéomorphes. Elle ne nous permettrait donc pas de distinguer les lettres de la figure 3. En effet, il suffit de déplacer un trait du A pour obtenir un R. Nous verrons comment l'homologie persistante permettra de détecter des différences entre ces lettres, en associant des nuages de points à ces espaces continus.

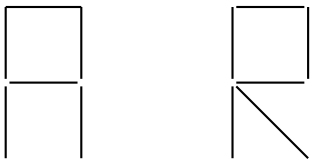


FIGURE 3 – Un A et un R

Plutôt que d'étudier un espace X dans sa globalité, la persistance va consister à considérer une suite de sous-espaces X_n tels que $\emptyset = X_0 \subset X_1 \subset \ldots \subset X_n = X$ et de calculer leur homologie. Au sein de cette suite, des caractéristiques topologiques vont apparaitre et disparaitre, et les groupes d'homologie associés vont nous permettre de considérer ces changements. Ces groupes nous renseigneront sur la « naissance » et la durée de vie de ces données topologiques.

Organisation du document

Dans un premier temps, le but de ce rapport est de donner au lecteur les outils nécessaires à la bonne compréhension de l'homologie simpliciale. Nous définirons ce que sont les *simplexes* et les *complexes simpliciaux* afin de pouvoir déterminer l'homologie d'espaces triangulables, c'est à dire des espaces topologiques homéomorphes à un complexe simplicial. Nous prendrons l'exemple du tore de dimension 2, qui est un exemple d'espace triangulable.

Nous expliciterons ensuite la notion de persistance en l'illustrant d'abord sur un exemple très simple, celui des fonctions réelles à valeurs réelles. Cet exemple nous per-

mettra de définir ce qu'est la persistance. Nous verrons également quels moyens nous avons pour représenter l'homologie persistante d'un objet sur des graphiques, appelés *diagrammes de persistance* ou *codes barre*. Pour finir, cet exemple mettra en évidence la stabilité de l'homologie persistante. En effet, nous verrons que pour deux graphes de fonctions assez similaires, comme ceux représentés dans la figure 4, nous obtiendrons des diagrammes de persistance très proches.

Nous définirons ensuite ce qu'est la persistance dans le cas des complexes simpliciaux.

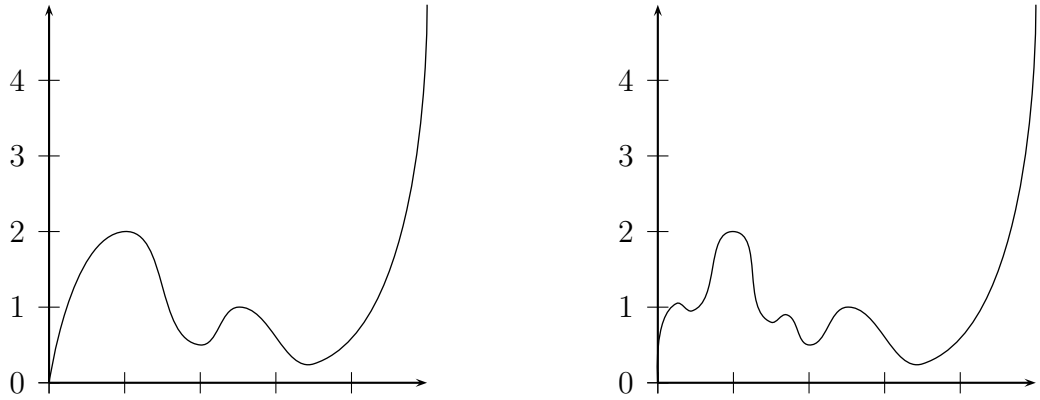


FIGURE 4 – Deux graphes de fonctions ressemblantes

Nous aborderons dans la troisième partie un côté plus effectif de l'homologie persistante. Comme il a été dit plus haut, on cherchera à appliquer l'homologie persistante à des nuages de points. Le point central de la troisième partie sera donc d'arriver à obtenir un complexe simplicial à partir de ces données. Plusieurs méthodes de filtration seront présentées dans cette partie, ainsi qu'un algorithme permettant d'automatiser le calcul de la persistance.

Enfin, la dernière section présente une application de l'homologie persistante à la sécurité informatique. Cette application a eu pour but d'arriver à déterminer si un ordinateur a été ciblé par une attaque informatique. La méthode repose sur l'étude d'un nuage de points générés par cette même machine afin d'en calculer l'homologie persistante. On exposera les difficultés auxquelles nous avons dû faire face au cours de cette expérience, ainsi que les différentes étapes que nous avons suivies.

2 L'homologie simpliciale

L'homologie simpliciale est une des théories de l'homologie qui permet d'associer une suite d'invariants algébriques à tout espace topologique triangulable X, c'est-à-dire à tout espace pour lequel on peut trouver un complexe simplicial homéomorphe à celui-ci.

2.1 Les complexes simpliciaux

2.1.1 Les simplexes

Définition 1. *Un* n-simplexe *est l'enveloppe convexe de* $n+1$ *points de l'espace* $\mathbb{R}^n$*, pris en position générale, c'est-à-dire qu'il n'existe pas d'hyperplan de* $\mathbb{R}^n$ *contenant plus de* n *points.*

C'est donc la généralisation à n dimensions du point, du segment, du triangle et du tétrahèdre qui sont respectivement les 0, 1, 2 et 3 simplexes. Les $n+1$ points formant un n-simplexe sont appelés les sommets.

On notera par la suite $[a_0, a_1, \ldots, a_n]$ le n-simplexe formé de la famille de points $(a_i)_{0\leq i\leq n}$. Le sens des sommets définit une orientation du simplexe de telle sorte que si l'on permute deux sommets adjacents du simplexe on obtient son opposé.

2.1.2 Chaines et faces de simplexes, bord d'une face

Définition 2. *On appelle* n-chaîne *une combinaison linéaire de* n*-simplexes à coefficients dans* $\mathbb{Z}$*.*

On note Δ_n le groupe abélien libre constitué de l'ensemble des n-chaines engendrées par une famille finie $\{\sigma_i, 1 \leq i \leq n\}$ de n-simplexes.

Définition 3. *On définit les* faces *d'un* n*-simplexe* σ *comme étant les* $(n-1)$*-simplexes* σ'_i *tels que les sommets de* σ'_i *soient tous des sommets de* σ*.*

En d'autres termes, les faces d'un n-simplexe $[v_0, \ldots, v_n]$ sont les simplexes de la forme $[v_0, \ldots, \hat{v_i}, \ldots, v_n]$, où $\hat{v_i}$ indique que l'on prend tous les sommets de σ excepté le i-ème.

Grâce à ces deux notions, on peut maintenant définir le morphisme de groupes abéliens $\partial_n : \Delta_n \to \Delta_{n-1}$ qui à un n-simplexe associe la $(n-1)$-chaine constituée par ses faces.

Définition 4. *On appelle* application bord *le morphisme linéaire* $\partial_n : \Delta_n \to \Delta_{n-1}$ *défini par :*

$$\forall\, \sigma \in \Delta_n, \quad \partial_n(\sigma) = \sum_{0\leq i\leq n} (-1)^i \sigma'_i \tag{1}$$

Par convention, le bord d'un 0-simplexe est 0. On appellera *cycle* une chaîne dont le bord est nul.

Les applications bords vérifient une propriété qui sera fondamentale pour la suite.

Propriété 1.

$$\forall\, n \in \mathbb{N}, \quad \partial_n \circ \partial_{n+1} = 0$$

Démonstration. Pour tout n-simplexe $\sigma = [v_0, \ldots, v_n]$ on a :

$$\partial(\sigma) = \sum_{0 \leq i \leq n} (-1)^i [v_0, \ldots, \hat{v_i}, \ldots, v_n]$$

Par conséquent :

$$\begin{aligned}
(\partial \circ \partial)(\sigma) &= \sum_{0 \leq i < n} \sum_{i < j \leq n} (-1)^{i+j} [v_0, \ldots, \hat{v_i}, \ldots, \hat{v_j}, \ldots, v_n] - \\
&\quad \sum_{0 < i \leq n} \sum_{0 \leq j < i} (-1)^{i+j} [v_0, \ldots, \hat{v_j}, \ldots, \hat{v_i}, \ldots, v_n] \\
&= \sum_{0 \leq i < n} \sum_{i < j \leq n} (-1)^{i+j} [v_0, \ldots, \hat{v_i}, \ldots, \hat{v_j}, \ldots, v_n] - \\
&\quad \sum_{0 \leq i < n} \sum_{i < j \leq n} (-1)^{i+j} [v_0, \ldots, \hat{v_i}, \ldots, \hat{v_j}, \ldots, v_n] \\
&= 0
\end{aligned}$$

□

2.1.3 Définition d'un complexe simplicial

Définition 5. *Un* complexe simplicial *K est une union finie de simplexes telle que :*
- *l'intersection de deux simplexes de K est soit vide, soit un simplexe de dimension inférieure ou égale qui est leur face commune de dimension maximale,*
- *toute face d'un n-simplexe de K est un (n-1)-simplexe de K.*

La *dimension* de K est la plus grande dimension des simplexes qui le constituent.

On notera par la suite $\Delta_n(K)$ le groupe abélien libre engendré par les n-simplexes de K.

2.1.4 Exemple

La figure 5 représente un complexe simplicial K de dimension 2.

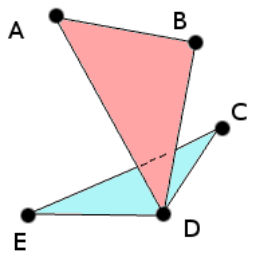


FIGURE 5 – Exemple d'un complexe simplicial

2.2 Groupes d'homologie

2.2.1 L'idée

Soit X un espace topologique dont on souhaite calculer des invariants algébriques. L'idée de l'homologie simpliciale consiste à trouver un complexe simplicial K dont l'espace sous-jacent est homéomorphe à X, puis de calculer ses groupes d'homologie. L'espace X et le complexe K étant homéomorphes, les groupes d'homologie associés à ces deux objets seront isomorphes. Le calcul des groupes d'homologie d'un complexe simplicial K revient à considérer l'ensemble des cycles de K (c'est-à-dire les chaînes de simplexes de bord nul) et distinguer ceux qui forment le bord d'un simplexe de dimension supérieure, de ceux qui ne sont le bord d'aucun simplexe.

2.2.2 Complexe de chaînes, suites exactes

Définition 6. *Un* complexe de chaîne *est une suite de morphismes de groupes abéliens* $(\partial_n)_{n\in\mathbb{N}}$ *tels que :*

$$\forall\, n \in \mathbb{N},\ \partial_n \circ \partial_{n+1} = 0$$

On a donc $\forall\, n \in \mathbb{N},\ Im(\partial_{n+1}) \subseteq Ker(\partial_n)$. On dit qu'un complexe est *exact* si on a égalité entre l'image et le noyau, i.e si l'on a : $\forall\, n \in \mathbb{N},\ Im(\partial_{n+1}) = Ker(\partial_n)$.

2.2.3 Définition des groupes d'homologie

Soient X un espace topologique et K un complexe simplicial homéomorphe à X. Les applications bords $\partial_n : \Delta_n(K) \to \Delta_{n-1}(K)$ définies en 1 sur les simplexes de K forment un complexe de chaîne de la forme :

$$\ldots \xrightarrow{\partial_{n+1}} \Delta_n(K) \xrightarrow{\partial_n} \Delta_{n-1}(K) \xrightarrow{\partial_{n-1}} \ldots \xrightarrow{\partial_2} \Delta_1(K) \xrightarrow{\partial_1} \Delta_0(K) \xrightarrow{\partial_0} 0$$

D'après la propriété 1, nous avons $\forall\, n \in \mathbb{N}, \partial_n \circ \partial_{n+1} = 0$ et par conséquent $Im(\partial_{n+1}) \subseteq Ker(\partial_n)$. Par suite, on peut écrire $Ker(\partial_n)/Im(\partial_{n+1})$.

Définition 7. *On appelle* n-ième groupe d'homologie *d'un espace* X *le groupe quotient* $Ker(\partial_n)/Im(\partial_{n+1})$*, que l'on note* $H_n(X)$*.*

On appelle *n-ième nombre de Betti* et on note β_n le rang du groupe $H_n(X)$.

On montre que le rang du groupe $H_0(X)$ est égal au nombre de composantes connexes de l'espace X.

De plus, dans le cas où X est un espace connexe, le groupe $H_1(X)$ est simplement l'abélianisé du groupe d'homotopie $\pi_1(X)$, [3].

2.2.4 Exemple : l'homologie du tore

Soit T le tore de dimension 2, représenté dans la figure 6. On peut trianguler T avec un complexe simplicial formé d'un 0-simplexe s, de trois 1-simplexes a, b et c, et de deux 2-simplexes R et S. On peut représenter ce complexe K comme dans la figure 7.

Les applications bords forment le complexe suivant :

$$0 \xrightarrow{\partial_3} \Delta_2(K) \xrightarrow{\partial_2} \Delta_1(K) \xrightarrow{\partial_1} \Delta_0(K) \xrightarrow{\partial_0} 0$$

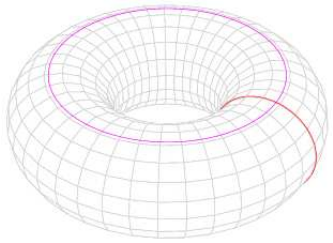


FIGURE 6 – Un tore

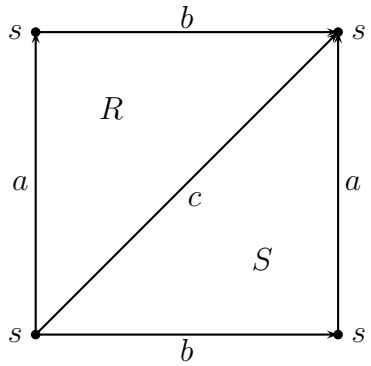


FIGURE 7 – Représentation d'une triangulation du tore

avec :
- $\Delta_0(K) = \mathbb{Z}[s]$
- $\Delta_1(K) = \mathbb{Z}[a, b, c]$
- $\Delta_2(K) = \mathbb{Z}[R, S]$

Calculons le groupe $H_0(T)$. Tout point à un bord nul, et donc $Ker(\partial_0) = \Delta_0(K)$. De plus, $\partial_1(a) = \partial_1(b) = \partial_1(c) = 0$ car les trois 1-simplexes sont des boucles partant de s et arrivant en s. On a donc $H_0(T) = \Delta_0(K) \simeq \mathbb{Z}$ et $\beta_0 = 1$.

Comme dit précédemment, les trois 1-simplexes sont de bord nul, on a donc

$$Ker(\partial_1) = \mathbb{Z}[a, b, c]$$

Il reste désormais à calculer les bords de R et S. On a $\partial_2(R) = \partial_2(S) = a + b - c$, et donc $Im(\partial_2) = \mathbb{Z}[a + b - c]$. Ceci entraine :

$$\begin{aligned} H_1(T) &= Ker(\partial_1)/Im(\partial_2) \\ &\simeq \mathbb{Z} \oplus \mathbb{Z} \end{aligned}$$

et $\beta_1 = 2$.

Pour calculer $H_2(T)$, on remarque que puisque R et S ont le même bord, on a $\partial_2(R - S) = 0$ et donc $Ker(\partial_2) = \mathbb{Z}[R - S]$. De plus ∂_3 est définie sur un groupe nul et donc $Im(\partial_3) = 0$, ce qui nous permet de conclure que $H_2(T) = \mathbb{Z}[R - S] \simeq \mathbb{Z}$ et $\beta_2 = 1$.

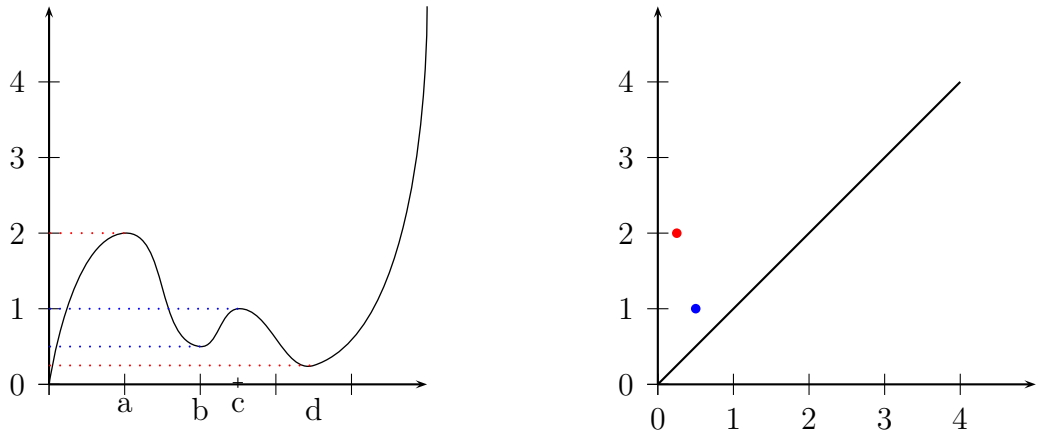


FIGURE 8 – Le graphe de f et le diagramme de persistance associé

Pour finir, K n'est pas constitué de simplexes de dimension supérieure ou égale à 3, ce qui veut dire que le noyau de l'application $\partial_n : \Delta_n \to \Delta_{n-1}$ sera nul, et par conséquent $\forall\, n \geq 3, H_n(T) = 0$ et $\beta_n = 0$.

3 La persistance

3.1 L'idée de la persistance dans les cas simples

3.1.1 Cas d'une fonction réelle

Considérons une application $f : \mathbb{R} \to \mathbb{R}$ et pour un réel t donné, on appelle $\mathbb{R}_t$ la pré-image de $]-\infty; t]$ par f, c'est-à-dire $\mathbb{R}_t = f^{-1}(]-\infty; t])$. L'idée de la persistance consiste à calculer les groupes d'homologie $H_n(\mathbb{R}_t)$ en fonction de t et de reporter ensuite sur un diagramme les valeurs de t pour lesquelles on observe l'apparition ou la disparition d'une classe d'homologie. Pour une fonction réelle à valeurs réelles, nous observons ces changements quand t est égal à un point d'extremum local.

Une classe d'homologie qui apparait pour une valeur i de t et qui disparait pour une valeur j de t sera représentée par le point de coordonnées (i, j). La persistance d'une classe sera le réel $j - i$. Ce diagramme est appelé *diagramme de persistance* puisqu'il va « coder » la persistance des groupes d'homologie de la courbe.

On peut aussi représenter la persistance sous forme de *code barre* dans lequel on représente la vie et la mort d'une classe d'homologie par un trait horizontal commençant en $t = i$ et finissant en $t = j$.

On note $\beta_n^{i,j}$ le nombre de n-classes d'homologie nées en $t \geq i$ et qui ne sont pas mortes en $t = j$. Ce nombre est appelé *nombre de Betti persistant*. Sur un diagramme de persistance, $\beta_n^{i,j}$ est le nombre de points se trouvant dans le quart de plan $[-\infty; i] \times [j; +\infty]$.

3.1.2 Exemple

Soit $f : \mathbb{R} \to \mathbb{R}$ une fonction ayant le graphe donné dans la figure 8. Aux points critiques, le groupe H_0 (dont le rang est égal au nombre de composantes connexes de l'espace) de $\mathbb{R}_t$ va changer. À un minimum local (en b ou d), une composante connexe apparait dans l'ensemble et à un maximum local (a ou c) on observe la fusion de deux composantes connexes.

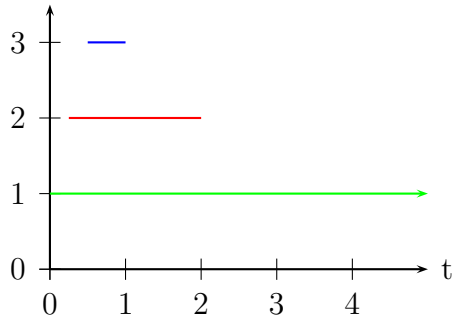


FIGURE 9 – Code barre

Le diagramme de persistance sera donc composé des points de coordonnées $(f(d), f(a))$ et $(f(b), f(c))$ (voir figure 8), et le code barre sera constitué de trois lignes. En effet, il existe une composante connexe dans $\mathbb{R}_t$ qui ne va pas disparaitre. L'utilité du code barre dans ce cas est qu'il montre cette classe, tandis que le diagramme de persistance devrait la représenter avec un point de coordonnées $(0, \infty)$. Une classe qui nait mais ne disparait pas est appelée *classe essentielle*.

3.1.3 Stabilité du diagramme de persistance

Soit g une application réelle ayant pour graphe la courbe de la figure 10. Le diagramme de persistance associé à ce graphe sera assez similaire à celui de la figure 8. En effet, on constate que les $\mathbb{R}_t$ ne seront différents que pour t au voisinage de 1, car deux classes d'homologie vont naitre et disparaitre très rapidement. Ceci signifie que la persistance de ces classes sera proche de 0, et donc que les points représentant ces classes sur le diagramme seront proches de la droite d'équation $y = x$.

Dans les problèmes de modélisation ou de reconnaissance d'images, les classes d'homologie de persistance faible seront considérées comme du « bruit ». En effet, elles ne renseignent pas sur la topologie globale de l'objet considéré, mais plutôt sur des petites variations. Par conséquent, on préfèrera les retirer de l'étude afin de ne conserver que les informations topologiques importantes.

3.2 La persistance pour un complexe simplicial

On s'intéresse maintenant à des objets de toute dimension. Pour un complexe simplicial, on applique la même idée mais cette fois-ci à une *filtration* du complexe. Une

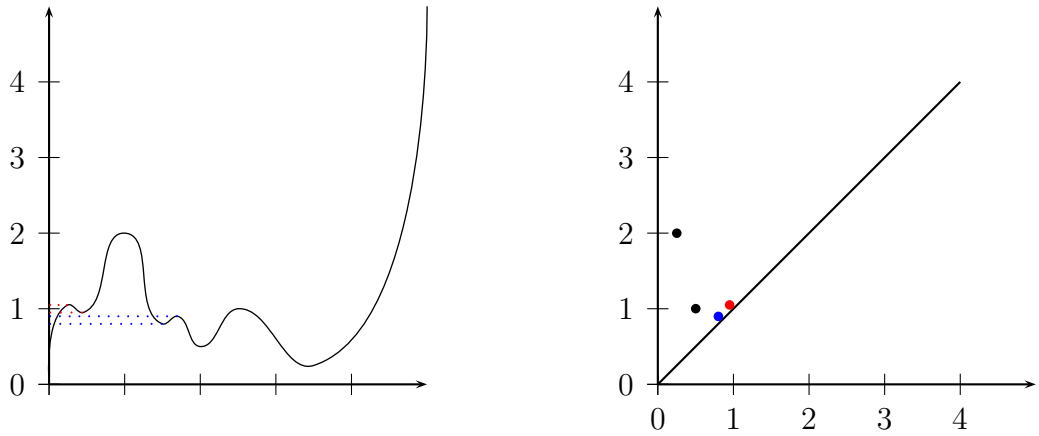


FIGURE 10 – Le graphe de g et le diagramme de persistance associé.

filtration $\mathcal{F}$ d'un complexe K est un ensemble de sous-complexes K_i de K formant une suite telle que

$$\emptyset \subseteq K_1 \subseteq \ldots \subseteq K_n = K$$

Dans le cas d'un complexe simplicial, on calcule les groupes d'homologie de chaque sous-complexe de cette filtration et, comme précédemment, on étudie les modifications apportées à ces groupes au sein de la filtration.

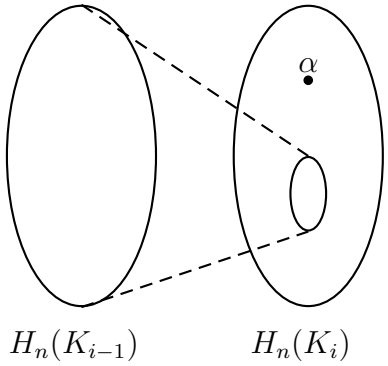


FIGURE 11 – Naissance d'une classe d'homologie

Des exemples de filtration peuvent être la réunion des k-squelettes de K, c'est à dire que chaque K_i sera l'ensemble des k-simplexes avec $k \leq i$, ou bien on pourra numéroter chaque simplexe et prendre $K_i = K_{i-1} \cup \sigma^i$. Pour chaque filtration on obtiendra un point de vue différent du complexe, et nous aurons donc des diagrammes de persistance différents. La question du choix de la filtration sera abordée dans la section 4.

On dira qu'une classe d'homologie « nait » dans K_i lorsqu'elle n'existe pas dans l'image de l'application induite par l'inclusion $K_{i-1} \subset K_i$. On dira qu'une classe née en K_i « meurt » en K_j lorsque qu'elle n'existe pas dans l'image de l'application induite par l'inclusion $K_{i-1} \subset K_{j-1}$ mais qu'elle existe dans l'image de l'application induite par l'inclusion $K_{i-1} \subset K_j$. La naissance et la mort d'une classe d'homologie sont représentées dans les figures 11 et 12.

Ceci signifie qu'une classe vit tant qu'elle n'est pas l'image d'une autre classe née à ou avant i et qu'elle meurt lorsqu'elle fusionne avec une de celles-ci. La persistance de cette classe sera l'entier $j - i$.

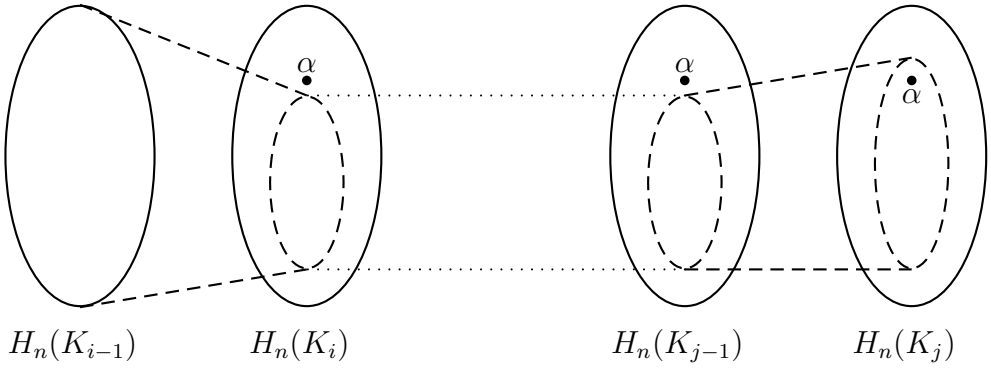


FIGURE 12 – Mort d'une classe d'homologie

3.3 Exemple : homologie persistante des lettres digitales

Dans cette section, nous allons expliquer le fonctionnement de l'homologie persistante sur un exemple basique. Une autre application de l'homologie persistante sera faite dans la section 5 sur un nuage de points beaucoup plus volumineux.

Nous appellerons *lettre digitale* les lettres que nous trouvons sur les réveils ou les montres digitales. Ce sont les lettres pouvant être formées à partir des traits représentés dans la figure 13. Nous poserons que la longueur d'un trait horizontal ou vertical est 1. Posons-nous maintenant le problème suivant. Considérons deux lettres digitales, par exemple le A et le R, réprésentés sur la figure 13. Est-il possible de différencier ces lettres en utilisant des invariants homologiques ? De plus, cette méthode permet-elle de différencier n'importe quelles lettres ?

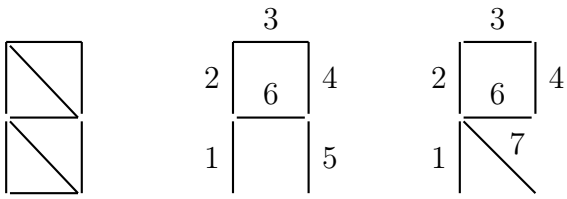


FIGURE 13 – Le A et le R en lettres digitales

Tout d'abord, il est utile de remarquer que le A et le R sont homéomorphes, car on peut déformer une barre du A pour obtenir un R. Les deux lettres auront donc des groupes d'homologies isomorphes. Les groupes $H_0(A)$ et $H_0(R)$ sont engendrés par une seule classe, correspondant à l'unique composante connexe des deux lettres. De plus, les groupes $H_1(A)$ et $H_1(R)$ seront également engendrés par une seule classe, correspondant au 1-cycle. L'homologie simpliciale n'est pas donc pas en mesure de différencier ces lettres.

Nous allons donc appliquer l'homologie persistante à ces deux espaces. Pour cela, nous devons définir un complexe simplicial et une filtration associée. Nous procédons à un échantillonnage des lettres, en ne prenant que les points se trouvant au milieu d'un segment allumé. Avec cet échantillonnage, nous considèrons donc les nuages de points de la figure 14.

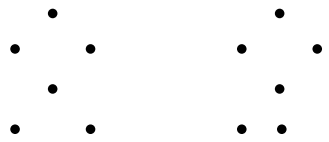


FIGURE 14 – Le A et le R une fois échantillonnés

Nous allons utiliser la méthode du complexe de Rips afin d'avoir une filtration de ces espaces. Cette filtration sera présentée en détail dans la section suivante. Tout ce que nous avons besoin de savoir ici, c'est que la filtration dépend d'un paramètre réel $r \geq 0$ et qu'un simplexe appartient au complexe si, et seulement si, la distance entre chacun de ses points est inférieure à r. Détaillons la démarche pour la lettre A.

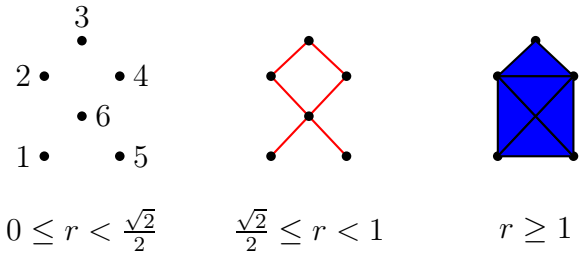


FIGURE 15 – Détail de la filtration de la lettre A

Tout d'abord, pour $0 \leq r < \frac{\sqrt{2}}{2}$, il y a six composantes connexes dans la filtration, i.e chaque point considéré séparément, et donc pour $0 \leq r < \frac{\sqrt{2}}{2}$ le groupe d'homologie $H_0(A_0)$ est engendré par six classes d'homologie. Pour les mêmes valeurs de r, il n'y a aucun segment et donc le groupe $H_1(A_0)$ est nul.

Nous voyons que pour $\frac{\sqrt{2}}{2} \leq r < 1$, des segments liant les points apparaissent (en rouge sur la figure 15), ce qui relie toutes les composantes connexes, et les quatre premiers segments créent un cycle. Par conséquent pour $\frac{\sqrt{2}}{2} \leq r < 1$, les groupes $H_0(A_r)$ et $H_1(A_r)$ sont de rang 1.

Pour finir, à partir de $r = 1$ tous les points sont reliés, ce qui « tue » le cycle existant. La persistance de la lettre A est représentée dans les codes barres de la figure 16.

En appliquant la même méthode pour la lettre R, nous obtenons les codes barre de la figure 17. On remarque donc que les codes barre obtenus sont différents. Il est maintenant logique d'essayer de quantifier cette différence, en définissant une métrique sur les codes barre.

FIGURE 16 – Persistance des groupes $H_0(A)$ et $H_1(A)$

FIGURE 17 – Persistance des groupes d'homologie du R

Définition d'une métrique sur les codes barre

Soit d la distance définie telle que la distance entre deux codes barre est la somme des longueurs des segments se situant sur un diagramme et pas sur l'autre.

Notons $CB_i(X)$ le code barre obtenu pour le groupe d'homologie $H_i(X)$. On remarque que la distance d permet de déterminer que les codes barre $CB_1(A)$ et $CB_1(R)$ sont strictement identiques (i.e $d(CB_1(A), CB_1(R)) = 0$). D'autre part, un simple calcul nous permet de déterminer que

$$d(CB_0(A), CB_0(R)) = \underbrace{\frac{\sqrt{2}}{2}}_{5} + \underbrace{(\frac{\sqrt{2}}{2} - \frac{1}{2})}_{6} + \underbrace{\frac{1}{2}}_{7} = \sqrt{2}$$

On peut prolonger la distance aux lettres en posant

$$d(X, Y) = \sum_{0}^{+\infty} d(CB_i(X), CB_i(Y))$$

et alors nous avons $d(A, R) = \sqrt{2}$.

Nous avons donc réussi à déterminer que les lettres digitales A et R sont différentes au moyen de l'homologie persistante. Il peut être intéressant de savoir si cela fonctionne pour une autre lettre, par exemple le O (figure 18) qui a le même nombre de traits que le A et le R.

On voit que pour $\frac{\sqrt{2}}{2} \leq r < 1$, l'espace ne comptera plus que deux composantes connexes et pour $1 \leq r < \sqrt{2}$ toutes les composantes connexes auront fusionné, créant ainsi un cycle. Pour finir, pour $r \geq \sqrt{2}$ le cycle sera détruit par le segment reliant les points 1 et 4. Les codes barres de la lettre O sont représentés dans la figure 19.

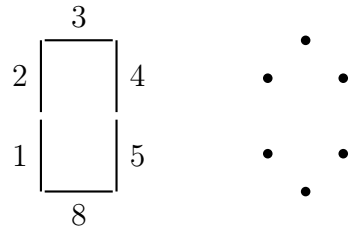


FIGURE 18 – La lettre O et le nuage de point associé

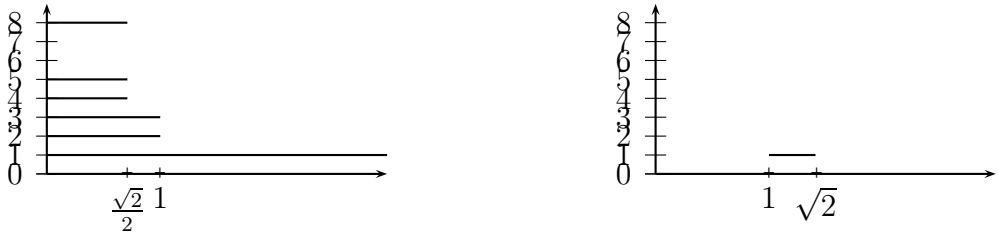


FIGURE 19 – Persistance des groupes $H_0(O)$ et $H_1(O)$

En s'intéressant aux distances obtenues entre les codes barre, nous obtenons

$$d(CB_0(A), CB_0(O)) = 2$$

et

$$d(CB_1(A), CB_1(O)) = 1 + \frac{\sqrt{2}}{2}$$

Par conséquent nous avons $d(A, O) = 3 + \frac{\sqrt{2}}{2}$. Ici encore, nous avons donc pu différencier les deux lettres.

4 Méthodes de filtration et algorithmes de calcul

4.1 Introduction

Nous avons vu dans la section précédente comment calculer la persistance pour un complexe filtré, mais sans mentionner de méthode de construction d'un complexe et d'une filtration en premier lieu. En pratique, l'espace topologique considéré sera un nuage de points pouvant provenir d'un échantillonnage fait à partir de données continues, par exemple un signal audio, ou provenant d'une collecte de données discrètes fournies par des capteurs. Dans la section 5 nous étudierons par exemple un nuage de 10000 points généré par un ordinateur. Plusieurs méthodes ont été conçues pour déterminer un complexe et sa filtration à partir d'un échantillon de points (complexe de Delaunay, complexe de Rips, α-complexe, ...) et toutes ne peuvent être traitées ici. Ces méthodes sont décrites dans [4, 5, 6].

4.2 Le complexe de Rips

Soient N un nuage de p points de $\mathbb{R}^n$ et d une métrique sur $\mathbb{R}^n$. La méthode de Rips consiste à dire qu'une famille $\{a_0, \ldots, a_k\}$ de points de N définit un k-simplexe si, et

seulement si, on a pour un réel $r > 0$ fixé :

$$\forall\, 0 \leq i, j \leq k, \quad d(a_i, a_j) \leq r$$

L'ensemble de ces simplexes sera noté $Rips(N, r)$.

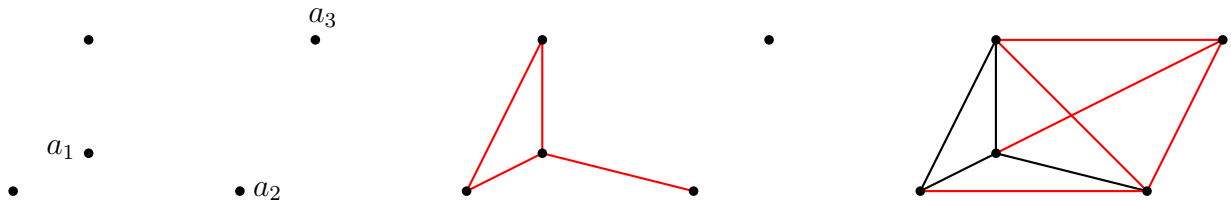
Pour tout $r' \geq r$, nous avons $Rips(N, r) \subseteq Rips(N, r')$. Si en plus on définit $Rips(N, 0) = \emptyset$, l'ensemble

$$\mathcal{F} = \{Rips(N, r),\ 0 \leq r \leq r_{max}\}$$

définit une filtration paramétrée par un réel, où $r_{max} \in \mathbb{R}_+$ est tel que $Rips(N, r_{max})$ est le $p-1$-simplexe formé par les p points de N.

De plus, on peut remarquer qu'un n-simplexe appartient au complexe si, et seulement si, toutes ses faces appartiennent également au complexe. Ceci signifie qu'on peut ne calculer que le 1-squelette du complexe, et qu'on peut ensuite déterminer les simplexes de dimension supérieure uniquement grâce aux simplexes de dimensions 0 et 1. Les complexes vérifiant cette propriété sont appelés *flag complexes*.

Cette méthode est très peu utilisable en pratique à partir du moment où nous devons travailler par ordinateur sur des nuages de points qui peuvent contenir plusieurs millions de points. En effet, le nombre de simplexes entrant dans la filtration croit très rapidement avec r, et ceci nécessite des capacités en mémoire informatique très importantes.



$0 \leq r < d(a_1, a_2)$ $\qquad$ $d(a_1, a_2) \leq r < d(a_1, a_3)$ $\qquad$ $r \geq d(a_1, a_3)$

FIGURE 20 – Construction d'un complexe de Rips

Dans la figure 20 sont présentées trois étapes de la construction d'un complexe de Rips à partir d'un nuage $N = \{a_1, a_2, a_3, a_4, a_5\}$ de quelques points. Quand trois sommets sont reliés, le 2-simplexe défini par ces trois sommets appartient au complexe, et quand quatre sommets sont reliés le tétraèdre plein entre dans le complexe.

Le complexe final est constitué de 5 points, 9 segments, 7 triangles et 2 tétraèdres, soit 23 simplexes, alors que nous n'avions que cinq points dans le nuage. Ceci illustre le fait que pour des nuages constitués de millions de points, le complexe de Rips ne peut pas être utilisé pour effectuer des calculs.

4.3 Le Witness Complex

La méthode dite du *Witness Complex* est une alternative au complexe de Rips qui a pour but de rendre calculable l'homologie persistante d'un nuage de points volumineux. Elle consiste à considérer un sous-ensemble de points du nuage, qui seront les seuls points à pouvoir entrer dans la filtration. Les autres points nous donneront un critère pour déterminer si une famille de points décrit un simplexe ou non.

Soient $N \subset \mathbb{R}^n$ un nuage de p points, L un sous-ensemble arbitraire de q points de N et d une métrique sur $\mathbb{R}^n$. Nous allons associer à N un ensemble de complexes simpliciaux de la manière suivante. Soient $r > 0$ un réel et $v \leq q$ un entier naturel. Chaque sous-ensemble $\{a_0, \ldots, a_k\} \subset L$ décrit un k-simplexe si, et seulement si, il existe un point ω de $N \backslash L$ tel que

$$max\{d(a_0, \omega), \ldots, d(a_k, \omega)\} \leq r + \delta_v$$

où δ_v est la v-ème plus petite distance entre ω et un point de L. Le cas échéant ce point est appelé *témoin* du simplexe $[a_0, \ldots, a_k]$.

L'ensemble de ces simplexes est alors noté $W(N, r, v)$ et nous avons toujours l'implication

$$r' \geq r \Rightarrow W(N, r, v) \subseteq W(N, r', v)$$

Dans [7], Carlsson utilise cette filtration pour retrouver les groupes d'homologie d'une 2-sphère après échantillonnage de celle-ci. Dans cet exemple, il ajoute que choisir $v \in \{0, 1, 2\}$ permet d'obtenir les meilleurs résultats.

Il peut être pratique, pour l'implémentation de cette méthode de filtration dans un logiciel, de considérer la matrice $\mathbf{D}$ où $\mathbf{D}[k, j] = d(a_k, a_j)$ avec $a_k \in L$ et $a_j \in N \backslash L$. Le paramètre δ_v est alors le v-ème plus petit scalaire dans la colonne de $\mathbf{D}$ représentant les distances entre le témoin et les points de L.

4.4 Le Lazy Witness Complex

Le Lazy Witness Complex est une simple adaptation du Witness Complex afin de vérifier la propriété des *flag complexes*. On dira alors qu'un couple $\{a_1, a_2\}$ de points décrit un segment du complexe si, et seulement si,

$$max\{d(a_1, \omega), d(a_2, \omega)\} \leq r + \delta_v$$

Ensuite, toute famille de points de cardinal supérieur à 2 décrit un simplexe si, et seulement si, toutes les faces appartiennent au complexe simplicial.

4.5 Algorithme de calcul de la persistance

L'algorithme décrit dans cette section a pour but de calculer l'homologie persistante d'un complexe simplicial. C'est le tout premier algorithme de calcul rapide ayant vu le jour dans ce domaine [8]. La complexité en temps de cet algorithme est en $\mathcal{O}(n^3)$, n étant le nombre de simplexes apparaissant dans le complexe simplicial final. L'algorithme permet à la fois de déterminer les nombres de Betti et le diagramme de persistance.

Soit K un complexe simplicial composé de n simplexes notés σ_i avec $1 \leq i \leq n$. Soit de plus $\mathcal{F} = \{K_r, r \in \mathbb{R}\}$ une filtration associée à ce complexe. Pour finir, soit $\mathbf{D}_\partial$ la matrice de l'application bord appliquée à l'ensemble des simplexes de K, telle que la i-ième colonne de la matrice représente la chaine formée par les faces du simplexe σ_i. Le coefficient $\mathbf{D}_\partial[j, i]$ sera 1 si σ_j est une face de σ_i et 0 sinon.

La première étape consiste à ré-ordonner les simplexes par rapport à leur ordre d'entrée dans la filtration. Ceci nous assure que si σ_i est une face de σ_j, alors $i < j$. Par conséquent ∂ est une matrice triangulaire strictement supérieure.

On définit l'entier $low(i)$ comme étant le numéro de la ligne où apparait le dernier 1 de la colonne i, ou 0 si la colonne i est nulle. ∂ étant triangulaire supérieure, on a nécessairement $low(i) < i$ pour tout entier $1 \leq i \leq n$. De plus, on dira d'un simplexe qui fait partie d'un cycle qu'il est *positif*, et dans le cas contraire qu'il est *négatif*.

L'algorithme va consister à apparier chaque simplexe qui crée une classe d'homologie avec le simplexe qui détruira cette même classe afin d'en déterminer la persistance. Dans le cas où un simplexe créant une classe n'est apparié à aucun simplexe, on sera en présence d'une classe essentielle. Chaque paire sera appelée *paire de persistance*.

Pour ceci, nous allons effectuer des opérations sur les simplexes afin de mettre en évidence les simplexes positifs et les simplexes négatifs, afin de déterminer les paires de persistance. Ces opérations sur les simplexes vont se traduire par des opérations sur les colonnes de la matrice $\mathbf{D}_\partial$. La matrice réduite de $\mathbf{D}_\partial$ sera notée $\mathbf{R}$. Voici l'algorithme :

$\mathbf{R} \leftarrow \mathbf{D}_\partial$
for $j = 1 \rightarrow m$ **do**
  while il existe $j_0 < j$ tel que $low(j_0) = low(j)$ **do**
    $Colonne(j) \leftarrow Colonne(j_0) + Colonne(j) \mod 2$
  end while
end for

Les paires de persistance seront, à l'issue de cet algorithme, les paires $(\sigma_{low(i)}, \sigma_i)$ avec i tel que $low(i) \neq 0$.

Il y a ici plusieurs points à vérifier afin de montrer la validité de l'algorithme :

- tout d'abord, si $low(i) = 0$, alors σ_i est un simplexe positif. En effet, à chaque étape de l'algorithme la colonne i représente le bord de $\sigma_i + \sum_{k<i} \epsilon_k \sigma_k$, avec $\epsilon_k \in \{0, 1\}$. On a donc $\partial(\sigma_i + \sum_{k<i} \epsilon_k \sigma_k) = 0$, c'est-à-dire que σ_i fait partie d'un cycle, i.e c'est un simplexe positif.
- de plus, si $low(i) = j \neq 0$, σ_j est un simplexe positif. En effet, nous avons cette fois

$$\partial(\sigma_i + \sum_{k<i} \epsilon_k \sigma_k) = \sigma_j + \sum_{p<j} \epsilon_p \sigma_p$$

  ce qui signifie que σ_j fait partie d'un cycle (car $\partial \circ \partial = 0$), et donc c'est un simplexe positif. Comme σ_j est le dernier simplexe de la chaine $\sigma_j + \sum_{p<j} \epsilon_p \sigma_p$ à être arrivé dans la filtration, c'est aussi lui qui crée ce cycle.
- pour finir, si $low(i) = j \neq 0$, alors σ_i tue la classe d'homologie créée par σ_j. En effet, avant que σ_i n'apparaisse dans la filtration, σ_j n'est associé à aucun autre simplexe

(sinon on aurait un $i' < i$ tel que $low(i') = low(i)$, et par conséquent l'algorithme pourrait encore continuer). Ce n'est donc qu'à l'arrivée de σ_i que σ_j est apparié, et donc σ_i tue la classe d'homologie qui avait été créée par σ_j.

Exemple

Étudions l'algorithme sur un exemple. Soit le complexe simplicial K constitué d'un triangle plein (avec les arêtes et les points). La filtration consistera en l'ajout des trois points, puis des trois segments et enfin du 2-simplexe final (voir figure 21). Les groupes abéliens libres des n-chaines seront les suivants :

- $\Delta_0(K) = \mathbb{Z}.[\sigma_1, \sigma_2, \sigma_3]$
- $\Delta_1(K) = \mathbb{Z}.[\sigma_4, \sigma_5, \sigma_6]$
- $\Delta_2(K) = \mathbb{Z}.[\sigma_7]$

La matrice de l'application bord associée à ce complexe est la suivante :

$$\mathbf{D}_\partial = \begin{array}{c} \begin{array}{ccccccc} \partial(\sigma_1) & \dots & \dots & \partial(\sigma_4) & \dots & \dots & \partial(\sigma_7) \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

On remarque immédiatement que les trois premières colonnes sont nulles, et donc que l'ajout des trois points créent trois 0-classes d'homologie (c'est-à-dire trois composantes connexes). D'après les termes définis ci-dessus, ce sont trois simplexes positifs.

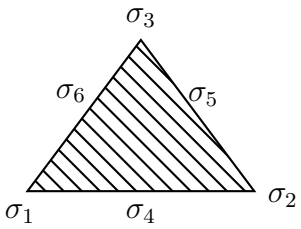


FIGURE 21 – Le triangle plein

Le bord de σ_4 est constitué de σ_1 et σ_2. C'est donc un simplexe négatif, qui est apparié à σ_2 car $low(4) = 2$. Le même raisonnement est fait pour σ_5, simplexe négatif apparié à σ_3.

Le simplexe σ_6 est plus particulier. En effet, son bord est constitué de σ_1 et σ_3, et $low(6) = 3$. Or $low(5) = 3$, ce qui signifie que σ_3 a déjà été apparié plus tôt dans la filtration. On effectue donc l'addition modulo 2 des colonnes 5 et 6, ce qui donne la matrice suivante :

$$\begin{array}{c} \\ \\ \\ \\ \\ \\ \\ \\ \end{array} \begin{array}{c} \\ \\ \\ \\ \partial(\sigma_5) \\ \\ \\ \\ \end{array} \begin{array}{c} \\ \\ \\ \\ \partial(\sigma_5+\sigma_6) \\ \\ \\ \\ \end{array} \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Désormais $low(4) = low(6)$ donc on réitère la procédure, afin d'obtenir la matrice finale suivante :

$$R = \begin{array}{c} \\ \\ \\ \\ \\ \\ \\ \\ \end{array} \begin{array}{c} \\ \\ \\ \\ \partial(\sigma_5) \\ \\ \\ \\ \end{array} \begin{array}{c} \\ \\ \\ \\ \partial(\sigma_4+\sigma_5+\sigma_6) \\ \\ \\ \\ \end{array} \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Pour finir, $low(6) = 0$ donc un cycle est créé. Ce cycle est le « trou » au milieu du triangle, et ce trou est comblé par l'apparition de σ_7. En effet, on a bien $low(7) = 6$.

Nous pouvons maintenant déterminer les diagrammes de persistance associés à ce complexe. Durant l'algorithme, nous avons créé les paires de simplexes (σ_2, σ_4), (σ_3, σ_5) et (σ_6, σ_7). Les deux premières décrivent la naissance et la mort de 0-classes d'homologie et la dernière représente la naissance et la mort d'une 1-classe d'homologie. Les coordonnées des points sur les diagrammes de persistance seront donc $(2, 4)$, $(3, 5)$ d'une part et $(6, 7)$ d'autre part. Les diagrammes correspondant sont représentés dans la figure 22.

FIGURE 22 – Représentation des diagrammes de persistance du triangle plein

5 Application de l'homologie persistante aux réseaux informatiques

5.1 Le contexte

On se propose d'essayer d'appliquer l'homologie persistante dans le cadre de la sécurité informatique. Philippe Saadé, de la société Picviz Labs[1] spécialisée dans le domaine de l'analyse de données, a fourni trois *logs d'écoute* de machines informatiques. Un log d'écoute est un fichier généré par un ordinateur, dans lequel celui-ci enregistre toutes les informations lui ayant été envoyées à travers un réseau informatique. Les machines ayant généré ces logs ont subi des attaques appelées *scans de ports* durant l'écoute.

Brièvement, ces scans peuvent permettre à un pirate informatique de savoir quels sont les endroits les plus faibles de la machine qu'il cible, en vue d'une attaque plus agressive. Il existe plusieurs logiciels permettant d'effectuer des scans de ports et ils n'emploient pas tous la même stratégie pour mener l'attaque. Les trois logs de Philippe Saadé ont été générés par trois logiciels différents, adoptant trois stratégies différentes.

En plus de ces trois logs, P. Saadé a procuré un log que nous appellerons *log sain*, celui d'une machine n'ayant pas subi de scan de ports. Nous n'avions pas connaissance de quel log était sain parmi les quatre. La problématique qui était posée était de savoir si au moyen de l'homologie persistante il était possible de séparer d'une part le log sain des trois autres. De plus, il était également proposé d'essayer de différencier les logs malsains entre eux.

Les logs consistaient en un enregistrement de 10000 *paquets* ayant circulé à travers la machine cible. Un paquet est un ensemble d'informations contenant des données (contenu d'une page Web, d'un fichier, etc...) ainsi que des informations techniques sur l'émetteur de l'information, son destinataire, l'état du réseau, etc...

Pour chaque paquet, on avait à notre à disposition quatre informations :
- l'adresse IP source,
- l'adresse IP de destination,
- le port source,
- le port de destination

sans savoir dans quel ordre les champs étaient placés.

Ces données étaient enregistrées sous forme de *double*, c'est-à-dire sous forme de nombres décimaux tous compris entre 0 et 1. Nous considèrerons ces données comme des nuages de points de $[0,1]^4 \subset \mathbb{R}^4$.

5.2 Première approche

La première étape à suivre était de tenter de se représenter ces nuages. En effet, il fallait tout d'abord essayer de différencier les logs à vue d'oeil, afin de comprendre de manière plus fine la forme globale des nuages de points.

On a programmé dans ce but un logiciel implémentant deux méthodes de visualisation différentes. Tout d'abord, on a représenté le nuage en coordonnées parallèles. Dans

1. `http://www.picviz.com`

cette représentation, on dessine quatre axes parallèles représentant les coordonnées des points, et chaque point se dessine sous forme d'une ligne brisée coupant les axes suivant ses coordonnées. Un exemple de cette représentation est donné dans la figure 23, et les représentations obtenues pour les quatre logs sont données dans la figure 24.

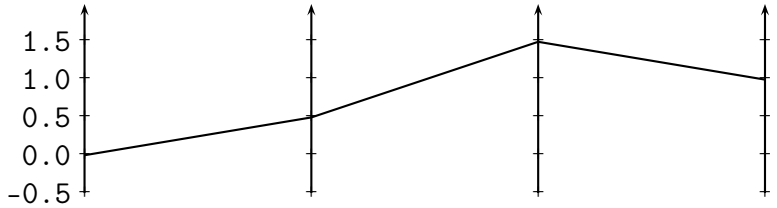


FIGURE 23 – Représentation du point `(0,0.5,1.5,1)` en coordonnées parallèles

La seconde méthode a été de projeter le nuage dans $\mathbb{R}^3$ à partir d'un point Ω de $\mathbb{R}^4$ que nous appellerons *pôle*. Chaque point $P = (x, y, z, t)$ du nuage est projeté à l'intersection de la droite (ΩP) et de l'hyperplan d'équation $t = 0$, c'est-à-dire l'espace usuel $\mathbb{R}^3$. Par analogie avec quelque chose de plus familier, c'est comme si l'on plaçait une source de lumière derrière un objet en trois dimensions et que l'on observait l'ombre de cet objet sur un plan. On perd donc de l'information en utilisant cette projection. Cependant, le logiciel utilisé est capable de réaliser la projection avec un pôle se déplaçant sur la sphère unité $S^3 \subset \mathbb{R}^4$, afin de nous permettre d'avoir plusieurs "clichés" différents du nuage de points. Grâce à ceci, nous avons pu obtenir des graphiques montrant de manière assez significative certaines différences entre les logs. Les projections des quatre nuages générées par le logiciel sont données par la figure 25.

Grâce à la représentation en coordonnées parallèles, nous avons pu déterminer que les coordonnées (x, y, z, t) d'un point représentaient l'IP source, le port source, l'IP de destination et le port de destination (dans cet ordre). En effet, le fonctionnement des réseaux implique une symétrie sur les graphiques. Ceci est dû au fait que les ordinateurs du réseau sont tour à tour source puis destination. En exploitant cette symétrie on peut donc regrouper les axes par paires (Adresse IP, Port). Pour finir, les valeurs des ports sont plus dispersées que celles des adresses IP, ce qui nous permet de déterminer dans chaque paire quel est l'axe des ports et quel est celui des adresses IP.

De plus, nous avons pu isoler le log sain des trois autres logs. En effet, dans un réseau, un déroulement normal est constitué d'échanges où la plupart des paquets envoyés reçoivent une réponse, ou un accusé de réception, de la part du destinataire. Sur un graphique, ceci induit une symétrie presque parfaite, symétrie que l'on retrouve dans le graphique en haut à gauche de la figure 24.

La projection dans $\mathbb{R}^3$ a permis de nous donner une idée des caractéristiques topologiques du nuage de points dans $\mathbb{R}^4$. Il y a plusieurs composantes connexes sur chacun des graphes, avec des zones de densité assez élevée pour les fichiers malsains. Il s'agit donc maintenant d'essayer de mettre en évidence ces caractéristiques en calculant l'homologie persistante des nuages de points afin de les différencier.

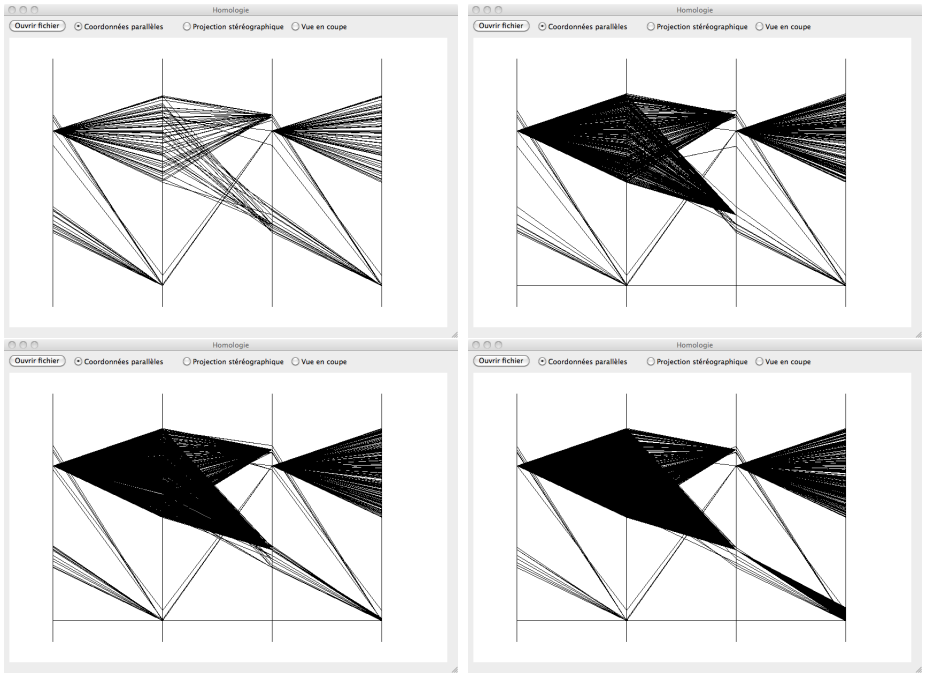


FIGURE 24 – Les représentations en coordonnées parallèles des quatre logs, avec en haut à gauche le log sain puis les trois logs infectés.

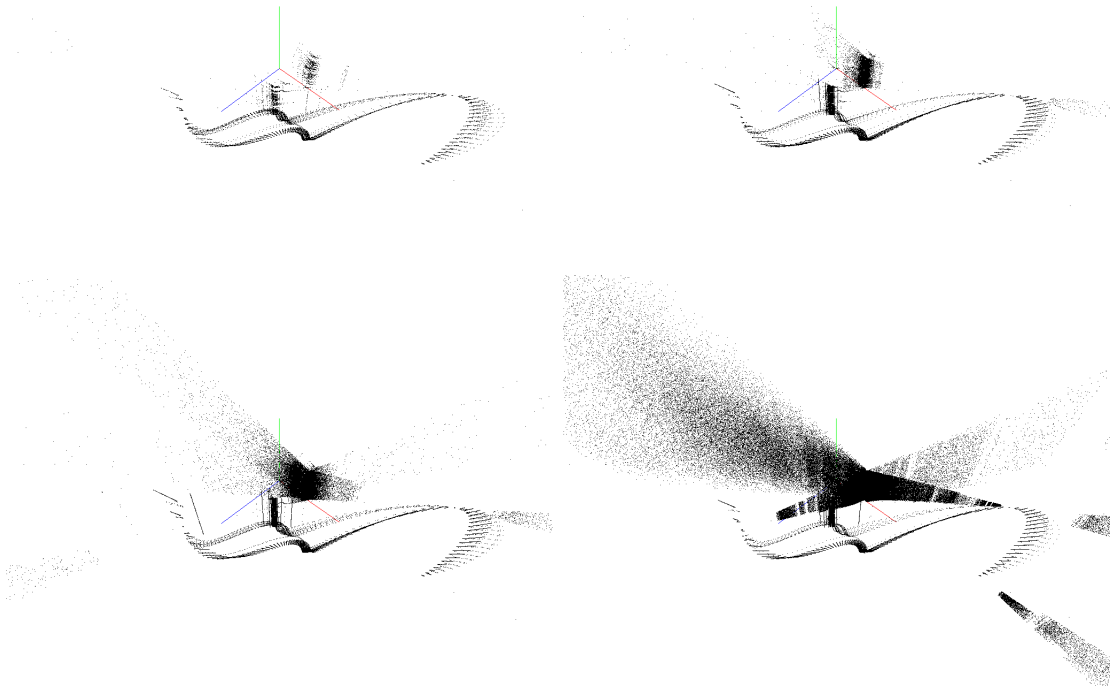


FIGURE 25 – Les projections des logs dans $\mathbb{R}^3$, avec en haut à gauche le log sain puis les trois logs infectés.

5.3 Calculs de l'homologie persistante

Afin d'effectuer ces calculs, nous avons utilisé le logiciel PLEX [2]. PLEX est un logiciel développé par Vin De Silva et Gunnar Carlsson à l'université de Stanford. Ce logiciel a été créé dans le but de pouvoir calculer l'homologie persistante de nuages de points en dimension quelconque. Les méthodes de filtration implémentées sont celles présentées dans la section 4 (c'est-à-dire le complexe de Rips, le Witness Complex et le Lazy Witness Complex).

Les premiers essais de calculs ont été réalisés avec la méthode du complexe de Rips. Étant donné le volume des données (10000 points), cette méthode est immédiatement apparue comme inapplicable avec un ordinateur classique disposant d'une mémoire vive de 2 Go. En effet, le nuage étant assez condensé, plusieurs dizaines de millions de simplexes apparaissent en très peu de temps dans la filtration, exigeant une capacité en mémoire très supérieure à celle dont nous disposions au moment de l'expérience. Il sera probablement instructif de réitérer cette expérience avec des moyens plus importants.

Tous les essais suivants ont donc été réalisés avec la méthode du Lazy Witness Complex, qui à l'avantage d'être un flag complex (comme le complexe de Rips) et de n'utiliser qu'un nombre réduit de points pour faire les calculs. Un des inconvénients de cette méthode, ou tout du moins de l'implémentation qui en est faite dans PLEX, est que l'on extrait le nuage L des points de repères de manière aléatoire. Ceci étant, il nous est impossible de conclure sur l'origine du nuage à partir des diagrammes de persistance.

En effet, les expériences ont montré qu'en effectuant les calculs plusieurs fois pour un même fichier, nous obtenons des diagrammes différents. Ces différences sont trop importantes (classes d'homologie de durée de vie très différente, nombre de classes essentielles différent, ...) pour être négligées.

Il sera donc intéressant d'implémenter dans PLEX une méthode permettant d'extraire un échantillonnage du nuage initial à partir des caractéristiques de celui-ci (dispersion des points, courbure du nuage, ...) et produisant un résultat unique. De plus, les expériences ont été réalisées avec un échantillon de 50 points sur les 10000, ce qui est faible. Une quantité supérieure de points pourrait être intéressante à exploiter, mais ceci nous ramène à l'incapacité de réaliser les calculs sur la machine utilisée.

5.4 Conclusion

Cette expérience n'a donc pas aboutie pour l'instant. Cependant, nous avons déterminé plusieurs pistes à suivre afin d'essayer de répondre à la problématique donnée, comme l'utilisation de machines de calcul plus puissantes, ou l'élaboration d'une stratégie pour choisir les points utilisés dans la méthode du Lazy Witness Complex.

De plus, dans [9], Afra Zomorodian donne de nouveaux algorithmes permettant un calcul rapide et moins gourmand en ressources des complexes de Rips à partir de nuages de points volumineux. Le meilleur de ces algorithmes permet un calcul 3 fois plus rapide

2. `http://comptop.stanford.edu/u/programs/jplex/index.html`

qu'avec PLEX et 5 fois moins consommateur de mémoire. En outre, un de ces algorithmes, bien que plus lent, est parallélisable. Étant donné que les grilles de calculs, c'est-à-dire les réseaux d'ordinateurs conçus pour réaliser de manière distribuée une tâche donnée, deviennent de plus en plus faciles d'accès, cet algorithme pourrait théoriquement procéder à la construction d'un complexe de Rips en un temps réduit.

Ces pistes de recherche seront étudiées par la suite lors d'un stage effectué dans la société Picviz Labs.

Références

[1] Herbert EDELSBRUNNER et John HARRER : *Computational Topology*, pages 199–206. American Mathematical Society, 2010.

[2] Olivier POURQUIÉ et AL. : Comparison of pattern detection methods in microarray time series of the segmentation clock. *PLoS ONE*, 3.8, 2008.

[3] Allen HATCHER : *Algebraic Topology*, pages 21–40. Cambridge University Press, 2002.

[4] Herbert EDELSBRUNNER et Nimish SHAH : Triangulating topological spaces. *In SCG '94 Proceedings of the tenth annual symposium on Computational geometry*, pages 285–292. Association for Computing Machinery, 1994.

[5] Herbert EDELSBRUNNER : Shape reconstruction with delaunay complex. *In LATIN '98 Proceedings of the Third Latin American Symposium on Theoretical Informatics*, pages 119–132. Springer-Verlag Berlin, 1998.

[6] Herbert EDELSBRUNNER et John HARRER : *Computational Topology*, pages 63–74. American Mathematical Society, 2010.

[7] Vin DE SILVA et Gunnar CARLSSON : Topological estimation using witness complexes. *In Symposium on Point-Based Graphics*. `https://diglib.eg.org/EG/DL/WS/SPBG/SPBG04`, 2004.

[8] Herbert EDELSBRUNNER, David LETSCHER et Afra ZOMORODIAN : Topological persistence and simplification. *Discrete computational geometry*, 28:511–533, 2002.

[9] Afra ZOMORODIAN : Fast construction of the Vietoris-Rips Complex. *Computer and Graphics*, 34, 2010.