Projet Kernel-CAH

Professeur M. Patrice Bertrand

Daniel Haik

Juin 2020

Abstract

Dans ce rapport nous allons résumer les principales idées parues dans deux articles scientifiques portant sur des variantes de l'algorithme de classification ascendante hiérarchique (CAH).

Contents

Références **12**

Dans ce rapport nous allons résumer les principales idées parues dans deux articles scientifiques portant sur des variantes de l'algorithme de classification ascendante hiérarchique (CAH). Le premier article s'intitule **"Kernel hierarchical agglomerative clustering: Comparison of different gap statistics to estimate the number of clusters"** (voir section de References (**art.1**)) et le deuxième **"Similarity Based Hierarchical Clustering with an Application to Text Collections"** (voir section de References **(art.2)**). En premier lieu nous rappellerons en quoi consiste l'algorithme classique CAH, puis nous présenterons l'algorithme du Kernel CAH, ensuite nous décrirons différents critères de sélection du nombre optimal de classes et nous finirons par présenter une famille de distances interclasses. A la fin du rapport nous pourrons trouver les résultats numériques obtenus en implémentant certains algorithmes présentés.

Classification Ascendante Hierarchique

L'algorithme de classification ascendante hierarchique (CAH) est un algorithme de classification non supervisé. Etant donné un ensemble d'objects de même nature ces algorithmes regroupent ces objects en K groupes (ou classes) distincs, où K est un entier naturel fixé à l'avance.

Contrairement aux algorithmes de classification supervisés (comme les réseaux de neurones par exemple), ces algorithmes ne disposent pas des variables expliquées, ils classent les objets uniquement à l'aide des variables explicatives. Ce type d'algorithme peut être utile lorsque l'on souhaite découvrir des classes étant donné que nous ignorons la répartition des données.

Definition

Soit $(E, <, >)$ un espace vectoriel réel de dimension finie muni d'un produit scalaire.

L'ensemble des élèments à classer est représenté par un ensemble fini $\Omega = \{ x_1, \ldots, x_n \} \subset E^n$ avec $n > 1$.

Tous les individus sont munis d'un poid p_i tel que $\sum_{i=1}^{n} p_i = 1$, on prends généralement $p_i = \frac{1}{n}$.

Classes

Soit $K \in [1, n]$.

On appelle classification en K classes une collection d'ensemble $C = C_1, \ldots, C_K$ telle que :

- $\forall\ k = 1, \ldots, K$, $C_k \subset \Omega$,
- $\cup_{k=1,..,K}\ C_k = \Omega$,
- $C_i \cap C_j = \emptyset$, pour $i \neq j$.

On définit $g_k = \frac{1}{\sum_{i \in I_k} pi} \sum_{i \in I_k} pixi$ le barycentre de la classe C_k et l'ensemble $I_k = \{ i, x_i \in C_k \}$, les indices des élèments de C_K.

Hiérarchie

On appelle hiérarchie H un ensemble d'ensemble tel que :

- $\Omega \subset H$,
- $\{ x_i \} \subset H$ pour tout $i = 1, \ldots, n$,
- $\forall\ (h,h') \in H^2$, soit $h \cap h' = \emptyset$, soit $h \subset h'$ ou soit $h' \subset h$.

Dissimilarité

On note d : E x $E \to \mathbb{R}_+$ l'application distance issue du produit scalaire de E

Etant donné la distance entre deux individus, on peut définir une nouvelle distance entre deux classes appelée distance de "dissimilarité" et notée : $dissim$: C x $C \to \mathbb{R}_+$.

Voici des exemples de dissimilarité :

- lien minimal : $dissim(C_1,C_2) = \min_{x \in C_1, y \in C_2} d(x,y)$
- lien maximal : $dissim(C_1,C_2) = \max_{x \in C_1, y \in C_2} d(x,y)$
- distance de Ward : $dissim(C_1,C_2) = \frac{n_1 n_2}{n_1+n_2} ||g_1 - g_2||^2$, où $n_k = \text{card}(I_k)$, pour $k = 1, ..., n$.

Problème de classification

Le but des algorithmes de classification non supervisé est de créer des classes de façon à ce qu'elles soient le plus hétérogènes que possible les uns des autres, autrement dit résoudre le problème suivant :

$$(P) \quad \min_{C_1,\ldots,C_K} \sum_{k=1}^{K} \sum_{i \in I_k} (x_i - g_k)^2$$

avec,

- $I_k = \{\ i,\ x_i \in C_k\ \}$, les indices des élèments de C_k,
- $g_k = \frac{1}{\sum_{i \in I_k} pi} \sum_{i \in I_k} pixi$ le barycentre de la classe C_k,

L'algorithme CAH

L'algorithme CAH est un algorithme itératif prenant en entrée un ensemble d'individu Ω défini plus haut et produisant une hiérarchie H. Grâce à cette hiérarchie nous pouvons choisir un nombre $K \in [\ 1\ ,\ \ldots,\ n\ ]$ de classes suivant lequel nous désirons partitionner les données. Toute la difficulté de la classification réside dans le choix de K et peut être obtenu grâce à différents critères que nous exposerons par la suite.

Etant donné la liberté du choix du nombre de classes que propose l'algorithme CAH il est raisonable de penser qu'il aura une complexité plus grande qu'un algrotihme de type K-means où K est choisi à l'avance.

Description de l'algorithme

1. Initialisation : $H_0 = \{\{x_i\}, i \in [1,n]\}$, pour i=1, ..., n.
2. Pour tout $k = 0, \ldots, n-1$, $k \to k+1$:

Calcul de la classe $C_k* = C_{k0} \cup C_{k1}$ où $(C_{k0},\ C_{k1}) = \arg \min_{i \neq j} \{\ dissim\ (C_i,\ C_j)$, C_i et C_j n'ont jamais été fusionné $\}$. $H_{k+1} = C_k* \cup H_k$.

On obtient ainsi en sortie la hiérarchie $H := H_n$.

Notons qu'à chaque étape, on crée une nouvelle classe en fusionnant les deux classes les plus proches, ainsi si on avait m classes à l'étape k de l'algorithme on a alors $m-1$ classes restantes à l'étape $k+1$ étant donné que les deux classes qui ont fusionné ne peuvent plus être choisies pour construire une nouvelle classe. Ainsi en répétant le procédé n fois, on obtient à la fin la classe formée de tous les éléments initiaux et l'algorithme est terminé.

Complexité

A chaque étape, le coût de calcul des dissimilarités entre chaque classe est de l'ordre de $\frac{n(n-1)}{2}$.

On répète l'opération n fois, l'algorithme est ainsi en temps $O(n^3)$.

L'algorithme Kernel CAH

Definition du Noyau

Soit E l'espace des individus.

Un noyau (kernel) est une fonction $k : E^2 \to \mathbb{R}$, $(x, y) \to k(x, y)$, telle que :

- k est symétrique i.e pour tout $(x, y) \in E^2$, $k(x, y) = k(y, x)$,
- k est défini positif i.e, Soit $n > 2$, pour toute suite d'élèments $(x_i, y_i)_{i=1,\dots,n} \in E^{2n}$, $(c_i)_{i=1,\dots,2n} \in \mathbb{R}^{\not\propto}$: $\sum_{i=1}^{n} \sum_{j=1}^{n} c_i \; c_j \; k(x_i, y_i) > 0$.

La deuxième propriété est similaire à la définition de matrice définie positive si on considère que E est un ensemble fini de taille n et V la matrice $V_{i,j} = k(x_i, y_i)$.

Theoreme de Mercer (Admis)

Soit $k : E^2 \to \mathbb{R}$ un noyau tel que $\int_{E^2} k(x, y)\, dxdy < \infty$, $T_k : L^2 \to L^2$, $f \to \int_E k(x, y) f(y)\, dy$ et considerons $(\lambda_i, e_i)_{i=1,\dots,n}$ l'ensemble des valeurs propres et fonctions propres de l'opérateur T_k.

Alors on a,

$$(M) \quad \forall (x, y) \in E^2, k(x, y) = \sum_{i=1}^{\infty} \lambda_i e_i(x) e_i(y)$$

References (art.3)

Caractérisation du noyau

Posons $\phi : E \to l_2$, $x \to (\sqrt{\lambda_i} e_i(x))_{i=1,\dots,n}$, où $l_2 = \{u_n : \mathbb{N} \to \mathbb{R}, \sum_n u_n < \infty\}$.

L'application ϕ prends en entrée un individu et renvoie une suite de l_2.

Alors on a,

$$\begin{aligned} (E) \quad \forall (x, y) \in E^2, \quad < \phi(x), \phi(y) > &= \sum_{i=1}^{\infty} \lambda_i e_i(x) e_i(y) \\ &= k(x, y) \end{aligned}$$

où $<, >$ est le produit scalaire dans l_2 et la deuxième égalité vient du Theorème de Mercer (M).

Ainsi on peut représenter deux individus avec la fonction auxiliaire ϕ dans l'espace de Hilbert l_2.

References (art.3)

Application aux calculs des dissimilarités

Pour mesurer la distance entre deux éléments x et y de E, on peut utiliser deux distances :

- $(*)$ $d^2(x,y) = ||x-y||^2 = < x-y, x-y >$), la distance usuelle de $(E, <, >)$,
- $(**)$ $d^2_\phi(x,y) := ||\phi(x) - \phi(y)||^2 = < \phi(x) - \phi(y), \phi(x) - \phi(y) >$), la distance issue de $(l_2, <, >)$ et de la fonction de représentation ϕ.

En combinant $(**)$ et (E) on obtient :

- $d^2_\phi(x,y) = < \phi(x), \phi(x) >$ - 2 $< \phi(x), \phi(y) >$) + $< \phi(y), \phi(y) > = k(x,x)$ - $2k(x,y) + k(y,y)$

En appliquant cette distance à la distance de Ward on obtient,

$$
\begin{aligned}
(1)\quad dissim(C_1, C_2) &= \frac{n_1 n_2}{n_1 + n_2} ||g_1 - g_2||^2 \\
&= \frac{n_1 n_2}{n_1 + n_2} || \frac{1}{n_1} \sum_{i \in I_1} \phi(x_i) - \frac{1}{n_2} \sum_{i \in I_2} \phi(x_i) ||^2 \\
&= \frac{n_1 n_2}{n_1 + n_2} (\frac{1}{n_1^2} \sum_{i,j \in I_1} < \phi(x_i), \phi(x_j) > -2 \frac{1}{n_1 n_2} \sum_{i \in I_1, j \in I_2} < \phi(x_i), \phi(x_j) > + \frac{1}{n_2^2} \sum_{i,j \in I_2} < \phi(x_i), \phi(x_j) >) \\
&= \frac{n_1 n_2}{n_1 + n_2} (\frac{1}{n_1^2} \sum_{i,j \in I_1} k(x_i, x_j) - 2 \frac{1}{n_1 n_2} \sum_{i \in I_1, j \in I_2} k(x_i, x_j) + \frac{1}{n_2^2} \sum_{i,j \in I_2} k(x_i, x_j))
\end{aligned}
$$

References (art.1) 2.2. Kernel HAC

Description de l'algorithme Kernel CAH

L'algorithme du Kernel CAH est similaire à l'algorithme classique CAH outre le fait que la distance de dissimilarité change. On peut utiliser la dissimilarité (1) ou bien tout autre dissimilarité mais la distance entre deux éléments de E s'opère par le kernel et non plus par la distance usuelle de E.

Pour se faire on peut par exemple utiliser un kernel gaussien, $k(x,y) = e^{\frac{-||x-y||^2}{2\sigma^2}}$,

où σ est un paramètre à estimer et $||.||$ représente la norme de E.

Choix du nombre de classes

Le problème (P) est un problème d'agencement de classes à K fixé qui depend uniquement de la distance de dissimilarité choisie.

Un autre problème est le choix du nombre de classes K. L'algorithme CAH fournit n classifications en K classes, K variant de 1 à n. L'idée est de choisir K répondant tel qu'il minimise la variance entre les classes, c'est à dire un nombre de classes qui rend les individus d'une même classes aussi proche que possible tout en ayant des classes qui soient le plus éloigné que possible. En d'autre terme on maximise l'inertie entre les classes et minimise l'inertie dans chaque classe.

Introduisons les définitions suivantes :

Definition

Soit $d^2(x, y)$ la distance usuelle entre deux éléments x et y de E et $k \in [1, n]$ un nombre de classes pour lequel les données sont partitionnées.

On définit les deu quantités suivantes :

- $D_r := \sum_{(x,y)\in C_r^2} d^2(x, y)$,
- $W_k := \sum_{r=1}^{k} \frac{1}{2n_r} D_r$.

D_r représente une mesure de la dispersion de la classe r, elle est d'autant plus grande que les individus de la classe sont éloignés entre eux. Il faut évidemment diviser par le nombre d'individus dans la classe si on veut pouvoir comparer des classes entre elles et par le facteur 0.5 car la distance est simétrique.

W_k mesure ici la dispersion totale des classes dans lesquelles les données sont réparties. Ainsi nous souhaitons un W_k assez petit tout en respectant la distance entre les classes. En effet, si l'on se concentre uniquement sur W_k, le partitionnement consistant à choisir n classes où chaque classe comprend une donnée nous donnerai un W_k nul mais la distance entre ces classes serait très petite. Nous devons donc faire un compromis.

References (art.1) 3.1 Gap Statistics

Critère du saut statistique

L'idée exposée dans l'article est d'introduire un autre jeu de donnée dans lequel on aurait une seule classe et de comparer sa classification avec celle de notre jeu de données initial. Supposons que les n données initiales $(x_i)_{i=1,..,n}$ soient des éléments de $\mathbb{R}_{\text{ı}}$ où $p \geq 1$.

Le nouveau jeu de données est obtenu de la façon suivante :

Simuler n réalisations indépendantes $(X_i)_{i=1,...,n}$ de loi uniforme :

$$X_i \sim U(\prod_{j=1}^{p} [a_j, b_j])$$

où,

- $a_j = min\{x_{i,j}, i \in [1, n]\}$,
- $b_j = max\{x_{i,j}, i \in [1, n]\}$.

On pose la variable aléatoire $Z := (X_1, \ldots, X_n)$ où les X_i sont iid. Ainsi notre jeu de donnée est composé de tirage indépendant suivant une loi uniforme. Avec une réalisation Z_m de Z, nous pouvons partitionner Z_j le tableau de donnée et calculer les W_k correspondant comme vu plus haut.

Appelons $W_{k,m}$ les W_k calculés relativement au tableau Z_m.

Soit k un nombre de classes de partitionnement des données.

L'idée est ici d'obtenir non pas les W_k pour une réalisation mais d'estimer leur valeurs moyennes par méthode de Monte Carlo. C'est à dire d'approcher la quantité $E(log(W_k)) \approx \frac{1}{M} \sum_{m=1}^{M} log(W_{k,m}) := \hat{\mu}$.

References (art.5) section 4

Le saut statistique $Gap : [1, n] \to \mathbb{R}$ est fonction du nombre de cluster des données et est définie de la façon suivante :

$$\forall k \in [1,n], \quad Gap(k) := \frac{1}{M}\sum_{m=1}^{M} log(W_{k,m}) - log(W_k)$$

A k fixé, cette fonction mesure ainsi l'écart de la dispersion (i.e les W_k) des k classes entre un jeu de données où toutes les données proviennent de la même variable aléatoire (c'est à dire un unique cluster) et le jeu données initial. Ainsi selon l'article le nombre de cluster optimal k^* est donné par :

$$k^* := \max_{k=1,..,n} Gap(k)$$

C'est à dire le k tel que le partitionnement des données soit le plus éloigné d'un paritionnement avec un seul cluster, en effet une faible quantité $Gap(k)$ indique qu'on a classer les données comme s'il y avait qu'une seule classe.

Critère du saut statistique ajusté

D'après (Reference (art.1)), un autre critère pour estimer le nombre de cluster peut être trouver à l'aide de l'écart type des $log(W_k)$.

On définit la fonction :

$$\forall k \in [1,n-1], \quad Gap_{adj}(k) := Gap(k) - Gap(k+1) + s_{k+1}$$

où $s_k = (1+\frac{1}{B})\sqrt{\frac{1}{M}\sum_{m=1}^{M}(log(W_{k,m}) - \hat{\mu})^2)}$.

La critère pour k^* est défini comme étant le premier k tel que $Gap_{adj}(k) > 0$.

Retour sur les distances de dissimilarité

Nous avons vu en "1.1.3 Dissimilarité" différentes distances de dissimilarité, c'est à dire des distances entre deux classes permettant de choisir les deux classes les plus proches et de les fusionner pour une obtenir une nouvelle classe dans l'algorithme CAH.

Bien d'autres distances peuvent être définies entre deux classes, on définit ici une famille de distances.

Distances de Lance-Williams

Soit d : C x $C \to \mathbb{R}_+$ une distance de dissimilarité sur un ensemble de classe C et d_E : $E^2 \to \mathbb{R}_+$ une distance entre les éléments de E.

On dit que d satisfait l'algorithme de **Lance-Williams** si d vérifie la relation de récurrence suivante :

$$\begin{cases} d(\{x_i\},\{x_j\}) = d_E(x_i,y_i), \quad \forall (x_i,y_i) \in E^2, \\ \\ d(C_i \cup C_j, C_k) = \alpha_i d(C_i,C_k) + \alpha_j d(C_j,C_k) + \beta d(C_i,C_j) + \gamma |d(C_i,C_k) - d(C_j,C_k)|, \forall\ (C_i,C_j,C_k) \in C^3 \end{cases} \quad (2)$$

La distance est d'abord définie pour les classes de singletons et est ensuite étendue à toutes les classes par la relation de récurrence (2). On note abusivement $d(x_i,x_j)$ la distance entre deux éléments de E. Cette distance n'est pas obligatoirement la distance euclidienne.

References (4)

Exemple :

Soit d la distance de **Lance-Wiliams** définie par $d_(x_i, x_j) = \frac{1}{2}||x_i - y_i||^2$, $\alpha_i = \frac{n_i+n_k}{n_i+n_j+n_k}$, $\alpha_j = \frac{n_j+n_k}{n_i+n_j+n_k}$, $\beta = -\frac{n_k}{n_i+n_j+n_k}$ et $\gamma = 0$.

Alors d est la distance de Ward i.e $d(C_i,C_j) = \frac{n_i n_j}{n_i+n_j}\ ||g_i - g_j||^2$, pour toute classe C_i et C_j.

La distance du cosinus

Proposition : Soit D la distance de **Lance-Williams** initialisée par $D(x,y) = ||\frac{x}{||x||} - \frac{y}{||y||}||^2$ avec des coefficients $\alpha_i, \alpha_j, \beta, \gamma$ tels que $\alpha_i + \alpha_j + \beta = 1$. Posons $S(x,y) := 1 - \frac{1}{2}D(x,y)$.

Alors S est une distance de **Lance-Williams**.

Preuve :

Pour $x, y \in E$ on a :

$$\begin{aligned} S(x,y) &= 1 - \frac{1}{2}D(x,y) \\ &= 1 - \frac{1}{2}||\frac{x}{||x||} - \frac{y}{||y||}||^2 \\ &= 1 - \frac{1}{2}(2 - 2 < \frac{x}{||x||}, \frac{y}{||y||} >) \\ &=< \frac{x}{||x||}, \frac{y}{||y||} > \end{aligned}$$

$S(x,y)$ est le cosinus de l'angle entre les vecteurs x et y. Notons que $S(x,x) = 1$ et que S n'est pas positive mais un ajustement peut remédier à cela.

Relation de récurrence de S :

Pour $(C_i, C_j, C_k) \in C^3$

$$\begin{aligned} S(C_i \cup C_j, C_k) &= -\frac{1}{2}D(C_i \cup C_j, C_k) + 1 \\ &= -\frac{1}{2}(\alpha_i D(C_i, C_k) + \alpha_j D(C_j, C_k) + \beta D(C_i, C_j) + \gamma |D(C_i, C_k) - D(C_j, C_k)|) + 1 \\ &= -\frac{1}{2}(\alpha_i(2 - 2S(C_i, C_k)) + \alpha_j(2 - 2S(C_j, C_k)) + \beta(2 - 2S(C_i, C_j)) + 2\gamma|S(C_j, C_k) - S(C_i, C_k)|)) + 1 \\ &= \alpha_i(S(C_i, C_k) - 1) + \alpha_j(S(C_j, C_k) - 1) + \beta(S(C_i, C_j) - 1) + \gamma|S(C_j, C_k) - S(C_i, C_k)|)) + 1 \\ &= \alpha_i S(C_i, C_k) + \alpha_j S(C_j, C_k) + \beta S(C_i, C_j) + \gamma|S(C_j, C_k) - S(C_i, C_k)| - (\alpha_i + \alpha_j + \beta) + 1 \\ &= \alpha_i S(C_i, C_k) + \alpha_j S(C_j, C_k) + \beta S(C_i, C_j) + \gamma|S(C_j, C_k) - S(C_i, C_k)| \end{aligned}$$

S est donc bien une distance de **Lance-Williams**.

Pour remédier au problème de la non positivité de S, l'auteur suggère de remplacer $S(x,y)$ par $\frac{S(x,y) + \max\limits_{x,y \in D} S(x,y)}{\max\limits_{x,y \in D} S(x,y) + 1}$, où D est le jeu de donnée.

Avantages de la distance du cosinus

Dans l'article (art.2), on propose de ne pas conserver les classes pour lesquelles la distance S serait inférieur à un seuil $s \in [0,1]$ donné. Ce qui permet de réduire le coût de stockage de la matrice de dissimilarité. En

effet pour l'algorithme classique, le coût de stockage est en $O(n^2)$, et avec ce critère il devient en $O(nm)$, où $m = card(\{S(Ci, Cj) > s, i \neq j\})$, le nombre de paire de classe telles que leur distance est supérieur au seuil s.

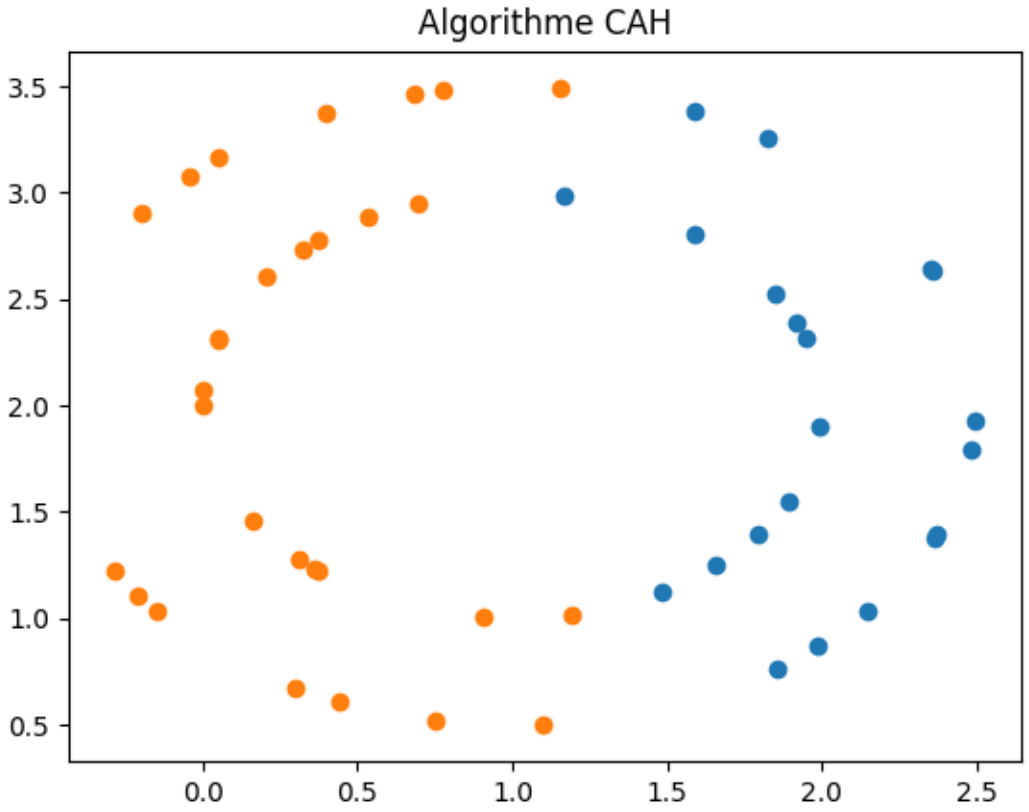
Notons bien que la compléxité de l'algorithme reste inchangée mais c'est la capacité de stockage qui est réduite, cela est utile lorsqu'on traite un large jeu de données, on peut ainsi choisir un seuil adapté à notre capacité de stockage.

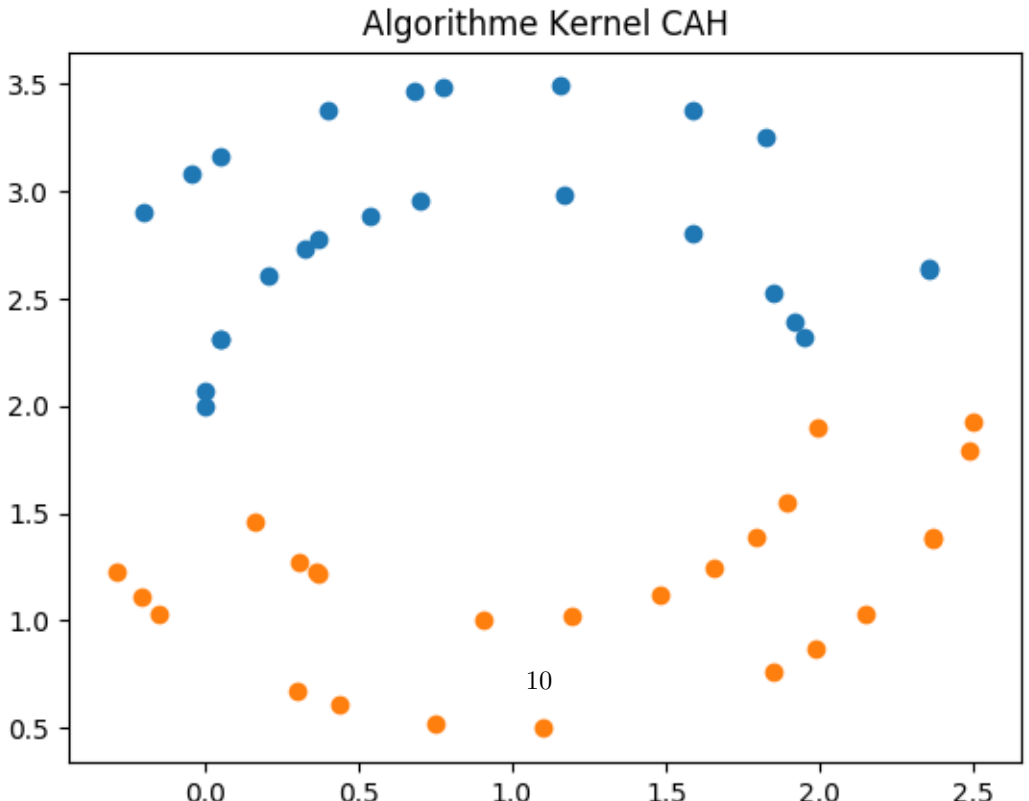
L'article montre aussi que le seuil ne diminue par toujours les performances de classification.

References (art.2)

Implémentations

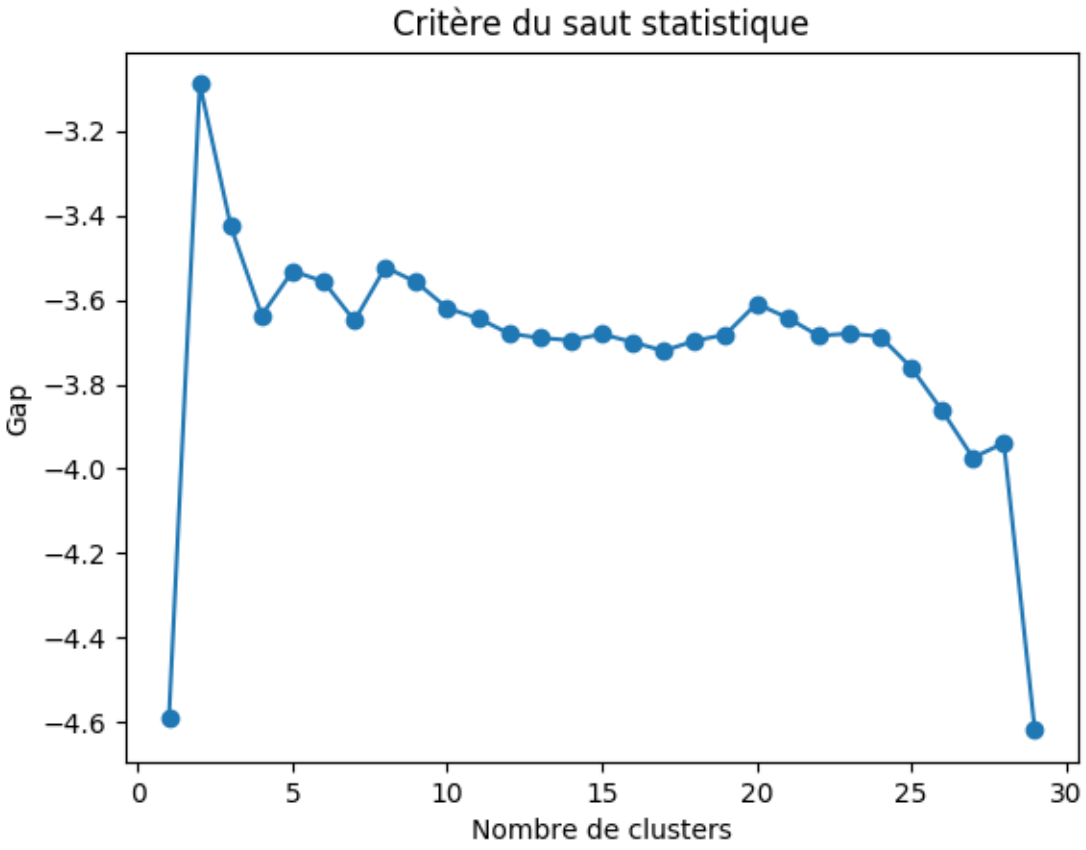
CAH vs Kernel CAH

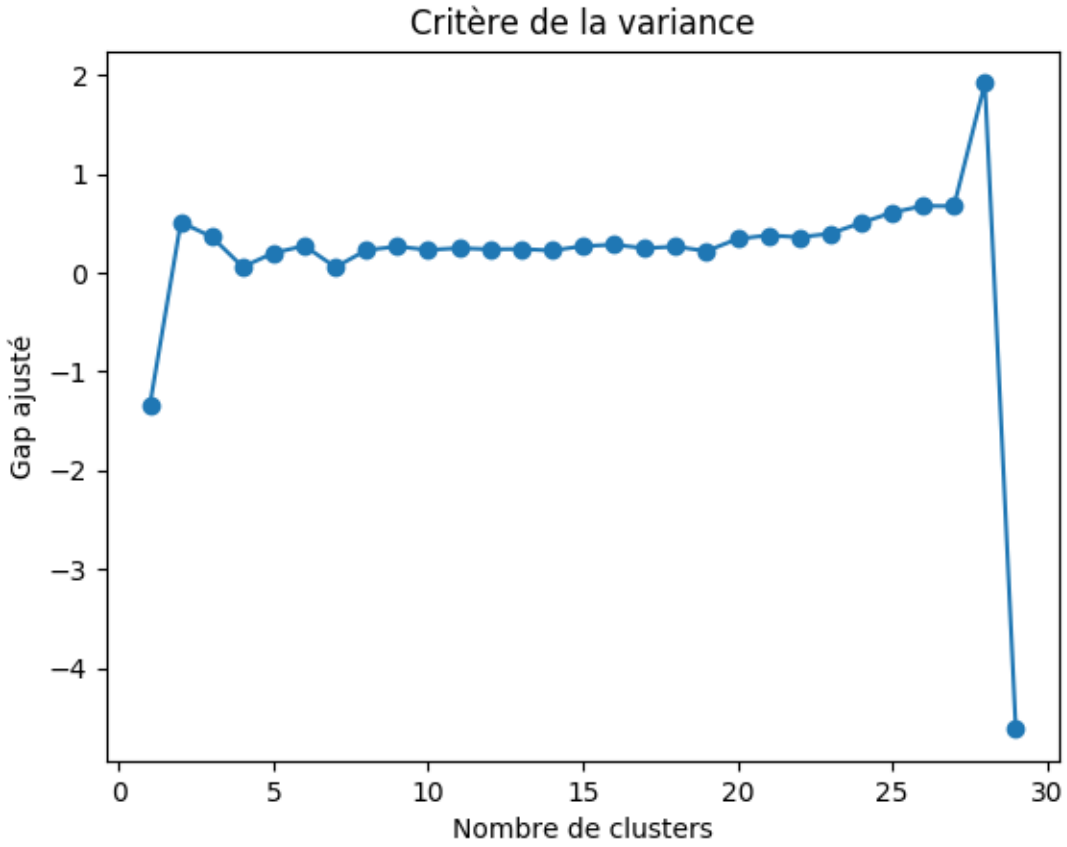




Critère du choix de classes

On a réalisé ici une classification à l'aide du Kernel CAH au jeux de données suivant. 25 points de $\mathbb{R}^{\not\vDash}$ suivant une loi normale $N((0,0),I)$ et 25 points de $\mathbb{R}^{\not\vDash}$ suivant une loi normale $N((5,5),I)$. L'estimation Monte-Carlo a été obtenue avec 150 simulations.





Références

- (art.1) https://www.researchgate.net/publication/288661963_Kernel_hierarchical_agglomerative_clustering_Comparison_of_different_gap_statistics_to_estimate_the_number_of_clusters
- (art.2) https://www.researchgate.net/publication/313067283_Similarity_Based_Hierarchical_Clustering_with_an_Application_to_Text_Collections
- (art.3) https://homes.cs.washington.edu/~thickstn/docs/mercer.pdf
- (4) https://en.wikipedia.org/wiki/Ward%27s_method
- (art.5) https://statweb.stanford.edu/~gwalther/gap#:~:text=We%20propose%20a%20method%20(the,an%20appropria