COURS DE DEUXIÈME ANNÉE
INGÉNIEUR ISIMA

Machine-Learning
Classification et clustering

Vincent Barra

Table des matières

Buffon écrit en 1749 « Le seul moyen de faire une méthode instructive et naturelle, est de mettre ensemble des choses qui se ressemblent et de séparer celles qui diffèrent les unes des autres ». La classification est un des thèmes majeur, d'abord de l'histoire naturelle, puis de la biologie, et maintenant qui touche tous les domaines scientifiques. Nourries de nombreuses branches de mathématiques et de l'informatique, de nombreuses méthodes sont apparues, notamment depuis l'apparition de la théorie statistique au XIX^e siècle.
La classification peut être abordée à partir d'un ensemble d'apprentissage, qui est utilisé pour caractériser les classes à rechercher, ou sans exemple d'apprentissage (clustering). Le présent chapitre se propose d'aborder ces deux aspects, en détaillant plus l'aspect clustering (apprentissage non supervisé), l'approche supervisée étant elle abordée tout au long du cours.

1 Classification supervisée

Dans les méthodes de classification supervisée, on suppose disposer d'un ensemble d'apprentissage $Z = \{(x_i, y_i), 1 \leq i \leq n, x_i \in X, y_i \in Y\}$. X peut être un ensemble de caractéristiques numériques, logiques, symboliques... L'objectif est alors de classer des individus x dont la classe est inconnue, en utilisant un algorithme construit sur Z.

Ce problème est central au cours de Machine Learning en général, et les autres chapitres abordent largement sa résolution par diverses méthodes (SVM, réseaux de neurones, arbres de décision,...). Nous décrivons ici deux algorithmes simples, permettant de faire le pendant de l'approche de classification non supervisée (clustering), plus largement détaillée dans la section 2.

1.1 Classifieur naïf de Bayes

Le classifieur naïf de Bayes est une méthode de classification supervisée qui repose sur une hypothèse simplificatrice forte : les descripteurs (coordonnées sur X) sont deux à deux indépendants conditionnellement à la classe à prédire dans Y.
Soit x un individu à classer. La règle bayésienne d'affectation optimale consiste à maximiser la probabilité a posteriori d'appartenance aux classes, i.e.

$$(x \text{ est dans la classe } k \in Y) \Leftrightarrow (k = arg\max_l P(y = l|x))$$

La décision repose donc sur une estimation de la probabilité conditionnelle $P(Y|x)$, qui peut s'écrire par la règle de Bayes

$$(\forall k \in Y) \quad P(y = l|x) = \frac{P(x|y = l)P(y = l)}{P(x)}$$

La maximisation ne dépendant pas de l, la maximisation s'écrit alors

$$(x \text{ est dans la classe } k \in Y) \Leftrightarrow (k = arg\max_l P(x|y = l)P(y = l))$$

La quantité $P(y = l)$ est facile à estimer à l'aide de Z par approche fréquentiste. En pratique, pour prendre en compte les petits effectifs, on calcule cette quantité selon

$$P(y = l) = p_l = \frac{n_l + m}{n + m|Y|}$$

où n_l est l'effectif de la classe l dans Z et m un paramètre.
L'estimation de $P(x|y = l)$ est plus délicate et sous-tend l'introduction d'hypothèses. Par exemple, l'analyse discriminante paramétrique stipule que la distribution est gaussienne, tandis que la régression logistique binaire suppose que le rapport $P(x|y = 1)/P(x|y = -1)$ suit une loi particulière. Dans le cas du classifieur naïf de Bayes, on suppose que les descripteurs sont deux à deux indépendants conditionnellement aux valeurs de la variable de classe, et ainsi

$$P(x|y = l) = \prod_{j=1}^{d} P\left(f_j = x_j|y = l\right)$$

où d est le nombre de descripteurs (la dimension de X), et f_j est le j^e descripteur. Le nombre de paramètres à estimer est alors très réduit, et pour une variable quelconque pouvant prendre Q valeurs, on utilise

$$(\forall q) \quad P\left(f_j = q|y = l\right) = \frac{n_{lq} + m}{n_l + mQ}$$

Le classifieur de Bayes opère généralement sur le logarithme des probabilités (ceci d'autant plus que d est grand) et la règle du classifieur naïf de Bayes est finalement :

$$(x \text{ est dans la classe } k \in Y) \Leftrightarrow \left(k = arg\max_{l} \left[log(P(y=l) + \sum_{j=1}^{d} logP\left(f_j = x_j | y = l\right) \right] \right)$$

Le classifieur naïf de Bayes est un classifieur linéaire.

1.2 Méthode des k plus proches voisins

Etant donnée une métrique δ sur X, la méthode des k plus proches voisins (k-PPV ou K-NN) détermine pour $x \in X$ les k points $x_1 \cdots x_k$ de Z les plus proches de x au sens de δ. La règle de décision consiste à affecter x à la classe majoritairement représentée dans les appartenances des x_i.
On peut rejeter le résultat de la classification selon deux règles classiques :

- — rejet en ambiguïté si la classe choisie n'est pas représentée par k' voisins parmi les k
- — rejet en distance si les plus proches voisins sont à une distance de x supérieure à une distance minimum

On peut montrer que la probabilité d'erreur P_1 dans la méthode du 1-PPV ($k = 1$) est reliée à la probabilité d'erreur bayésienne P (errerur minimale) par l'inégalité suivante, lorsque n devient grand :

$$P \leq P_1 \leq P\left(2 - \frac{|Y|}{|Y|-1}P\right)$$

2 Classification non supervisée

Les méthodes de classification non supervisée (clustering en anglais) sont par définition des algorithmes qui ne nécessitent pas d'apprentissage. Ces méthodes ont pour objectif de grouper un ensemble d'objets en classes, de sorte que des objets similaires se retrouvent dans les mêmes groupes, et des objets dissimilaires dans des groupes différents. Cette définition reste très imprécise, voire ambiguë puisque les deux objectifs peuvent être contradictoires. En effet, la similarité n'est pas une relation transitive, alors que l'appartenance à une même classe est une relation d'équivalence et, de ce fait, transitive. Ainsi, si $x_1 \cdots x_n$ est un ensemble d'objets tel que pour tout i x_i est très similaire à ses deux voisins x_{i-1} et x_{i+1}, il se peut également que x_1 et x_n soient très dissemblants. Regrouper les objets semblables entre eux amène à ne créer qu'une seule classe, mais dans ce cas les points extrêmes violent le deuxième objectif fixé.
De même, l'absence de "vérité" (contrairement à l'apprentissage supervisé, ou un ensemble d'apprentissage est disponible) rend l'évaluation de ces méthodes parfois difficile.

2.1 Description de quelques méthodes

D'un point de vue formel, il s'agit, à partir d'un ensemble $X = \{x_i, 1 \leq i \leq n, x_i \in \mathbb{R}^d\}$ d'objets, de former une partition de cet ensemble, i.e. un ensemble $P = (P_1, P_2, \cdots P_g)$ de parties non vides de X tel que :

1. $(\forall k \neq l) P_k \cap P_l = \emptyset$
2. $\cup_{i=1}^{g} P_i = X$

Dans un ensemble X partitionné en g classes, chaque élément de l'ensemble appartient à une classe et une seule. Une manière pratique de décrire cette partition P consiste à lui associer la matrice de classification $C = (c_{ij})_{1 \leq i \leq n, 1 \leq j \leq g}$, où $c_{ij} = 1$ si l'individu i appartient à P_j, et $c_{ij} = 0$ sinon. Dans le cas où l'on accepte qu'un individu appartienne à plusieurs classes (avec des degrés d'appartenance), on autorise c_{ij} à couvrir l'intervalle [0,1] et on parle alors de classification floue.

2.1.1 Clustering par minimisation de fonction

Fonctions de coût Les méthodes de classification non supervisée par minimisation de fonction définissent un coût sur l'ensemble paramétré de toutes les partitions possibles, et optimisent le coût pour trouver la "meilleure" partition possible. Très souvent, la fonction objectif nécessite de connaître a priori le nombre de classes g.
On pourrait penser qu'il suffit d'énumérer l'ensemble des partitions possibles de X et de ne retenir que la meilleure, mais la combinatoire rend la démarche impossible. Le nombre de partitions en g classes d'un ensemble à n éléments, que l'on note S_n^g est le nombre de Stirling de deuxième espèce. En posant $S_0^0 = 1$ et pour tout $n > 0$, $S_n^0 = S_0^n = 0$, il peut être calculé par récurrence grâce à la relation $S_n^g = S_{n-1}^{g-1} + gS_{n-1}^g$.
On peut montrer que

$$S_n^g = \frac{1}{g!} \sum_{i=1}^{g} C_g^i (-1)^{g-i} i^n$$

et donc $S_n^g \sim \frac{g^n}{g!}$ lorsque $n \to \infty$. En pratique, sur un ordinateur calculant 10^6 partitions par seconde, il faut 126 000 ans pour calculer l'ensemble des partitions d'un ensemble à $n = 25$ éléments.

Parmi les fonctions de coût les les plus utilisées dans les méthodes de clustering par optimisation, on note :

— *la fonction objectif des k-means* : dans l'algorithme correspondant, chaque classe P_i est représentée par un centroïde μ_i. On suppose donc que X est plongé dans un espace métrique (X', d) avec $\mu_i \in X'$. La fonction objectif des k-means mesure la distance au carré entre chaque $x_i \in X$ et le centroïde de sa classe. Etant donné que

$$\mu_i = arg \min_{\mu \in X'} \sum_{x \in P_i} d(x, \mu)^2$$

la fonction objectif est définie par

$$\begin{aligned} f_{k-means} &= \sum_{i=1}^{g} \sum_{x \in P_i} d(x,\mu_i)^2 \\ &= \min_{\mu_1 \cdots \mu_g \in X'} \sum_{i=1}^{g} \sum_{x \in P_i} d(x,\mu_i)^2 \end{aligned}$$

— *la fonction objectif des k-médioïdes* : très similaire à la précédente, cette fonction demande à ce que les centres de classe soient des éléments de X :

$$f_{k-medioides} = \min_{\mu_1 \cdots \mu_g \in X} \sum_{i=1}^{g} \sum_{x \in P_i} d(x,\mu_i)^2$$

— *la fonction objectif du k-médian* : ici encore, la formulation est la même, la distance n'étant plus élevée au carré :

$$f_{k-median} = \min_{\mu_1 \cdots \mu_g \in X} \sum_{i=1}^{g} \sum_{x \in P_i} d(x,\mu_i)$$

Ces fonctions recherchent des centres de classes μ_i, et affectent chaque point x_j au centre le plus proche. D'autres fonctions n'utilisent pas les centres de classe comme objectif, comme par exemple :

— *la somme des distances inter classes* $f_{SOD} = \sum_{i=1}^{g} \sum_{x,y \in P_i} d(x,y)$

— *le MinCut* $f_{cut} = \sum_{i=1}^{g} \sum_{x \in P_i, y \notin P_i} W_{x,y}$, où $W_{x,y}$ est une mesure de similarité entre x et y

Un exemple : l'algorithme des k-means La minimisation de $f_{k-means}$ est en pratique le plus souvent infaisable (problème NP-complet), et on lui préfère une alternative algorithmique (algorithme 1) [**?**].

Algorithme 1 : Algorithme des K-means

Entrées : X, g

Initialisation : tirage au hasard de g points $\mu_1 \cdots \mu_g$

tant que *non convergence* **faire**

$$\begin{aligned} (\forall 1 \leq i \leq g) \quad P_i &= \left\{ x \in X ; i = arg\min_j d(x,\mu_j)^2 \right\} \\ (\forall 1 \leq i \leq g) \quad \mu_i &= \frac{1}{|P_i|} \sum_{x \in P_i} x \end{aligned}$$

Il reste à prouver que cet algorithme minimise $f_{k-means}$. Rappelons tout d'abord que

$$f_{k-means} = \sum_{i=1}^{g} \sum_{x \in P_i} d(x,\mu_i)^2$$

et considérons l'étape t de mise à jour dans l'algorithme 1. On note $P_1^{(t-1)} \cdots P_g^{(t-1)}$ la partition obtenue à l'itération $t-1$ et $\mu_i^{(t-1)}$ les centres de classe correspondant. On note enfin $f_{k-means}^{(t)}$ la valeur de la fonction objectif à l'itération t.
En utilisant la définition de la fonction objectif on a :

$$f_{k-means}^{(t)} \leq \sum_{i=1}^{g} \sum_{x \in P_i^{(t)}} d(x, \mu_i^{(t-1)})^2$$

La définition de la nouvelle partition $P_1^{(t)} \cdots P_g^{(t)}$ implique qu'elle minimise $\sum_{i=1}^{g} \sum_{x \in P_i} d(x, \mu_i^{(t-1)})^2$ sur l'ensemble des partitions $(P_1 \cdots P_g)$ possibles. Ainsi :

$$\sum_{i=1}^{g} \sum_{x \in P_i^{(t)}} d(x, \mu_i^{(t-1)})^2 \leq \sum_{i=1}^{g} \sum_{x \in P_i^{(t-1)}} d(x, \mu_i^{(t-1)})^2$$

Le terme de droite est égal à $f_{k-means}^{(t-1)}$ et on a alors

$$f_{k-means}^{(t)} \leq f_{k-means}^{(t-1)}$$

$f_{k-means}$ est donc monotone décroissante, mais l'algorithme n'apporte aucune garantie sur le nombre d'itérations nécessaires pour assurer la convergence. De plus, pour un nombre d'itérations donné, la valeur retournée par l'algorithme peut être très loin du minimum théorique du problème, et, pire, l'algorithme peut converger vers un point qui n'est même pas un minimum local.
Pour améliorer les résultats de l'algorithme, il est conseillé de le répéter plusieurs fois avec des initialisations différentes.

2.1.2 Clustering spectral

Un problème de clustering peut être représenté par la résolution d'un problème sur un graphe [**?**]. On construit un graphe G où chaque sommet est un point $x_i \in X$, les noeuds x_i et x_j étant connectés par un arc valué par une mesure de similarité $W_{ij} = s(x_i, x_j)$ entre les points. Un choix classique pour s est :

$$s(x_i, x_j) = e^{-\frac{d(x_i, x_j)^2}{2\sigma^2}}$$

où d est une distance sur $\mathbb{R}^n$ et σ un paramètre (largeur de bande : visibilité des voisins d'un point donné). Il est également possible d'utiliser une distance type cosinus en posant $W = Y^T Y$, où $Y \in \mathcal{M}_{d,n}(\mathbb{R})$ est la matrice dont les colonnes sont les individus de X, et dont les lignes sont normalisées.
Le regroupement des points de X en classes peut alors se formuler comme suit : trouver une partition du graphe telle que les arcs reliant les différentes parties ont des poids faibles, les arcs dans les parties ayant des poids élevés.

Coupe Soit $G = (X, E)$ représenté par sa matrice de similarité $W \in \mathcal{M}_n(\mathbb{R})$.Partitionner G en g classes $P_1 \cdots P_g$ peut être effectué en résolvant le problème du mincut :

$$(P_1 \cdots P_g) = Arg \min_{C_1 \cdots C_g} \sum_{i=1}^{g} \sum_{r \in C_i, s \notin C_i} W_{rs}$$

Le plus souvent cependant, les partitions résultantes ne sont pas satisfaisantes : le problème du mincut isole parfois dans une classe isolée un noeud des autres points.
Une solution à ce problème est de normaliser la coupe :

$$(P_1 \cdots P_g) = Arg \min_{C_1 \cdots C_g} \sum_{i=1}^{g} \frac{1}{|C_i|} \sum_{r \in C_i, s \notin C_i} W_{rs}$$

ce qui rend le calcul de l'optimum beaucoup plus difficile à réaliser. Le clustering spectral apporte une solution efficace à la minimisation de la fonction précédente.
On note dans la suite $R(C_1 \cdots C_g) = \sum_{i=1}^{g} \frac{1}{|C_i|} \sum_{r \in C_i, s \notin C_i} W_{rs}$.

Laplacien du graphe Le laplacien non normalisé d'un graphe G est la matrice $L \in \mathcal{M}_n(\mathbb{R})$ définie par

$$L = D - W$$

où D est la matrice des degrés de G :

$$D = diag(d_{ii}) = diag\left(\sum_{j=1}^{n} W_{ij}\right)$$

Pour une partition $(C_1 \cdots C_g)$, on note $H \in \mathcal{M}_{n,g}(\mathbb{R})$ la matrice définie par :

$$(\forall 1 \leq i \leq n)(\forall 1 \leq j \leq g) \quad H_{ij} = \frac{1}{\sqrt{|C_i|}} \mathbf{1}_{\{i \in C_j\}}$$

Les colonnes h_i de H sont clairement orthonormées. De plus des opérations simples d'algèbre linéaire permettent d'écrire $Tr(H^T L H) = \sum_{i=1}^{g} h_i^T L h_i$ et pour tout $u \in \mathbb{R}^n$:

$$\begin{aligned} u^T L u &= \frac{1}{2}\left(\sum_{k=1}^{n} D_{kk} u_k^2 - 2\sum_{k,l=1}^{n} u_k u_l W_{kl} + \sum_{l=1}^{n} D_{ll} u_l^2\right) \\ &= \frac{1}{2}\sum_{k,l=1}^{n} W_{kl}(u_k - u_l)^2 \end{aligned}$$

Si $u = h_i$, et remarquant que $((h_i)_k - (h_i)_l)^2$ est non nul si et seulement si $k \in C_i$ et $l \notin C_i$, on obtient

$$h_i^T L h_i = \frac{1}{|C_i|} \sum_{r \in C_i, s \notin C_i} W_{rs}$$

et donc

$$Tr(H^T LH) = \sum_{i=1}^{g} \frac{1}{|C_i|} \sum_{r \in C_i, s \notin C_i} W_{rs} = R(C_1 \cdots C_g)$$

Ainsi, pour résoudre le problème de mincut non normalisé, on cherche H de colonnes orthonormées telle que $H_{ij} = 0$ ou $H_{ij} = \frac{1}{\sqrt{|C_i|}}$. En pratique, pour des raisons d'efficacité, on relaxe cette dernière contrainte et on cherche H qui minimise $Tr(H^T LH)$. De la même manière que dans le cas de l'ACP, la solution (algorithme 2) consiste à rechercher les g plus petits vecteurs propres de L.

Algorithme 2 : Algorithme de clustering spectral non normalisé

Entrées : $W \in \mathcal{M}_n(\mathbb{R})$, g
$L = D - W$
$U \in \mathcal{M}_{n,g}(\mathbb{R}) \leftarrow$ matrice des g plus petits vecteurs propres de L
$U_{i,.}, 1 \leq i \leq n$: lignes de U
$(P_1 \cdots P_g) =$Kmeans$(U_{1,.} \cdots U_{n,.}, g)$

Plusieurs versions normalisées de cet algorithme existent, la plus simple d'entre elles [**?**] consistant dans l'algorithme précédent à rechercher les g premiers vecteurs propres du problème généralisé $Lu = \lambda Du$.

2.1.3 Estimation des densités des classes

Les méthodes basées sur l'estimation des densités trouvent dans $\mathbb{R}^d$ les régions de haute densité qui sont séparées des autres par des régions de faible densité.

DBSCAN DBScan [**?**], comme d'autres méthodes, estime la densité pour un point x particulier en comptant le nombre de points de X (x compris) présents dans un voisinage de rayon ε donné. Si la méthode est simple, elle dépend fortement de ε.
Les points de X sont alors classés dans trois catégories :

- points intérieurs : un point x est intérieur si le nombre de voisins de x dans le voisinage défini par ε est supérieur à un seuil donné s.
- points de bords : x est un point de bord s'il n'est pas intérieur mais est dans le voisinage d'un (ou de plusieurs) points intérieur(s)
- points de bruit : x est un point de bruit s'il n'est ni intérieur ni de bord.

DBSCAN est alors décrit par l'algorithme 3.
L'algorithme est très dépendant de ses deux paramètres, ε et s :

- pour choisir ε, on peut calculer la distance de tous les points de X à leur k^e plus proche voisin (pour k fixé), les trier par ordre croissant, et retenir pour valeur de ε le saut max des distances. En effet, pour des points appartenant à la même classe, cette distance sera en moyenne petite pourvu que k ne dépasse pas le cardinal de la classe. En revanche, cette distance sera grande pour des points hors classe.
- une fois ε choisi de cette manière, $s = k$ permet d'étiqueter comme points intérieurs les points dont la distance à leur k^e plus proche voisin est inférieure à ε .

Algorithme 3 : Algorithme DBSCAN

Entrées : X, ε, s

1. Étiqueter les éléments de X en points intérieurs, de bord ou de bruit
2. Éliminer les points de bruit
3. Relier les points intérieurs qui sont à une distance de moins de ε l'un de l'autre
4. Grouper les points intérieurs connectés dans des classes
5. Affecter chaque point de bord à l'une des classes contenant ses points intérieurs

Il est également possible de tracer un graphique de validité de la partition en fonction de ε, et de retenir la valeur maximisant cette validité (voir par exemple la figure 1, où la mesure de validité choisie est l'indice de silhouette (paragraphe 2.2.1)).

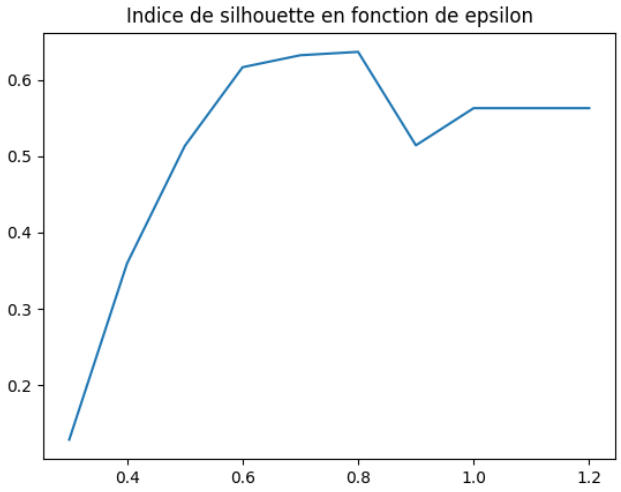

Fig. 1: Détermination de ε par maximisation de l'indice de silhouette

MeanShift L'algorithme MeanShift [**?**] est une technique de classification non supervisée et non paramétrique, qui ne nécessite aucun a priori sur le nombre de classes, et ne fait pas d'hypothèse sur la forme des nuages de points dans l'espace des paramètres.
Etant donnés les n points de données $x_i, 1 \leq i \leq n$, l'estimation de la densité multivariée obtenue avec un noyau K de bande passante h est donnée par

$$f(x) = \frac{1}{h^d} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right)$$

Pour des noyaux symétriques, il suffit de définir K en fonction d'un profil k tel que $K(x) = c_{k,d} k(\|x\|^2)$, où $c_{k,d}$ est une constante de normalisation assurant que le noyau K est de somme 1 sur le domaine de x. Les modes de la fonction de densité sont les zéros du

gradient de f, qui s'écrit, en posant $g = -k'$:

$$\begin{aligned}\nabla f(x) &= \frac{2c_{k,d}}{nh^{d+2}} \sum_{i=1}^{n} (x_i - x) g\left(\left\|\frac{x-x_i}{h}\right\|^2\right) \\ &= \frac{2c_{k,d}}{nh^{d+2}} \left[\sum_{i=1}^{n} g\left(\left\|\frac{x-x_i}{h}\right\|^2\right)\right] \left[\frac{\sum_{i=1}^{n} x_i g\left(\left\|\frac{x-x_i}{h}\right\|^2\right)}{\sum_{i=1}^{n} g\left(\left\|\frac{x-x_i}{h}\right\|^2\right)} - x\right]\end{aligned}$$

Le premier terme entre crochets est proportionnel à la densité estimée en x calculée avec le noyau $G(x) = c_{g,d} g\left(\|x\|^2\right)$, et le second terme, noté $m_h(x)$ est appelé mean shift, et est un vecteur qui pointe dans la direction d'une augmentation maximale de la densité .
L'algorithme meanshift est une recherche des modes de densité des données : Partant des points initiaux, l'application itérative

1. du calcul de $m_h(x^t)$
2. de la translation $x^{t+1} = x^t + m_h(x^t)$

permet de trouver les points stationnaires de la fonction de densité. Un filtrage des maxima locaux permet de retenir les modes pertinents, qui sont représentatifs des classes présentes dans les données initiales. L'ensemble des points qui converge vers le même mode est appelé bassin d'attraction de ce mode. Les points qui sont dans le même bassin sont associés à la même classe.
La figure 2 présente une application de l'algorithme sur des données de synthèse en 3D.

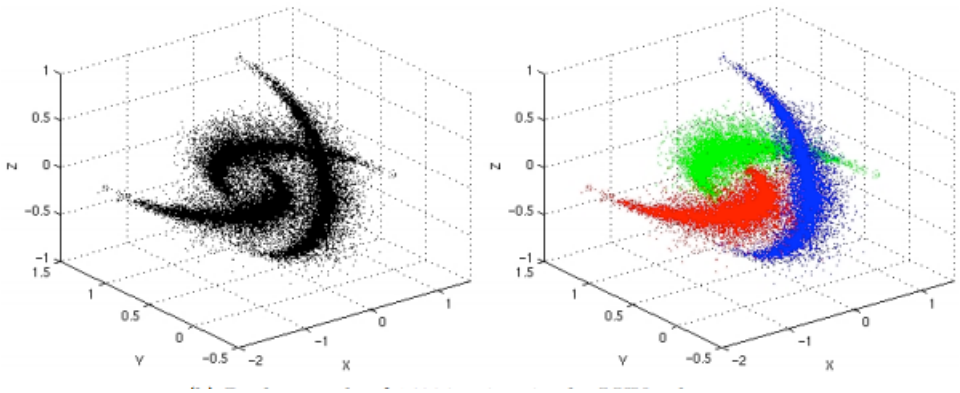

Fig. 2: Exemple de l'application de l'algorithme Meanshift sur des données 3D

2.1.4 Clustering hiérarchique

Principe Les algorithmes de cette famille procèdent de manière itérative. Partant de la partition classique où chaque partie est un singleton $\{x_i\}$, les méthodes agrègent les parties les "plus proches " de l'itération précédente, faisant diminuer la taille de la partition jusqu'à la solution triviale $P_1 = X$.

Plus qu'une partition, ces algorithmes construisent une hiérarchie indicée, c'est à dire un ensemble H d'ensembles tels que :

1. $X \in H$
2. $(\forall x \in X)\{x\} \in H$
3. $(\forall h, h' \in H) h \cap h' = \emptyset$ ou $h \subset h'$ ou $h' \subset h$

Une hiérarchie est souvent représentée par l'intermédiaire d'un indice, qui est une fonction i de H dans $\mathbb{R}^+$, strictement croissante vis à vis de l'inclusion et de noyau l'ensemble des singletons de X.
Si $P = (P_1 \cdots, P_g)$ est une partition de X, l'ensemble H formé des classes P_k de P, des singletons de X et de l'ensemble X lui-même forme une hiérarchie. Inversement, il est possible d'associer à chaque niveau d'une hiérarchie indicée une partition. Une hiérarchie indicée correspond donc à un ensemble de partitions emboîtées.
Deux paramètres déterminent la nature de l'algorithme de clustering hiérarchique :

— la mesure de la distance entre parties, qui définit la notion d'agrégation
— le critère d'arrêt, qui donne la partition optimale.

Mesures de distance La mesure de la distance est une extension de la métrique d utilisée entre deux points de données x_i. Les distances les plus classiques sont :

— le lien simple : $D_{min}(P_i, P_j) = \min_{x \in P_i, y \in P_j} d(x,y)$
— le lien maximum : $D_{max}(P_i, P_j) = \max_{x \in P_i, y \in P_j} d(x,y)$
— le lien moyen : $D_{moy}(P_i, P_j) = \dfrac{\sum_{x \in P_i} \sum_{y \in P_j} d(x,y)}{|P_i||P_j|}$
— le lien de Ward : Lorsque X est mesuré par d variables quantitatives, il est possible de lui associer un nuage de points pondérés dans $\mathbb{R}^d$ muni de la distance euclidienne d. Généralement, les pondérations seront toutes égales à 1. Le critère d'agrégation le plus utilisé dans cette situation est alors le critère d'inertie de Ward :

$$D(P_i, P_j) = \frac{p_i p_i}{p_i + p_i} d^2(g(P_i), g(P_j))$$

où p_k représente la somme des pondérations des éléments d'une partie P_k et $g(P_k)$ est le centre de gravité de P_k.

A chaque itération, les parties les plus proches sont agrégées en une nouvelle partie, contenant la réunion des x_i des parties initiales.

Critère d'arrêt et partition L'ensemble des itérations peut être visualisé sous la forme d'un arbre, appelé dendrogramme (figure 3).

Le critère d'arrêt permet de déterminer la partition de X la plus appropriée. Ici encore, plusieurs choix sont possibles :

— en fixant a priori un nombre de classes

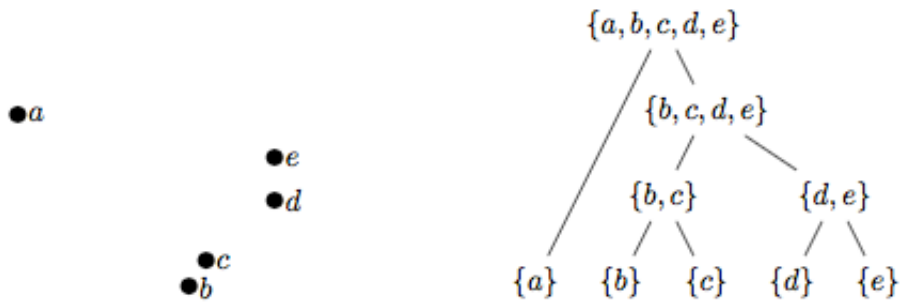

Fig. 3: Exemple de dendrogramme sur $X = \{a, b, c, d, e\}$

— en fixant une borne supérieure r pour D, et en stoppant les itérations dès que les distances calculées par les liens dépassent r. A noter que r peut être également calculé par $r = \alpha max\{d(x,y), x, \in X\}$ (critère dit "scale distance upper bound").
— en coupant le dendrogramme au saut de distance D maximal.

2.2 Evaluation des partitions

L'évaluation d'une méthode de classification non supervisée n'est pas chose aisée, contrairement à des méthodes supervisées où un ensemble de test et un ensemble de validation permettent de calculer et de caractériser numériquement une solution.
Plusieurs aspects rentrent en jeu lorsqu'il s'agit de caractériser une partition :

— Existe t il réellement une structure en nuage de points dans les données X ?
— Si oui, quel est le nombre de classes ?
— Comment évaluer la qualité d'une partition ?
— Comment comparer deux partitions ?

Les mesures de qualité d'une partition peuvent être :

1. complètement non supervisées (ne s'appuyer sur aucune information externe) : on distingue ici les mesures de cohésion d'un nuage de points, et de séparation des nuages de points. On trouve aussi les indices se basant sur la matrice de proximité.
2. supervisées, qui vérifient à quel point une partition est conforme à une consigne extérieure. L'entropie est un exemple de mesure de ce type
3. relatives, qui permettent de comparer différentes partitions, ou nuages individuels.

Dans la suite, nous nous intéressons au premier point, ce chapitre traitant des algorithmes de classification non supervisés.

2.2.1 Cohésion et séparation

Lorsque les nuages de points sont caractérisés par leur centre μ_i (k-means par exemple), cohésion et séparation peuvent être exprimées par :

$$\begin{aligned} cohesion_1(P_i) &= \sum_{x \in P_i} \Delta(x, \mu_i) \\ separation(P_i, P_j) &= \Delta(\mu_i, \mu_j) \\ separation_1(P_j) &= \Delta(\mu_i, \mu) \end{aligned}$$

où μ est le barycentre des μ_i et Δ est une mesure de similarité (ou dissimilarité).
Lorsque la partition est calculée à l'aide d'un graphe, ces définitions peuvent s'écrire :

$$\begin{aligned} cohesion^1(P_i) &= \sum_{x,y \in P_i} \Delta(x, y) \\ separation^1(P_i, P_j) &= \sum_{x \in P_i, y \in P_j} \Delta(x, y) \end{aligned}$$

Il est alors possible d'exprimer la validité I d'une partition $P = (P_1 \cdots P_g)$ en calculant une combinaison linéaire des validités calculées sur chacune des P_i. Les poids dépendent de la mesure de validité (cohésion, séparation) utilisée, et par exemple :

— pour des nuages de points caractérisés par leur centre on peut trouver

$$\begin{aligned} I_1 &= \sum_{i=1}^{g} cohesion_1(P_i) \\ I_2 &= \sum_{i=1}^{g} |P_i| separation_1(P_j) \end{aligned}$$

— pour des nuages de points caractérisés par des graphes :

$$\begin{aligned} I^1 &= \sum_{i=1}^{g} \frac{cohesion^1(P_i)}{|P_i|} \\ I^2 &= \frac{1}{cohesion^1(P_i)} \sum_{i=1}^{g} \sum_{j \neq i} separation^1(P_i, P_j) \end{aligned}$$

Il est possible de combiner cohésion et séparation en un indice unique, l'indice de silhouette [**?**] d'un point de données, d'une classe ou de la partition. Cet indice se calcule en trois étapes, illustrées dans le cas d'un point x_i :

1. calcul de la distance moyenne de x_i à chaque x_j de sa classe P_i : $a_i = \frac{1}{|P_i|-1} \sum_{x_j \in P_i} d(x_i, x_j)$
2. pour une classe P_j ne contenant pas x_i, calcul de la distance moyenne de x_i à tous les points de P_j : $b_i = \frac{1}{|P_j|} \sum_{x_k \in P_j} d(x_i, x_k)$
3. calcul de l'indice de silhouette de x_i : $s_i = \frac{b_i - a_i}{max(a_i, b_i)}$.

On a clairement $-1 \leq s_i \leq 1$. De plus $s_i < 0$ signifie que $a_i > b_i$ ce qui ne correspond pas à une situation acceptable pour une bonne partition. s_i doit donc être positif, et a_i le plus proche possible de 0 (pour avoir $s_i \approx 1$).
A partir des indices de silhouette individuels, il est possible de calculer les indices des classes P_j (par moyenne des indices des points de la classe), et de la partition P (par moyenne des indices des classes P_j).

2.2.2 Nombre de classes

Le nombre de classes peut être suggéré en traçant l'un des indices précédents (souvent l'indice de silhouette) en fonction du nombre de classes, et en regardant le nombre de classes permettant d'atteindre un extremum de l'indice (le max pour l'indice de silhouette, voir par exemple la figure 4).

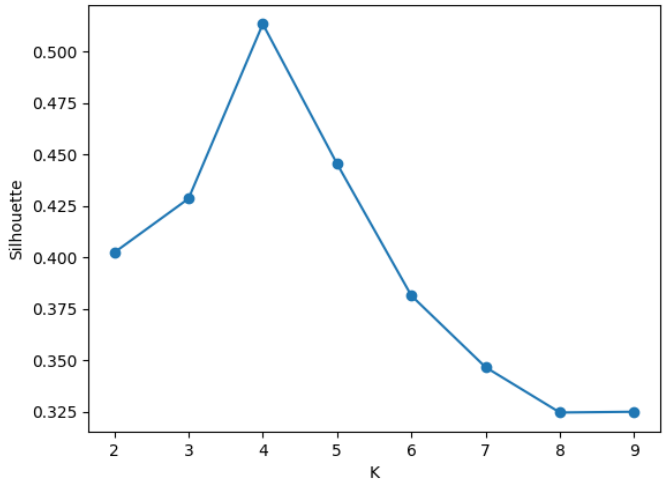

Fig. 4: Indice de silhouette en fonction du nombre de classes

3 Partie pratique

3.1 Prise en main de `sklearn.cluster`

3.1.1 K-means

Le module `sklearn.cluster` inclut de nombreux algorithmes de classification non supervisés, dont `KMeans`.

1. utiliser les fonctions fournies dans le module `sklearn.datasets` pour créer un jeu de données comme celui proposé en figure 5.
2. Appliquer l'algorithme des Kmeans et tracer une évolution des résultats, comme proposé sur la figure 6. On affichera la somme des inerties intra-classes, qui doit diminuer avec les itérations.
3. Les Kmeans, dans leur version initiale, ne fonctionnent que pour des nuages hypersphériques. Proposer des données de synthèse mettant en défaut l'algorithme de classification, comme proposé dans la figure 7. On pourra utiliser des fonctions du module `sklearn.datasets`.
4. utiliser la fonction `load_digits` pour charger un jeu de 1797 images 8×8 de chiffres manuscrits, étiquétés par leur label (le numéro correspondant). A l'aide

d'une mesure de l'indice de silhouette en fonction du nombre de classes, trouver le nombre de classes optimal (qui doit être 10...). Classer ensuite les données brutes (les images) et vérifier à l'aide des labels disponibles dans le jeu de données le taux de bonne classification.

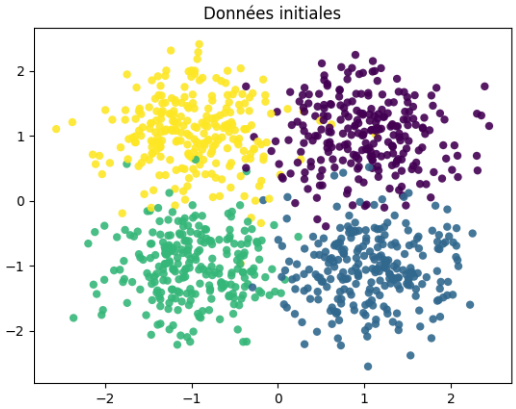

Fig. 5: Jeu de données de synthèse

3.1.2 Clustering spetral

De même, scikit-learn propose une classe `SpectralClustering` permettant d'appliquer des algorithmes de clustering spectral. Appliquer l'algorithme sur les données qui ont mis en défaut les K-means (figure 7).

3.1.3 MeanShift

Générer n nuages de 1000 points 2D , centrés sur des points choisis aléatoirement. Le nombre de classes n est lui aussi choisi aléatoirement dans [2,10]. Utiliser les fonctions `MeanShift` et `estimate_bandwidth` pour classer vos données à l'aide de l'algorithme MeanShift (figure 9). Estimer le nombre de classes n résultant de l'algorithme.

3.1.4 Clustering hiérarchique

La classe `AgglomerativeClustering` implémente les algorithmes de clustering hiérarchique. En générant des données comme pour l'algorithme MeanShift (mais avec 60 points pour une meilleure visualisation, notamment des dengrogrammes), construire les hiérarchies indicées en utilisant les liens moyen, max et de Ward, et déterminer le nombre optimal de classes.
Scikit ne propose pas d'afficher le dendrogramme, mais SciPy le permet. Utiliser la fonction `scipy.cluster.hierarchy.dendrogram` pour afficher les dendrogrammes correspondants aux trois classifications.

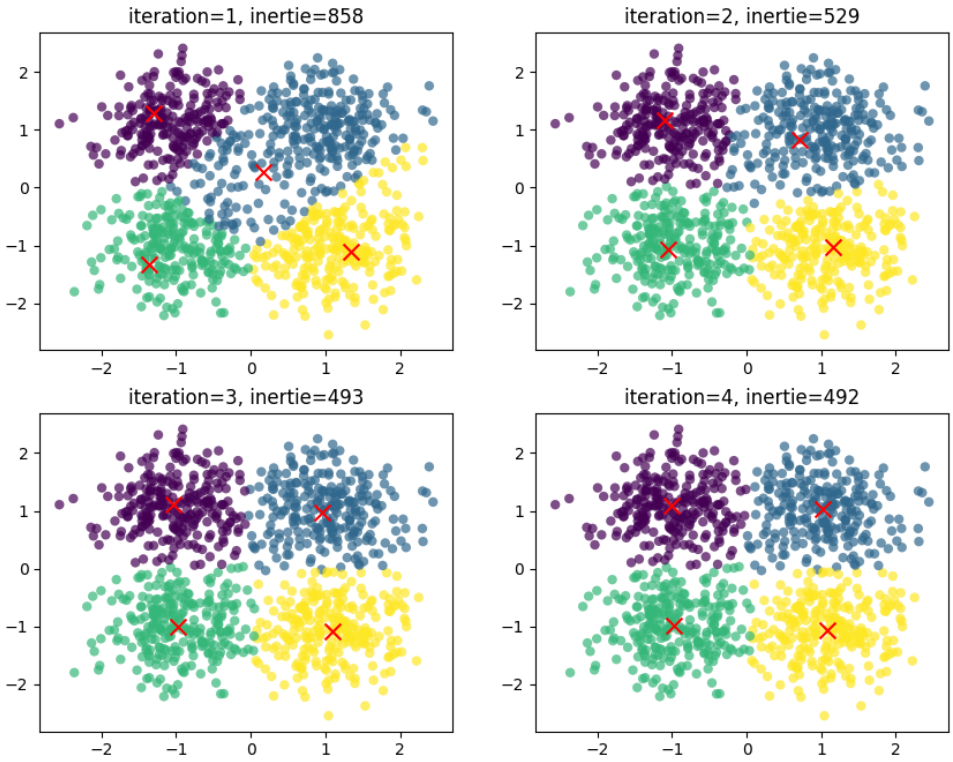

Fig. 6: Résultat de la classification par K-means

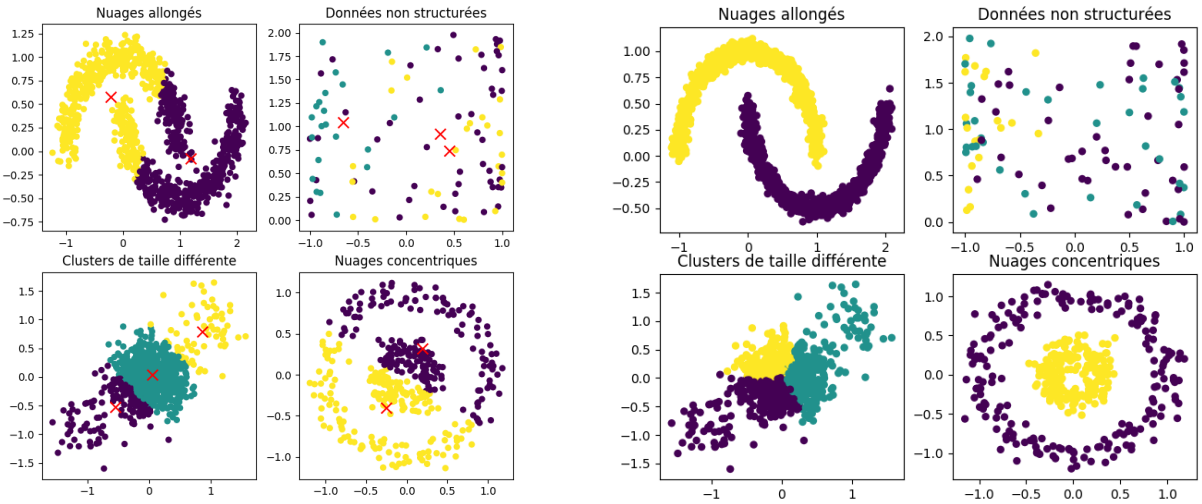

Fig. 7: Mise en défaut des k-means sur des nuages particuliers (gauche). Résolution par le clustering spectral (droite)

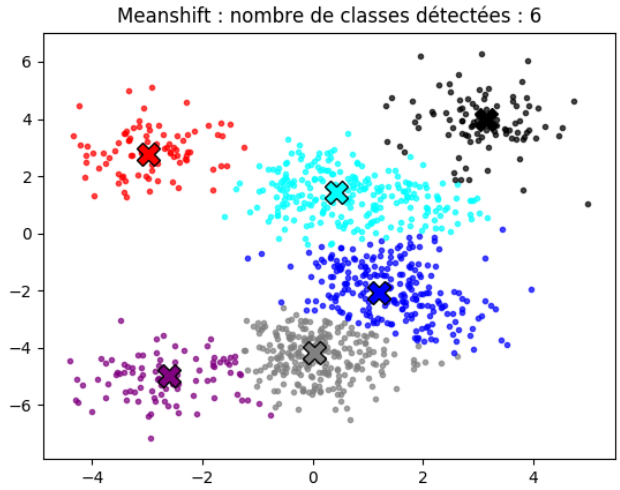

Fig. 8: Exemple de classification d'un jeu de données aléatoire par MeanShift

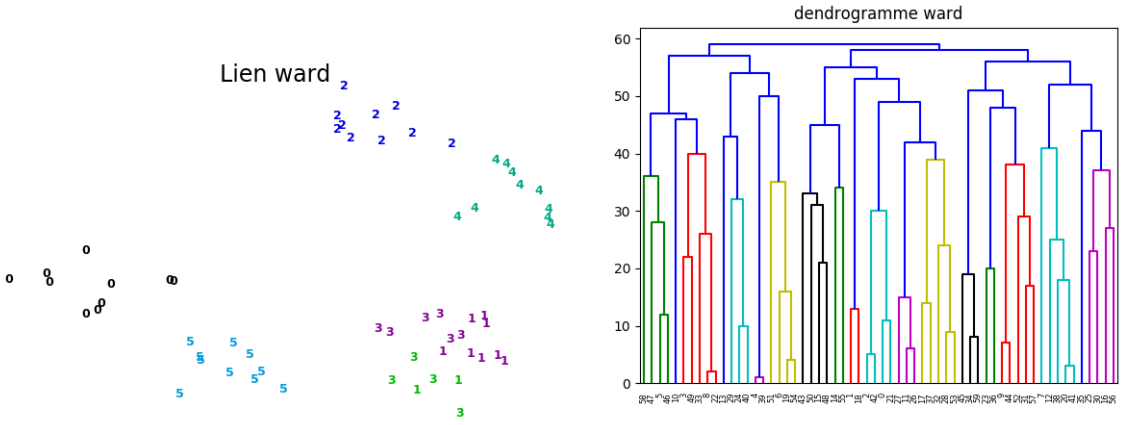

Fig. 9: Classification hiérarchique avec lien de Ward.

3.2 Challenge

On donne le jeu de données `challenge_clustering` (figure 10). Proposer une méthodologie (recherche du nombre de classes, algorithme de classification, choix des paramètres de l'algorithme, évaluation de la partition), à partir des algorithmes précédemment testés, ou d'autres que vous pourrez trouver dans le module `sklearn.cluster`. Évaluer pour chaque méthode testée la pertinence des résultats, et expliquer la partition obtenue en fonction des caractéristiques de l'algorithme utilisé et de ses paramètres.

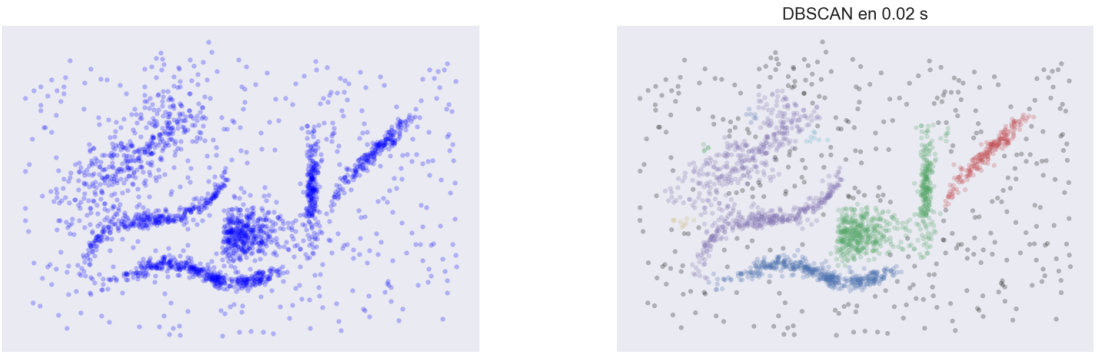

Fig. 10: Jeu de données du challenge et résultat de DBSCAN