14-16 rue Voltaire
94270 Kremlin Bicêtre

Benjamin DEVÈZE
Matthieu FOUQUIN
PROMOTION 2005
SCIA

DATAMINING
C4.5 - DBSCAN

Mai 2004

Responsable de spécialité SCIA : **M. Akli Adjaoute**

Table des matières

Table des figures

Liste des Algorithmes

Chapitre 1

Introduction

Le but du présent rapport est de présenter deux algorithmes utilisés dans le cadre du datamining. Dans un premier temps nous présenterons l'algorithme classique qu'est C4.5. Celui-ci étant une extension naturelle de l'algorithme ID3 nous présenterons d'abord ID3 puis forts de cette étude nous aborderons plus en détails C4.5 et ses améliorations. Puis dans un second temps nous aborderons l'algorithme DBSCAN qui est un algorithme de clusterisation non supervisé qui travaille sur les densités et qui comme nous allons le voir s'avère extrêment basique.

Chapitre 2

L'algorithme C4.5

2.1 Petit rappel concernant le datamining

Commençons par une entrée en matière très succinte sur le domaine vaste et pluridisciplinaire qu'est le datamining.

2.1.1 Définition

Ensemble des techniques et de méthodes du domaine des statistiques, des mathématiques et de l'informatique permettant l'extraction, à partir d'un important volume de données brutes, de connaissances originales auparavant inconnues. Il s'agit de fouilles visant à découvrir de l'information cachée que les données renferment et que l'on découvre à la recherche d'associations, de tendances, de relations ou de régularités.

2.1.2 Quelques applications

Comme le suggère la définition, le datamining est une spécialité transverse : elle regroupe un ensemble de théories et d'algorithmes ouverts à tout domaine métier susceptible de drainer une masse de données. La liste suivante illustre des applications courantes du datamining, mais elle reste loin de l'exhaustivité :

- **CRM**
  - scoring client pour une gestion adapté du risque dans un établissement financier
  - profiling client pour un service marketing d'une entreprise de distribution
  - profiling produit pour améliorer le cross-selling dans la grande distribution

- **industrie**
  - optimisation / fiabilisation d'une chaîne de montage
  - système expert de résolution de panne par la description des symptômes
  - prévision de pics de consommation d'un réseau (téléphone, énergie électrique...)

- **traitement d'images**
  - reconnaissance de forme
  - reconnaissance de signatures biométriques

- **outils de collaboration**
  - classification dynamique et contextuelle de documents non structurés
  - mise en relation de personnes par la création automatique de profil de centres d'intérêt

L'appel croissant et varié au datamining tient principalement aux facteurs suivants :

- la gestion des données est facilitée par la puissance accrue des ordinateurs

- les entreprises se sont accoutumées à manipuler des volumes toujours plus importants de données sous la contrainte des optimisations qu'elles doivent sans cesse accomplir pour leur survie.
- progressivement les entreprises prennent conscience qu'au delà de l'usage courant que chacun fait de ses données, celles-ci renferment également des notions invisibles à l'oeil nu.
- les méthodes de datamining sont très efficaces pour la compréhension approfondie de l'information que recèle un ensemble de données.

2.1.3 Aperçu général des techniques classiques

Selon le type de données disponibles et le type de connaissances recherchées, la méthode d'obtention des règles finales va varier grandement. Chacune des techniques décrites ci-dessous possède certains avantages et certains inconvénients. Face à une problématique il convient de connaître chacune d'entre elle pour apporter la solution la plus efficace :

- **Les ensembles fréquents :** c'est une technique consistant à considérer les occurences communes de plusieurs valeurs au sein d'un grand nombre d'enregistrements. Surtout utile pour des enregistrements constitués d'une liste d'éléments discrets. A ce niveau, on peut soulever le prblème selon lequel ce genre de méthode possède une limite : les ensembles d'éléments à valeurs continues. L'aboutissement de cette méthode est un ensemble de règles qui font sortir de la base des tendances à avoir certains éléments lorsqu'ils sont dans un ensemble d'autres éléments bien définis.

- **Les arbres de décision :** c'est une technique de classification automatisée. L'analyse d'un ensemble d'enregistrements préalablement classifiés permet de générer une structure arborescente optimale pour la classification des autres enregistrements disponibles. Ceci est fait en partitionnant successivement la population initiale des enregistrements disponibles. Ceci est fait en partitionnant successivement la population initiale des enregistrements de sorte à isoler au mieux les enregistrements d'une même classe. Ce sont pour la plupart des algorithmes légers, performants, et la forme arborescente des résultats permet une grande lisibilité.

- **Les réseaux neuronaux :** c'est une technique de classification automatisée. De même que pour les arbres de décision, le principe consiste à apprendre à correctement classifier des données à partir d'un jeu d'exemples déjà classifiés. Présentement, les champs d'information des enregistrements forment les entrées d'un réseau dont la sortie correspond à la classe de l'enregistrement. L'apprentissage consiste alors à faire passer les enregistrements classés en entrée du réseau, et à corriger un petit peu l'erreur fatalement obtenue en sortie en modifiant les noeuds internes du réseau. Au fur et à mesure, celui-ci s'adapte, et finit par classer correctement les enregistrements. Si ces algorithmes sont puissants, ils nécessitent bien évidemment plus de traveux de mise en oeuvre.

- **Les réseaux bayésiens :** c'est une technique permettant la modélisation de la connaissance sous la forme d'un réseau dont les noeuds correspondent à des évènements affectés de leur probabilités respectives. Des liens de causalité permettent en outre de modeler ces probabilités selon la connaissance que l'on a de certains autres évènements. Dans le cadre de l'extraction de connaissance, cela signifie que ces applications sont capables d'inférer des connaissances à partir d'enregistrements incomplets.

- **Les algorithmes génétiques :** c'est une technique d'optimisation permettant d'éviter l'explosion combinatoire du nombre de solutions à un problème. Leur application à l'extraction de connaissance permet l'exploration de l'ensemble de toutes les règles possibles entre les données, afin de converger vers les plus intéressantes.

Les méthodes d'apprentissage peuvent être représentée sur un arbre. Leur position dans l'arbre dépend du type d'apprentissage donc de la nature des algorithmes auxquelles elles font appel. L'arbre de classification des méthodes pourrait se présenter ainsi :

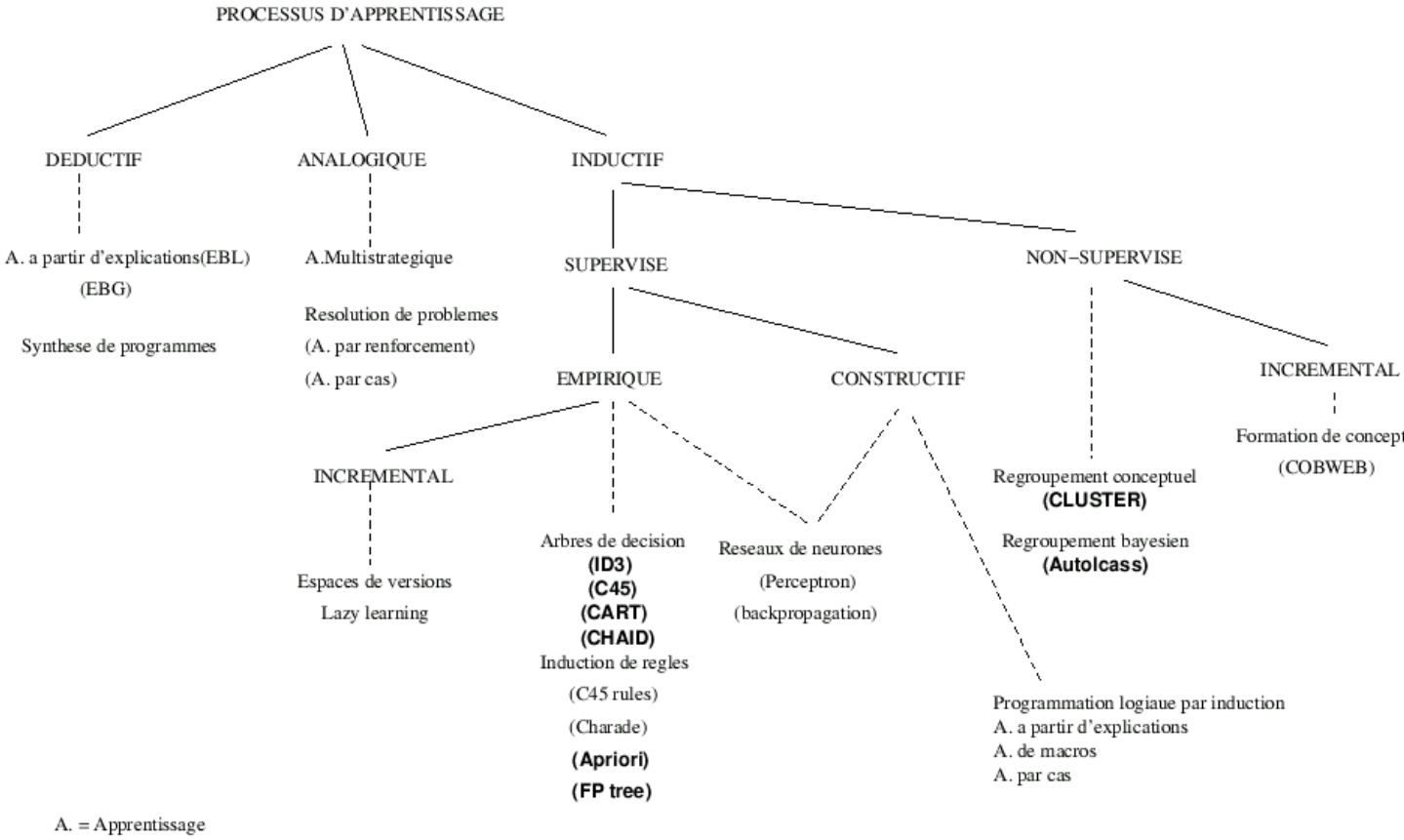


FIG. 2.1 – Arbre de classification des méthodes de datamining

2.2 L'algorithme ID3

2.2.1 Arbres de décision

Un arbre de décision est une structure qui permet de déduire un résultat à partir de décisions successives. Pour parcourir un arbre de décision et trouver une solution il faut partir de la racine. Chaque noeud est une décision atomique. Chaque réponse possible est prise en compte et permet de se diriger vers un des fils du noeud. De proche en proche, on descend dans l'arbre jusqu'à tomber sur une feuille. La feuille reprśente la réponse qu'apporte l'arbre au cas que l'on vient de tester.

- Débuter à la racine de l'arbre
- Descendre dans l'arbre en passant par les noeuds de test
- La feuille atteinte à la fin permet de classer l'instance testée.

Très souvent on considère qu'un noeud pose une question sur une variable, la valeur de cette variable permet de savoir sur quels fils descendre. Pour les variables énumérées il est parfois possible d'avoir un fils par valeur, on peut aussi décider que plusieurs variables différentes mènent au même sous arbre. Pour les variables continues il n'est pas imaginable de créer un noeud qui aurait potentiellement un nombre de fils infini, on doit discrétiser le domaine continu (arrondis, approximation), donc décider de segmenter le domaine en sous ensembles.

Plus l'arbre est simple, et plus il semble techniquement rapide à utiliser. En fait, il est plus intéressant d'obtenir un arbre qui est adapté aux probabilités des variables à tester. La plupart du temps un arbre équilibré sera un bon résultat. Si un sous arbre ne peut mener qu'à une solution unique, alors tout ce sous-arbre peut être réduit à sa simple conclusion, cela simplifie le traitement et ne change rien au résultat final. Ross Quinlan a travaillé sur ce genre d'arbres de décision.

2.2.2 Introduction

L'algorithme ID3 fut proposé par Quinlan en 1979 afin de générer des arbres de décisions à partir de données. Imaginons que nous ayons à notre disposition un ensemble d'enregistrements. Tous les enregistrements ont la même structure, à savoir un certain nombre de paires attribut/valeur. L'un de ses attributs représente la catégorie de l'enregistrement. Le problème consiste à construire un arbre de décision qui sur la base de réponses à des questions posées sur des attributs non cibles peut prédire correctement la valeur de l'attribut cible. Souvent l'attribut cible prend seulement les valeurs vrai, faux ou échec, succès.

Étudions tout de suite un exemple simple pour clarifier nos propos. Nous avons à notre disposition des enregistrements concernant les conditions météorologiques pour jouer au golf. L'attribut cible ici détermine si compte tenu des conditions il est possible de jouer ou non, il prend les valeurs : JOUER et NE PAS JOUER. Les attributs non cibles sont :

- **VISIBILITE :** soleil, pluie, couvert
- **TEMPERATURE :** valeur
- **HUMIDITE :** valeur
- **VENT :** vrai, faux

Voici les données dont nous disposons pour servir de données d'apprentissage :

VISIBILITE	TEMPERATURE	HUMIDITE	VENT	JOUER
soleil	85	85	faux	NE PAS JOUER
soleil	80	90	vrai	NE PAS JOUER
couvert	83	78	faux	JOUER
pluie	70	96	faux	JOUER
pluie	68	80	faux	JOUER
pluie	65	70	vrai	NE PAS JOUER
couvert	64	65	vrai	JOUER
soleil	72	95	faux	NE PAS JOUER
soleil	69	70	faux	JOUER
pluie	75	80	faux	JOUER
soleil	75	70	vrai	JOUER
couvert	72	90	vrai	JOUER
couvert	81	75	faux	JOUER
pluie	71	80	vrai	NE PAS JOUER

FIG. 2.2 – Données d'apprentissage pour notre exemple

Notons que dans cet exemple 2 des attributs peuvent prendre des valeurs sur un intervalle continu, la température et l'humidité. ID3 ne gère pas directement ce genre de cas, cependant nous verrons plus tard comment il peut être étendu pour ce type de situation. Un arbre de décision est important non pas parce qu'il résume ce que l'on sait, à savoir les données d'apprentissage, mais parce que nous espérons qu'il pourra classer correctement les nouveaux cas. C'est pourquoi lorsque l'on cherche à construire un tel arbre nous devons disposer à la fois de données d'apprentissage mais également de données de test pour estimer la qualité de l'arbre construit.

Les principales idées sur lesquels repose ID3 sont les suivantes :

- Dans l'arbre de décision chaque noeud correspond à un attribut non cible et chaque arc à une valeur possible de cet attribut. Une feuille de l'arbre donne la valeur escomptée de l'attribut cible pour l'enregistrement testé décrit par le chemin de la racine de l'arbre de décision jusqu'à la feuille. (Définition d'un arbre de décision)

- Dans l'arbre de décision, à chaque noeud doit être associé l'attribut non cible qui apporte le plus d'information par rapport aux autres attributs non encore utilisés dans le chemin depuis la racine. (Critère d'un bon arbre de décision)

- L'entropie est utilisée pour mesurer la quantité d'information apportée par un noeud. (Cette notion a été introduite par Claude Shannon lors de ses recherches concernant la théorie de l'information qui sert de base à énormément de méthodes du datamining.)

2.2.3 Quelques définitions de théorie de l'information

Les théories de Shannon étant à la base de l'algorithme ID3 et donc de C4.5 nous allons faire quelques rappels succints de théorie de l'information.

Si il y a n messages possibles équiprobables, alors la probabilité p de chacun est de $\frac{1}{n}$ et l'information portée par un message est de -log(p) = log(n) (Dans ce qui suit tous les logarithmes sont en base 2). Ainsi si il y a 16 messages, alors log(16) = 4 et on a besoin de 4 bits pour identifier chaque message.

En général, si on nous donne une distribution de probabilité P = (p1, p2, ..., pn) alors l'Information portée par cette distribution, aussi appelée **l'Entropie** de P, est :

$$I(P) = -(p1*\log(p1) + p2*\log(p2) + ... + pn*\log(pn))$$

Par exemple, si P est (0.5, 0.5) alors I(P) est 1, si P est (0.67, 0,33) alors I(P) vaut 0.92, si P est (1, 0) alors I(P) vaut 0. (Notons que plus la distribution est uniforme, plus l'information est grande)

Si un ensemble T d'enregistrements forment une partitition C1, C2, ..., Ck sur la base de la valeur de l'attribut cible, alors l'information nécessaire à l'identification de la classe d'une élément de T est Info(T) = I(P), où P est la distribution probabiliste de la partition (C1, C2, ..., Ck) :

$$P = (\frac{|C1|}{|T|}, \frac{|C2|}{|T|}, ..., \frac{|Ck|}{|T|})$$

Dans notre exemple, nous avons ainsi Info(T) = I(9/14, 5/14) = 0.94.

Si nous partitionnons d'abord T sur la base des valeurs d'un attribut non cible X en ensembles T1, T2, ..., Tn alors l'information nécessaire pour identifier la classe d'un élément de T devient la moyenne pondérée de l'information nécessaire à l'idenfication de la class d'un élément de Ti, à savoir la moyenne pondérée de Info(Ti) :

$$\text{Info(X, T)} = \sum_{i=1}^{n}(\frac{|Ti|}{|T|} * Info(Ti))$$

Dans le cas de notre exemple, pour l'attribut Visibilite nous avons :

$$\text{Info(Visibilite, T)} = 5/14 * I(2/5, 3/5) + 4/14 * I(4/4, 0) + 5/14 * I(3/5, 2/5) = 0.694$$

Considérons la quantité Gain(X, T) défini comme suit :

$$\text{Gain(X, T)} = \text{Info(T)} - \text{Info(X, T)}$$

Cela représente la différence entre l'information nécessaire pour identifier un élément de T et l'information nécessaire pour identifier un élément de T après que la valeur de l'attribut X ait été obtenu, en d'autre terme il s'agit du gain en information dû à l'attribut X.

Dans notre exemple, pour la Visibilité le gain est de :

$$\text{Gain(Visibilite,T)} = \text{Info(T)} - \text{Info(Visibilite,T)} = 0.94 - 0.694 = 0.246.$$

Si nous considérons l'attribut Vent, nous trouvons que Info(Vent, T) vaut 0.892 et Gain(Vent, T) vaut 0.048. On en déduit que la Visibilité offre plus d'informations que l'attribut Vent.

Nous pouvons utiliser cette notion de gain pour classer les attributs et construire un arbre de décision où à chaque noeud se trouve l'attribut qui a le gain le plus grand par rapport aux attributs non encore instanciés.

L'intérêt de cet ordonnancement est de créer un petit arbre de décision ce qui permet d'identifier un enregistrement avec un petit nombre de question.

2.2.4 L'algorithme ID3

Comme nous l'avons vu l'algorithme ID3 est utilisé pour construire un arbre de décision, étant donné un ensemble d'attributs non cibles C1, C2, ..., Cn, l'attribut cible C, et un ensemble S d'enregistrements d'apprentissage.

Algorithm 1 Algorithme ID3

Require: R : un ensemble d'attributs non cible, C : l'attribut cible, S : données d'apprentissage
Ensure: retourne un arbre de décision

1: **function** ID3(R, C, S)
2: **if** S est vide **then**
3: **return** un simple noeud de valeur Echec
4: **end if**
5: **if** S est constitué uniquement de valeurs identiques pour la cible **then**
6: **return** un simple noeud de cette valeur
7: **end if**
8: **if** R est vide **then**
9: **return** un simple noeud avec comme valeur la valeur la plus fréquente des valeurs de l'attribut cible trouvées dans S
10: **end if**
11: D ← l'attribut qui a le plus grand gain(D,S) parmi tous les attributs de R
12: $\{d_j avec j = 1, 2, ..., m\}$ ← les valeurs des attributs de D
13: $\{S_j avec j = 1, 2, ..., m\}$ ← les sous ensembles de S constitués respectivement des enregistrements de valeur d_j pour l'attribut D
14: **return** un arbre dont la racine est D et les arcs sont étiquetés par $d_1, d_2, ..., d_m$ et allant vers les sous arbres ID3(R-{D}, C, S1), ID3(R-{D}, C, S2), .., ID3(R-{D}, C, Sm)
15: **end function**

A titre d'exemple voici l'arbre qu'on obtiendrait pour l'exemple du golf :

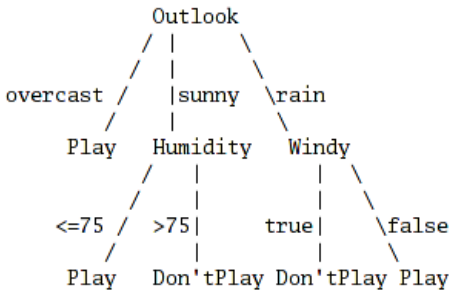


FIG. 2.3 – Arbre de classification obtenu pour l'exemple

2.3 L'algorithme C4.5

Cet algorithme a été proposé en 1993, toujours par Ross Quinlan, pour pallier les limites de l'algorithme ID3 vu précédemment. Nous n'allons pas tout redévelopper pour décrire C4.5 car il repose complètement sur l'algorithme ID3 que nous avons déjà décrit. Nous nous focaliserons donc davantage ici sur les limites de l'algorithme ID3 et les améliorations apportées par C4.5.

2.3.1 Les limites de l'algorithme ID3

ID3 pose quelques problèmes. D'abord nous avons vu qu'il vaut mieux avoir des données exhaustives, ensuite, la quantité de calcul est assez importante. Cet algorithme n'applique pas une recherche exhaustive, mais permet de se fier aux probabilités des attributs qui ont le plus de chance d'aiguiller le résultat. Un problème apparaît si on considère que par exemple on peut jouer au golf si il fait beau ou si il n'y a pas de vent. La présence du OU va provoquer de la redondance dans les tests des différents sous arbres. Le XOR est encore plus problématique. Si l'on résume l'algorithme ID3 ne peut pas traiter les enregistrements incomplets, les attributs sont discrétisés, ce qui n'est pas toujours une solution acceptable, enfin l'arbre produit peut comporter des sous arbres dans lesquels on ne va presque jamais. Voyons comment l'algorithme C4.5 permet de répondre à ces limitations de l'algorithme ID3.

2.3.2 Les extensions de l'algorithme C4.5

C4.5 introduit un certain nombre d'extensions à ID3.

Les attributs de valeur inconnue

Durant la construction de l'arbre de décision il est possible de gérer les données pour lesquels certains attributs ont une valeur inconnue en évaluant le gain, ou le gain ratio[1] pour un tel attribut en considérant seulement les enregistrements pour lesquels cet attribut est défini.

En utilisant un arbre de décision, il est possible de classer les enregistrements qui ont des valeurs inconnus en estimant la probabilité des différents résultats possibles. Dans notre exemple du golf, si on nous donne un nouvel enregistrement pour lequel la Visibilité est soleil et l'humidité est inconnu, nous procédrons alors comme suit :
Nous allons du noeud racine Visibilité vers le noeud humidité en suivant l'arc soleil. Comme on ne connait pas la valeur de l'Humidité on observe que si l'humidité est inférieure à 75 il y a 2 enregistrements où l'on joue, et si l'humidité est au dessus de 75 il y a 3 enregistrements dont l'un où l'on ne joue pas. On peut donc donner comme réponse pour l'enregistrement les probabilités suivantes (0.4, 0.6) de jouer ou de ne pas jouer.

[1]Cf. la notion de Gain Ratio

Les attributs à valeur sur intervalle continu

Il est maintenant également de gérer le cas d'attributs à valeur dans des intervalles continus, de la façon suivante. Disons que l'attribut Ci a un intervalle continu de valeurs. On examine les valeurs de cet attribut dans les données d'apprentissage. Disons que ces valeurs sont en ordre croissant, A1, A2, ..., Am. Ensuite pour chacune de ces valeurs, on partitionne les enregistrements entre ceux qui ont des valeurs de Ci inférieures ou égales à Aj et celles qui ont des valeurs supérieures à Aj. Pour chacune de ces partitions on calcule le gain, ou le gain ratio et on choisit la partition qui maximise le gain.

Dans notre exemple de golf, pour l'humidité, si T est l'ensemble d'apprentissage, on détermine l'information de chaque partition et on trouve la meilleur partition à 75. Donc l'intervalle pour cet attribut devient ¡=75, ¿75. On notera quand même que cette méthode nécessite un nombre conséquent d'opérations.

La notion de Gain Ratio

La notion de Gain introduite plus tôt tend à favoriser les attributs qui ont un nombre important de valeurs. Par exemple, si nous avons un attribut D qui a une valeur différente pour chaque enregistrement, alors Info(D, T) vaut 0, donc Gain(D, T) sera maximal. Pour compenser cet état de fait, Quinlan suggère d'utiliser le calcul pondéré suivant plutôt que le Gain :

$$\text{GainRatio(D, T)} = \frac{Gain(D,T)}{SplitInfo(D,T)}$$

où SplitInfo(D, T) = I($\frac{|T1|}{T}$, $\frac{|T2|}{T}$, ..., $\frac{|Tm|}{T}$) avec {T1, ..., Tm} la partition de T induite par la valeur de D.

Dans le cas de notre exemple :
SplitInfo(Visibilité, T) = -5/14 * log(5/14) - 4/14 * log(4/14) - 5/14 * log(5/14) = 1.577
GainRatio(Visibilité) = 0.246/1.577 = 0.156.

SplitInfo(Vent, T) = -6/14 * log(6/14) - 8/14 * log(8/14) = 6/14 * 0.1.222 + 8/14 * 0.807 = 0.985.
GainRatio(Vent) = 0.048/0.985 = 0.049

L'élagage de l'arbre de décision

L'arbre de décision construit en utilisant l'ensemble d'apprentissage, du fait de la façon dont il a été construit, traite correctement la plupart des enregistrements du jeux d'apprentissage.

L'élagage de l'arbre de décision s'effectue en remplaçant un sous arbre entier par une feuille. Cet substitution a lieu si une règle de décision établit que le taux d'erreur attendu dans le sous arbre est supérieur que celui d'une simple feuille. Exemple :

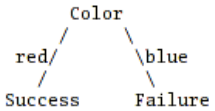


FIG. 2.4 – Exemple sur un arbre simple

Si cet arbre de décision est obtenu avec 1 enregistrement rouge succès et 2 bleus échec, et que dans le jeux de test on trouve 3 enregistrement rouge échec et un bleu succès, on peut remplacer ce sous arbre par un simple noeud échec. Après cette substitution on aura plus que 2 erreurs au lieu de 5.

Winston a montré comment utiliser le test de Fischer pour déterminer si la catégorie de l'attribut est vraiment dépendante d'un attribut non cible. Si ce n'est pas le cas, alors l'attribut non cible n'a

pas besoin d'apparaître dans l'arbre de décision.

Quinlan et Breiman ont suggéré des heuristiques d'élagages plus sophistiquées.

C'est simple de dériver un ensemble de règle à partir d'un arbre de décision : il suffit d'écrire une règle pour chaque chemin de l'arbre qui va de la racine à une feuille. Dans cette règle la partie gauche est construite facilement à partir des noeuds et des arcs.

L'ensemble des règles qui en résulte peut être simplifié :
posons LHS la partie gauche de la règle, LHS' est obtenu à partir de LHS en éliminant certaines de ses conditions. On peut remplacer LHS par LHS' dans cette règle si le sous ensemble de l'ensemble d'apprentissage qui satisfait respectivement LHS et LHS' sont égaux.

Une règle peut être éliminée en utilisant des métaconditions telles que « Si aucune autre règle ne s'applique ».

Commentaires

Depuis d'autres versions améliorées ont vu le jour, les efforts concernent notamment la vitesse de calcul, la quantité de mémoire utilisée, et la taille des arbres générés.

On mesure la capacité de prédiction au pourcentage d'erreur que peuvent provoquer les algorithmes. Tout comme ID3, C4.5 est pénalisé par les systèmes ayant des règles utilisant des OR our des XOR. Dans certains cas on peut être réduit à seulement 50% de taux de prédiction. Un arbre de décision produit ce que l'on appelle des Hyper Rectangles, pour représenter les classes, ainsi les limites sont très facilement représentées et très nettes. Selon les cas ID3 et C4.5 peuvent être très efficaces (relation matière-indice de réfraction) ou bien moins bons (problèmes de Monk2, détection des maladies cardiques) que les autres algorithmes de datamining. C4.5 produit plus d'erreurs de prédiction que ID3, mais ces taux restent néanmoins très faibles.

Chapitre 3

L'algorithme DBSCAN

3.1 Les systèmes de clustering

3.1.1 Les Clusters

A la base, un cluster est un ensemble d'éléments. Cet ensemble est distinct des autres. Donc chaque élément d'un cluster a de fortes ressemblances avec les autres éléments de ce même cluster, et doit être différent des éléments des autres clusters. C'est ce que l'on appelle : la forte similarité intra-classe, et la faible similarité inter-classe. Il y a donc une idée de recherche des groupes distincts. Les méthodes d'analyse de clusters sont des algorithmes non-supervises, ils permettent de générer et de trouver des classes naturelles. Par exemple ce genre de méthodes de data mining est utilisé dans le marketing pour découvrir le profile de certains groupes de clients, et ainsi s'adapter à un marché. Une méthode d'analyse de clusters doit se montrer fiable, donc elle doit pouvoir créer des clusters bien distincts, être faiblement sensible au bruit, mettre à jour des patterns cachés, être insensible a l'ordre d'entrée des transactions. La souplesse face aux transactions est primordiale. Le prototype d'un cluster est son centre, aussi appelé centroid. Il existe deux types de clusters :

- les clusters durs qui sont totalement distincts les uns des autres, ainsi un élément d'un cluster n'est pas du tout dans un autre.
- les clusters mous comportent des éléments dont l'appartenance est pondérée, donc un élément peut être distribué parmi plusieurs clusters.

3.1.2 Propriétés d'un cluster

Les deux propriétés importantes définissant un cluster pertinent sont :

- sa cohésion interne (que les objets appartenant à ce cluster soient les plus similaires possibles)
- son isolation externe (que les objets appartenant aux autres clusters soient les plus éloignés possible).

Pour observer cela, plusieurs mesures sont associées à un cluster :

- sa densité (la masse d'objets par unité volumique)
- sa variance (le degré de dispersion des objets dans l'espace depuis le centre du cluster)
- sa dimension (typiquement son radius ou son diamètre)
- sa forme (hypersphérique/allongée/concave/convexe,...)
- sa séparation (par rapport aux autres clusters).

Si on regarde un cluster il forme un ensemble. Et cet ensemble occupe donc un espace. Pour pouvoir mesurer l'appartenance d'un élément a un cluster et pouvoir prendre des décisions il nous faut une fonction de mesure. On utilise beaucoup la distance de Minkowski :

$$d(i,j) = \sqrt[q]{|x_{i1} - x_{j1}|^q + |x_{i2} - x_{j2}|^q + \cdots + |x_{in} - x_{jn}|^q}$$

FIG. 3.1 – Distance de Minkowski

Plusieurs variantes sont utilisées, notamment avec q = 1 ou q = 2. D'autres distances sont utilisées parfois, telles que la paramétrique de Pearson...

3.1.3 Etapes d'un système de clustering

Typiquement, les systèmes de clustering se différencient par la fonction objectif choisie pour évaluer la qualité du clustering, et la stratégie de contrôle pour parcourir l'espace des clusters possibles. Mais tous suivent le principe général traditionnel en clustering qui consiste à maximiser la similarité des observations à l'intérieur d'un cluster, et minimiser la similarité des observations entre clusters, pour arriver à une partition de la base aussi pertinente que possible. Les différentes étapes d'une tâche de clustering sont les suivantes :

1. Représentation des données (inclut éventuellement extraction et/ou sélection d'attributs)
La représentation des données se réfère à la spécification du nombre de classes, nombre de données, et nombre, type et échelle des attributs disponibles pour l'algorithme de clustering.
- L'extraction des attributs correspond à l'utilisation d'une ou plusieurs transformations des attributs fournis en entrée pour produire de nouveaux attributs pertinents.
- La sélection des attributs est le processus permettant d'identifier le sous-ensemble des attributs le plus efficace à utiliser pour le clustering.

2. définition d'une mesure de proximité appropriée au domaine des données
La proximité entre données est typiquement mesurée par une fonction de distance définie entre paires de données.

3. regroupement (clustering)
Les clusterings résultant peuvent être hard (partition des données en groupes distincts), ou fuzzy (chaque donnée a un degré variable d'appartenance à chacun des clusters formés).

4. abstraction des données (si nécessaire)
L'abstraction des données est le processus d'extraction d'une représentation simple et compacte de l'ensemble des données (typiquement, la description de chaque cluster).

5. évaluation de la sortie (si nécessaire)
L'évaluation de la partition peut se faire de trois manières :
- évaluation externe : comparer la structure à une structure a priori
- évaluation interne : déterminer si la structure est intrinsèquement appropriée aux données
- évaluation relative : comparer différentes structures possibles

3.1.4 Les méthodes de clustering

On distingue trois grandes familles de clustering :

1. le clustering hiérarchique, dont le but est de former une hiérarchie de clusters, telle que plus on descend dans la hiérarchie, plus les clusters sont spécifiques à un certain nombre d'objets considérés comme similaires

2. le clustering par partition, dont le but est de former une partition de l'espace des objets, selon une certaine fonction critère, chaque partition représentant alors un cluster dans cette famille, plusieurs

méthodes se distinguent fortement :

- **le clustering K-means**, dont le but est d'identifier un certain nombre (K) de points représentatifs des clusters, auxquels sont ensuite associés l'ensemble des autres points, selon leur proximité avec les points représentatifs considérés

- **le clustering basé sur la densité**, dont le but est d'identifier, dans l'espace, les zones de forte densité entourées par des zones de faible densité, qui formeront les clusters

- **le clustering basé sur l'utilisation de grilles**, dont l'idée est d'utiliser une grille pour partitionner l'espace en un ensemble de cellules, puis d'identifier les ensembles de cellules denses connectées, qui formeront les clusters

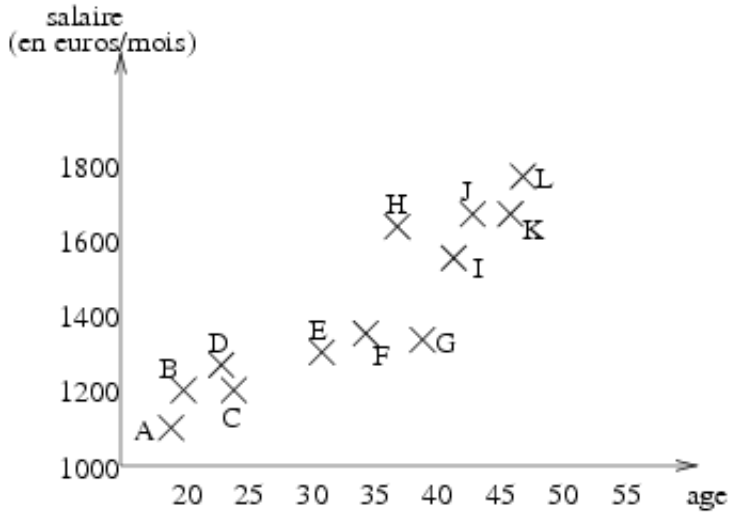
- **le clustering statistique**, qui fait l'hypothèse que les données ont été générées en suivant une certaine loi de distribution (avec une certaine probabilité), le but étant alors de trouver les paramètres (cachés) de cette distribution

- **le clustering via la théorie des graphes**, qui cherche, dans le graphe connectant les objets entre eux, les arcs à conserver pour former les clusters

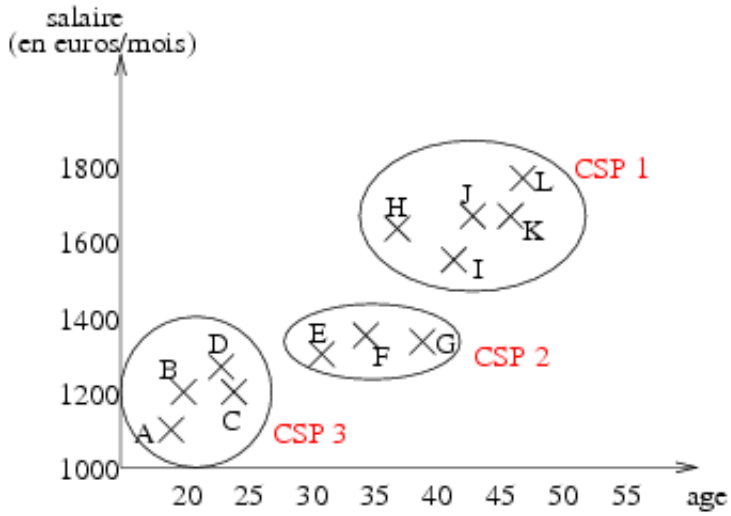
- **les clusterings basés sur la recherche stochastique** : algorithmes génétiques, recherche Tabou ou recuit simulé, qui parcourent l'espace des partitions possibles selon différentes heuristiques, et sélectionnent la meilleure qu'ils trouvent dans le temps qui leur est imparti

- **le clustering basé sur les réseaux de neurones**, appelés auto-associatifs, qui recherche les poids à attribuer à l'unique couche du réseau, qui correspondent le mieux à l'ensemble des données

3. le subspace clustering, dont le but est de cibler les clusters existant dans des sous-espaces de l'espace original.



(a) un exemple d'ensemble d'individus à classer...



(b) et le clustering résultat attendu

FIG. 3.2 – Le clustering sur un exemple

3.1.5 Propriétés des techniques de clustering

Plusieurs propriétés peuvent être associées aux différentes techniques de clustering :

- Ascendant versus descendant :
Une méthode ascendante va démarrer avec autant de clusters que d'objets, puis va concaténer successivement les clusters jusqu'à ce qu'un critère d'arrêt soit satisfait.

A l'inverse, une méthode descendante va démarrer avec un cluster réunissant tous les objets, puis va diviser les clusters jusqu'à ce qu'un critère d'arrêt soit satisfait.

- Déterministe versus stochastique :
Avec les mêmes données en entrée, un algorithme déterministe exécutera toujours la même suite d'opérations, et fournira donc toujours le même résultat.

A l'inverse, une méthode stochastique pourra donner des résultats différents pour des données en entrée identiques, car elle permet l'exécution d'opérations aléatoires.

Les algorithmes stochastiques sont donc moins précis mais moins coûteux. C'est pourquoi ils sont utilisés lorsqu'on a à faire face à de larges bases de données.

- Incrémental versus non-incrémental :
Une méthode incrémentale va être exécutée de façon continue, et va intégrer les données au fur et à mesure de leur arrivée dans l'algorithme.

A l'inverse, une méthode non-incrémentale va considérer un ensemble de données fournies en entrée, et sera exécutée sur cet ensemble de données. Si, par la suite, une nouvelle donnée devait être fournie en entrée de l'algorithme, celui-ci devrait être relancé à nouveau.

- Hard versus Fuzzy :
Comme indiqué précédemment, une méthode Hard va associer à chaque objet un unique cluster, alors qu'une méthode Fuzzy va associer à chaque objet un degré d'appartenance à chaque cluster.

A noter qu'un Fuzzy clustering peut être converti en un Hard clustering en assignant chaque donnée au cluster dont la mesure d'appartenance est la plus forte.

- Monothetic versus polythetic : Un algorithme monothetic va utiliser séquentiellement les attributs des données dans le processus de clustering. A l'inverse, un algorithme polythetic va utiliser simultanément les attributs des données dans le processus de clustering.

3.2 DBSCAN

3.2.1 Intérêt de DBSCAN

De nombreuses applications ont besoin d'une gestion de données spatiales tel que les SDBS (Spatial Database Systems). Une quantité croissante de données est obtenue d'images satellites, de crystallographie aux rayons X ou d'autres équipements automatiques. Ainsi, les découvertes automatiques de connaissances deviennent de plus en plus nécessaires dans les bases de données spatiales.

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) permet l'identification de classes, c'est à dire le regroupement des objets d'une base de données en sous-classes significatives. Cela permet des applications pratiques telle que le regroupement des classes de maisons le long d'une rivière lors de l'observation de la terre.

Toutefois, les applications aux bases de données spéciales conséquentes augmentent les exigences des algorithmes de clustering tel que :

- un minimum de connaissances sur les domaines afin de déterminer les paramètres d'entrée car les valeurs appropriées ne sont pas souvent connues à l'avance lorsque l'on travaille avec des bases de données importantes.
- la découverte de cluster de forme arbitraire car les formes de clusters dans les bases de données spatiales peuvent être sphérique, étiré, linéaire, allongé etc...
- une bonne efficacité sur les larges bases de données, c'est à dire celle contenant plus que quelques milliers d'objets.

De nombreux algorithmes de clustering ne permettent pas de résoudre ses problématiques. DBSCAN qui intègre une notion de cluster basée sur la densité permet de découvrir des clusters de forme arbitraire. Cet algorithme requiert seulement 2 paramètres d'entrée afin que l'utilisateur puisse

spécifier une valeur appropriée. DBSCAN se révèle être un algorithme particulièrement efficace et nous verrons qu'il bat les algorithmes antérieurs tels que CLARANS par un facteur de plus de 100 en terme d'efficacité.

3.2.2 Notion de Cluster basé sur la densité

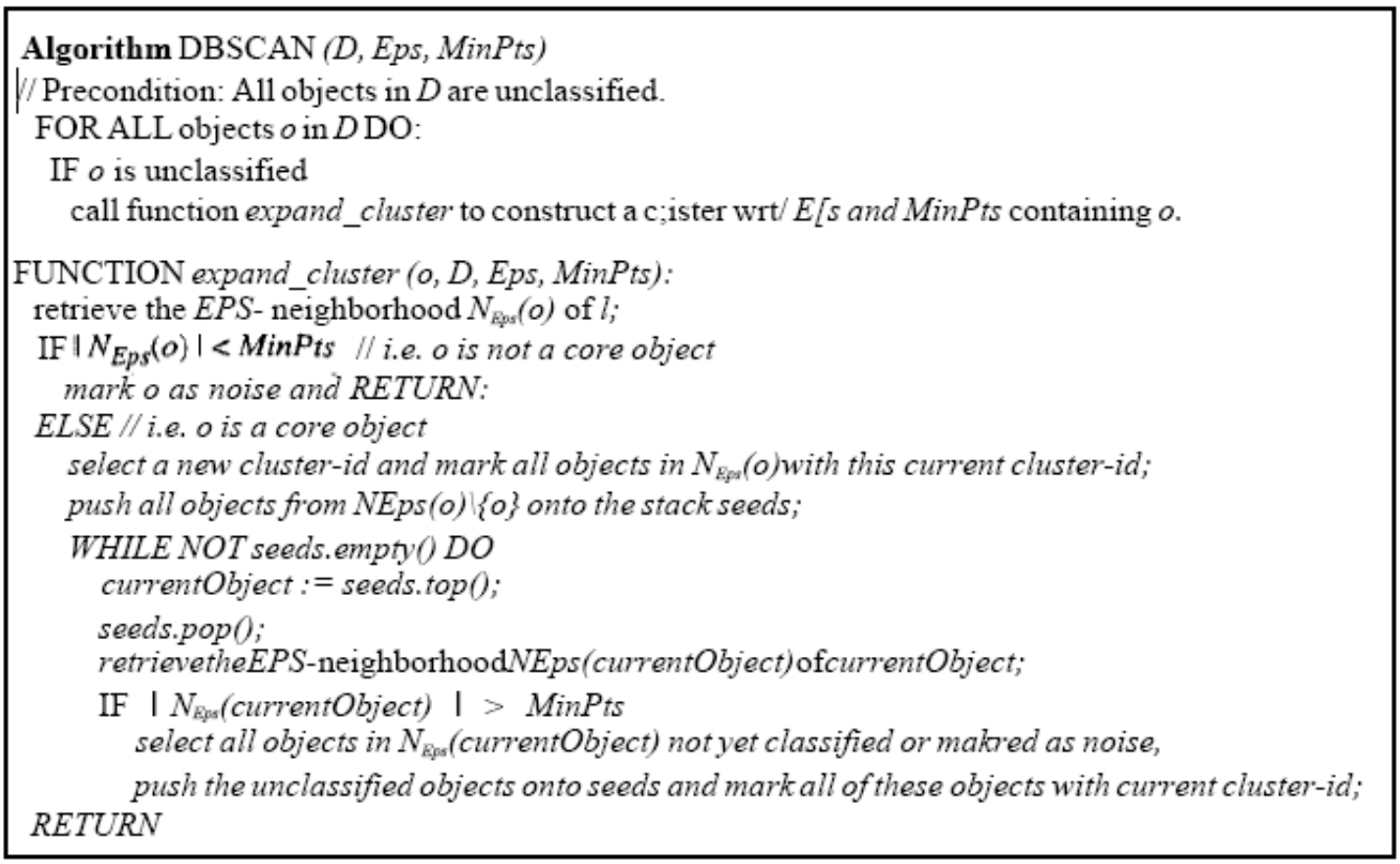
FIG. 3.3 – Une idée intuitive de la densité

Lorsque l'on regarde ces regroupements simples de points de la figure ci-dessus, il est possible de détecter facilement et sans aucune ambiguïté les points qui appartiennent à un cluster et ceux qui n'appartiennent à aucun et sont non significatifs (on parle de bruit). La principale raison qui nous permet de les reconnaître est qu'à l'extérieur des clusters la densité des zones de bruit est inférieure à celle de chacun des clusters.

Nous allons maintenant essayer de formaliser de façon intuitive la notion de cluster et de bruit dans une base de donnée D de points d'un espace S de k dimensions. Bien sûr, la notion de cluster de l'algorithme DBSCAN s'applique aussi bien dans un espace 2D, 3D euclidien ou aux espaces comportant de nombreuses dimensions. On fixe Eps le rayon du voisinage à étudier et MinPts le nombre minimum de points qui doivent être contenus dans le voisinage. L'idée clé du clustering basé sur la densité est que pour chaque point d'un cluster, ses environs pour un rayon donné Eps doit contenir un nombre minimum de points MinPts. Ainsi, le cardinale de son voisinage doit dépasser un certain seuil. Cette forme de voisinage est déterminée par le choix d'une fonction de distance de 2 points p et q, noté dist(p, q). Par exemple, pour un espace 2D, il s'agira d'un rectangle en utilisant une distance de Manhattan. Cette approche fonctionne quelle que soit la fonction de distance ce qui permet selon une application donnée de choisir une application appropriée. Afin qu'il soit facilement compréhensible tous nos exemples seront dans un espace 2D utilisant une distance euclidienne.

3.2.3 Algorithme DBSCAN

Dans cette partie nous allons présenter l'algorithme DBSCAN (Density-Based Spatial Clustering of Applications with Noise) dont le but est de découvrir les clusters et le bruit dans une base de donnée spatiale. Idéalement, nous devrions connaître les paramètres appropriés Eps et MinPts de chaque cluster et un point de chacun des clusters respectifs. Nous pouvons ensuite retrouver tous les points de densité accessibles des points donnés à partir de ces paramètres corrects. Mais il n'est pas facile d'obtenir ces informations à l'avance pour chaque cluster de la base de données. Pourtant il existe une heuristique simple et efficace (voir partie précédente) pour déterminer les paramètres Eps et MinPts des clusters les plus minces. Ces paramètres de densité sont des bons candidats pour les paramètres globaux spécifiant des densités les plus basses qui ne sont pas considérés comme du bruit.



Algorithm DBSCAN *(D, Eps, MinPts)*
// Precondition: All objects in *D* are unclassified.
FOR ALL objects *o* in *D* DO:
IF *o* is unclassified
call function *expand_cluster* to construct a c;ister wrt/ *E[s and MinPts* containing *o.*

FUNCTION *expand_cluster (o, D, Eps, MinPts):*
retrieve the *EPS*- neighborhood $N_{Eps}(o)$ of *l;*
IF $| N_{Eps}(o) | < MinPts$ *// i.e. o is not a core object*
mark o as noise and RETURN:
ELSE // i.e. o is a core object
select a new cluster-id and mark all objects in $N_{Eps}(o)$*with this current cluster-id;*
push all objects from NEps(o)\{o} onto the stack seeds;
WHILE NOT seeds.empty() DO
currentObject := seeds.top();
seeds.pop();
retrievetheEPS-neighborhood*NEps(currentObject)*of*currentObject;*
IF $| N_{Eps}(currentObject) | > MinPts$
select all objects in N_{Eps}*(currentObject) not yet classified or makred as noise,*
push the unclassified objects onto seeds and mark all of these objects with current cluster-id;
RETURN

FIG. 3.4 – Algorithme DBSCAN

Pour trouver un cluster, DBSCAN commence par un point arbitraire p et recherche tous les points de densité accessibles à partir de p. Si p est un point central, la procédure ajoute p au cluster. Si p est un point de bordure alors aucun point n'est atteignable à partir de p et DBSCAN visitera le prochain point de la base de donnée.

Grâce à l'utilisation des valeurs globale Eps et MinPts, DBSCAN peut fusionner 2 clusters dans le cas où 2 clusters de densité différente sont proches l'un de l'autre. Deux ensemble de point ayant au moins la densité la plus petite seront séparés l'un de l'autre si la distance entre les deux est plus large que Eps. En conséquence, un appel récursif de DBSCAN peut se révéler nécessaire pour les clusters détectés avec la plus haute valeur de MinPts. Cela n'est pas forcément un désavantage car l'application récursive de DBSCAN reste un algorithme basique, et n'est nécessaire que sous certaines conditions.

Ainsi, pour chaque objet que l'on ajoute, on a une zone de croissance qui va permettre d'entendre le cluster. Evidemment plus cette zone (une sphère) est grande et plus le cluster aura de chances de s'étendre. La notion de voisinage est la clé de cette méthode. On forme donc le cluster de proche en proche. La difficulté que nous pouvons rencontrer vient de la taille de la zone (rayon de la sphère) d'extension. DBSCAN a une complexité en (n * log n).

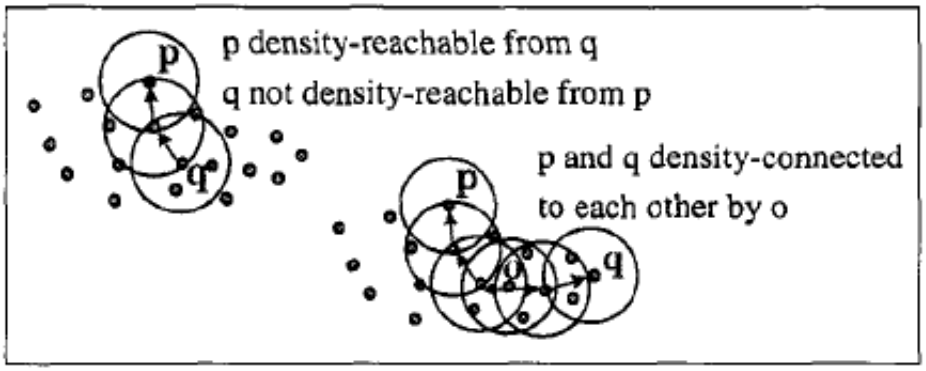


FIG. 3.5 – Notions de points dense-accesibles et de dense-connexité

3.2.4 Déterminer les paramètres Eps et MinPts

Dans cette partie, nous allons présenter une heuristique simple et efficace de déterminer les paramètres Eps et MinPts du plus petit cluster de la base de donnée.

Cette heuristique est basée sur les observations suivantes :

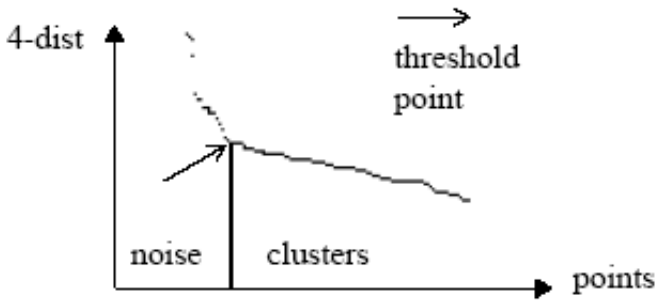


FIG. 3.6 – Heuristique de fixation des paramètres

En général, il peut être délicat de détecter la première vallée automatiquement, mais il est relativement simple pour l'utilisateur de voir cette vallée sur une représentation graphique. C'est pourquoi une approche interactive pour déterminer ce seuil est intéressante.

DBSCAN a besoin des paramètres EPS et MinPts. Les expériences ont montré que les graphes de distance k (k > 4) ne diffèrent pas vraiment des graphes de distance 4 mais nécessite des calculs bien plus important. Ainsi, nous éliminons le paramètre MinPts en le fixant à 4 pour toutes les bases de données (d'espace 2D).

Il s'agit ensuite d'utiliser une approche interactive pour déterminer le paramètre Eps de DBSCAN :

- Le système calcul et affiche le graphe de distance 4 pour la base de donnée
- Si l'utilisateur peut estimer le pourcentage de bruit, ce pourcentage est entré et le système en déduit une proposition pour le seuil de point.
- L'utilisateur peut accepter ou non le seuil proposé ou sélectionner un autre seuil qui est alors utilisé dans DBSCAN.

3.2.5 Evaluation de la performance de DBSCAN

Il est intéressant de comparer les performances de DBSCAN avec un algorithme antérieur, CLARANS qui était le premier et le seul conçu pour résoudre ce type de problème. 3 bases de données

différentes ont été utilisées pour comparer DBSCAN et CLARANS. Malgré le fait qu'ils s'agissent de 2 algorithmes de clustering de différents type, il n'y pas de mesure commune pour quantifier leur performance. Mais cela ne nous empêche pas de constater leurs différences visuellement sur des exemples simples.

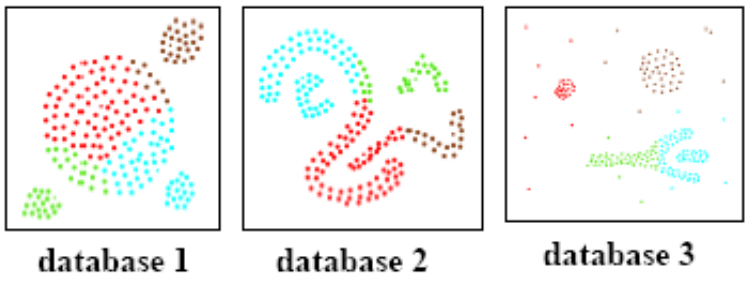


FIG. 3.7 – Clusters trouvés par CLARANS

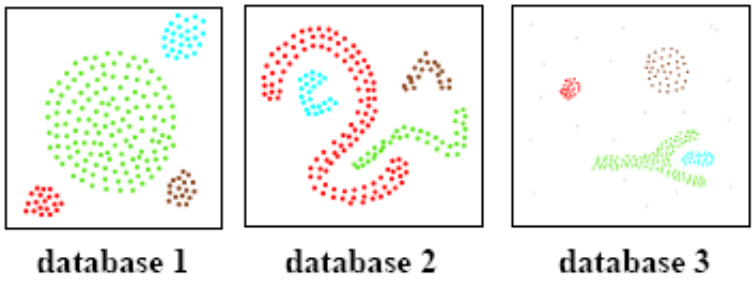


FIG. 3.8 – Clusters trouvés par DBSCAN

Nous visualisons les clusters détectés par des couleurs différentes. Sur les 2 premières bases de données, il n'y a pas de présence e bruit. Et sur la dernière le bruit à été fixé à 10%. Alors que DBSCAN découvre tous les clusters et détecte par conséquent les points de bruit, CLARANS découpe les clusters en plusieurs régions. De plus CLARANS n'a pas de gestion explicite du bruit car tous les points sont assignés à un cluster.

number of points	1252	2503	3910	5213	6256
DBSCAN	3.1	6.7	11.3	16.0	17.8
CLAR-ANS	758	3026	6845	11745	18029
number of points	7820	8937	10426	12512	
DBSCAN	24.5	28.2	32.7	41.7	
CLAR-ANS	29826	39265	60540	80638	

FIG. 3.9 – Temps d'exécution comparé en secondes

Les résultats obtenus montre que les performances de DBSCAN sont bien supérieures que CLARANS et que le temps de calcul est presque linéaire en fonction du nombre de point ce qui n'est pas le cas de CLARANS. Ainsi, il surperforme CLARANS par un facteur compris entre 250 et 7000 au fur et à mesure de l'augmentation du nombre de points.

3.3 Autres approches

Les approches basées sur la densité donnent de bons résultats dans le data-mining des vues satellite, la détection de zones dans l'espace même si elles sont concaves : clusters creux, clusters entourant un autre cluster, reconnaissance des sites actifs d'une protéine. Les performances de ces méthodes permettent aussi de travailler sur des ensembles d'objets bruités. D'autres algorithmes de la même famille existent, tels que DGLC, OPTICS, IncrementalDBSCAN.

Avec de très bonnes performances également, les algorithmes à base de grilles accélèrent le processus de requête. Citons : WaveCluster, DenClue, CLIQUE. On utilise un principe base sur une structure qui a une granularité de plusieurs finesses. On peut donc dire que les clusters sont formes en utilisant des cases pour découpes l'espace en boites, on essaie d'obtenir des boites de plus en plus grandes.

Il existe également des approches à base de modèle : pour trouver un cluster, on va cherche la forme qui va le mieux a la base des objets. On utilise donc des modèles : sphère, ellipse...

De plus, on peut constater que des algorithmes dérivés de DBSCAN ont vu le jour. Ainsi, GDBSCAN est une généralisation de DBSCAN, il permet une gestion plus complète dans l'espace. Il définit des conditions plus souples au niveau de la définition de voisinage et de la densité.

De même, afin de rendre l'algorithme utilisable pour les bases de données conséquentes, une adaptation parallèle à du être réalisée. Ainsi, depuis 1999, PDBSCAN permet de répondre à cette problématique. Le choix de la représentation des données ainsi que la centralisation des solutions des sous-problèmes locaux permet d'obtenir un DBSCAN distribué et performant pour les données très volumineuses.

Bibliographie

[1] Laurent Candillier. *Classification non supervisée contextualisée*, 2003.

[2] Ultra Fluide. *Tour d'horizon sur le datamining*, 2003. url : `http://www.ultra-fluide.com/agence-web/datamining.htm`.

[3] Giorgio Ingargiola. *Building Classification Models : ID3 and C4.5*, 1997. url : `http://www.cis.temple.edu/~ingargio/cis587/readings/id3-c45.html`.

[4] Mathieu Beal Paul Balez. Algorithmes de datamining. 2002.

[5] Jörg Sander Martin Ester Hans-Peter Kriegel Xiaowei Xu. *Density-Based Clustering in Spatial Databases : The Algorithm GDBSCAN and its Applications*.

[6] Martin Ester Hans-Peter Kriegel Jörg Sander Xiaowei Xu. *A Density-Based Algorithm for Discovering Cluster in Large Spatial Databases with Noise*, 1996.

[7] Xiaowey XU. *A Fast Parallel Clustering Algorithm for Large Spatial Databases*, 1999.