Tuareg : Classification non supervisée contextualisée

Laurent Candillier[1,2], Isabelle Tellier[1], Fabien Torre[1]

[1] GRAppA - Université Charles de Gaulle - Lille 3

[2] Pertinence Data Intelligence

Résumé : Cet article s'intéresse à la tâche de *clustering* de données numériques dans le cas particulier où toutes les dimensions de description des données ne sont pas également pertinentes pour le problème : certaines peuvent être tout simplement inutiles, d'autres n'être intéressantes que pour le rassemblement d'une partie des données, mais pas pour la totalité.

Les méthodes classiques se comportent mal sur ce type de problème et nous proposons pour l'aborder un algorithme original, **Tuareg**, basé sur un partitionnement élémentaire sur une dimension et sur une utilisation stochastique de cette opération élémentaire. Cet algorithme ne nécessite aucun réglage de paramètre de la part de l'utilisateur et est d'une complexité très raisonnable.

Des expériences, menées sur un problème classique en apprentissage non supervisé et sur des données artificielles, montrent que notre méthode a de bonnes capacités prédictives dans le cadre que nous nous sommes fixés et que, de plus, elle est capable de fournir une description intelligible de la solution découverte.

Introduction

L'objectif général de la *classification* est de pouvoir étiqueter des données en leur associant une classe. L'*apprentissage automatique* se propose de construire automatiquement une telle procédure de classifi cation en se basant sur des exemples, c'est-à-dire sur un ensemble limité de données disponibles. Si les classes possibles sont connues et si les exemples sont fournis avec l'étiquette de leur classe, on parle d'*apprentissage supervisé*. *A contrario*, nous nous plaçons pour cet article dans le cadre de l'*apprentissage non supervisé*, c'est-à-dire dans le cas où seuls des exemples sans étiquette sont disponibles et où les classes et leur nombre sont inconnus. L'apprentissage se ramène alors à regrouper les exemples de la manière la plus naturelle possible.

Cette volonté de *regrouper naturellement* est bien sûr ambiguë et le plus souvent formalisée par l'objectif de défi nir des groupes d'exemples tels que la distance entre exemples d'un même groupe soit minimale et que la distance entre groupes soit maximale (ces deux contraintes vont dans des sens opposés et c'est le meilleur compromis qui doit être trouvé). Cette vision de l'apprentissage non supervisé contraint donc à disposer d'une distance défi nie sur le langage de description des exemples. On se place ici dans le cas où l'espace de description des exemples est un espace vectoriel numérique (en pratique, $\mathbb{R}^n$) dans lequel chaque dimension correspond à un attribut distinct. Chaque exemple est donc décrit par un vecteur d'attributs à valeurs réelles.

À partir de ce paradigme, plusieurs familles de méthodes de *clustering* ont vu le jour : méthodes *hiérarchiques* (Fisher, 1987; Gennari *et al.*, 1989; Fisher, 1995; Guha *et al.*, 1998; Karypis *et al.*, 1999), méthodes basées sur les *K-moyennes* (Ng & Han, 1994; Zhang *et al.*, 1997), méthodes basées sur la *densité* (Ester *et al.*, 1996; Xu *et al.*, 1998; Hinneburg & Keim, 1998; Ankerst *et al.*, 1999), méthodes basées sur les *grilles* (Wang *et al.*, 1997; Sheikholeslami *et al.*, 2000; Brézellec & Didier, 2001), méthodes *statistiques* (Berkhin, 2002), méthodes basées sur la théorie des *graphes* (Hinneburg & Keim, 1999), méthodes basées sur la recherche *stochastique* (Jain *et al.*, 1999), ou méthodes basées sur les *réseaux de neurones* (Su & Chang, 2000).

Toutes ces méthodes et leurs performances sont fortement dépendantes de la distance utilisée. En pratique, il peut s'agir de la distance euclidienne ou, au mieux, d'une distance fournie par un expert du domaine. Celui-ci affecte alors un poids à chaque attribut, poids qui traduit l'importance de cet attribut pour le problème considéré.

Dans cet article, nous nous focalisons sur les problèmes où chaque groupe (ou *cluster*) à découvrir est caractérisé par certaines dimensions qui lui sont propres. Autrement dit, toutes les dimensions ne sont pas forcément utiles et ces dimensions pertinentes ne sont pas nécessairement les mêmes d'un groupe à l'autre. Ainsi, les distances globales que nous avons évoquées, qu'elles soient ajustées ou non par un expert, ne peuvent convenir ici : nous avons besoin d'une notion de proximité qui change avec le *contexte*, c'est-à-dire avec le groupe d'exemples considéré.

Prenons l'exemple fictif d'une base de patients victimes d'accidents cardio-vasculaires et décrits par différents attributs (pression artérielle, hérédité, antécédents, taux de cholestérol, etc.), chacun traduit par une dimension numérique positive.

Sur la figure 1, la projection sur trois de ces dimensions montre que trois groupes de patients peuvent être distingués : les patients ayant une hérédité à haut risque d'une part, ceux fumant beaucoup d'autre part, et enfin ceux ayant un taux de cholestérol élevé. C'est le type de description finale que nous voulons obtenir par apprentissage : nous voulons un algorithme qui *découvre des groupes aux dimensions spécifiques tout en ignorant les attributs non pertinents.*

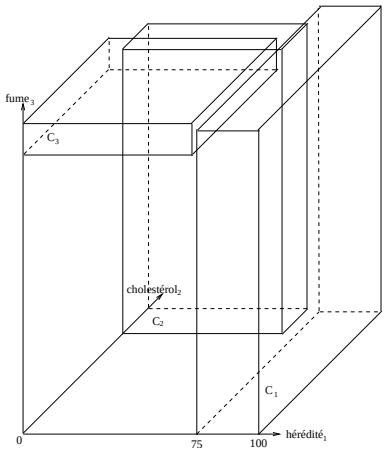


FIG. 1 – Exemple de base à partitionner

Plusieurs méthodes de clustering ont été récemment développées suivant cette idée, créant une nouvelle famille appelée *subspace clustering* (Agrawal *et al.*, 1998; Cheng *et al.*, 1999; Nagesh *et al.*, 1999; Aggarwal *et al.*, 1999; Aggarwal & Yu, 2000; Liu *et al.*, 2000; Woo & Lee, 2002; Yang *et al.*, 2002; Wang *et al.*, 2002; Procopiuc *et al.*, 2002; Cao & Wu, 2002; Yip *et al.*, 2003; Friedman & Meulman, 2004; Parsons *et al.*, 2004). Elles ont pour but d'identifier les sous-espaces de l'espace original de description des exemples, dans lesquels se trouvent des clusters denses. Les domaines d'application mis en avant sont l'indexation de bases de données OLAP, l'analyse de ventes ou l'analyse génétique, mais leurs expériences ont principalement été menées sur des données artificielles.

Nous proposons ici **Tuareg**, un algorithme original s'apparentant à cette famille, ne présentant pas certains défauts des algorithmes existant (pas de paramètre à régler par l'utilisateur, pas d'hypothèse sur le nombre de clusters attendus) et ayant des propriétés intéressantes (bonne complexité et intelligibilité du résultat produit).

L'article est organisé comme suit : la section 1 décrit **Tuareg**, notre algorithme de *classification non supervisée contextualisée* ; la section 2 détaille les résultats de nos expérimentations ; enfi n, nous discutons dans la section 3 les avantages et limites de notre approche.

1 Présentation de l'algorithme Tuareg

Tuareg est un algorithme descendant, son principe général étant de fractionner successivement l'ensemble des exemples. Dans l'esprit, l'approche est comparable à celle de C4.5 en apprentissage supervisé : il s'agit de repérer à chaque étape la dimension permettant le partitionnement le plus pertinent pour le groupe d'exemples considéré, et d'itérer ce processus jusqu'à ce que plus aucun groupe ne soit amélioré par découpage.

Chaque dimension de description est considérée l'une après l'autre et indépendamment des autres. La brique de base est donc un algorithme de partitionnement sur une dimension. C'est ce sujet que nous traitons d'abord, après avoir introduit les défi nitions et notations nécessaires. La suite de la présentation expose comment ce partitionnement élémentaire est intégré dans une stratégie globale visant à identifi er, pour chaque groupe, ses dimensions pertinentes, et comment un regroupement complet des exemples est fi nalement constitué.

1.1 Définitions et notations

Dans un espace S à p dimensions numériques, soit un ensemble E de N objets $\overrightarrow{X_1}, ..., \overrightarrow{X_N}$. Pour tout $i \in \{1, ..., N\}$, l'objet $\overrightarrow{X_i}$ est défi ni par ses coordonnées dans S : $(X_{i1}, ..., X_{ip})$. Notons que certains objets peuvent être dupliqués en plusieurs exemplaires, et dans ce cas chaque instance différente est considérée comme distincte. Nous devrions donc parler de multi-ensembles plutôt que d'ensembles. Mais par souci de simplifi cation nous continuerons à utiliser les notations ensemblistes traditionnelles pour désigner et manipuler les clusters.

On se donne deux vues sur un cluster C : d'une part $E(C)$ le sous-ensemble des objets de E contenus dans C, et d'autre part un ensemble $D(C)$ d'intervalles défi nis chacun sur une dimension. On note $DimSet(C)$ le sous-ensemble des dimensions de S sur lesquelles les intervalles de $D(C)$ sont défi nis. $DimSet(C)$ contient les *dimensions caractéristiques* du cluster C, et correspond à l'*ensemble minimal* des dimensions supports des intervalles qui caractérisent l'appartenance d'un objet à ce cluster.

- $N(C) = |E(C)|$ représente le nombre d'objets contenus dans C,
- $ND(C) = |DimSet(C)|$ le nombre de ses dimensions caractéristiques,
- et pour toute dimension caractéristique $d \in DimSet(C)$, l'intervalle de défi nition $D_d(C) \in D(C)$ du cluster C est donné par $D_d(C) = [Min_d(C), Max_d(C)]$, avec $Min_d(C) = Min_{\{i|\overrightarrow{X_i} \in E(C)\}} X_{id}$ et $Max_d(C) = Max_{\{i|\overrightarrow{X_i} \in E(C)\}} X_{id}$.

Par exemple, pour le cluster C_1 de la base fi ctive de la fi gure 1, on aura $ND(C_1) = 1$, $DimSet(C_1) = \{\texttt{hérédité}_1\}$, et $D_1(C_1) = [75, 100]$.

1.2 Partitionnement sur une dimension

Nous nous plaçons ici dans le cadre où l'on cherche à partitionner un ensemble de n éléments $E = \{x_j | 1 \leq j \leq n\}$, les valeurs x_j étant des valeurs numériques rangées par ordre croissant.

Notre méthode de partitionnement sur une dimension, nommée `Clustering1D`, est très simple : elle consiste à diviser E en deux, en coupant entre les deux valeurs consécutives séparées par la plus grande distance, ce qui revient à chercher :

$$Max_j\{x_{j+1} - x_j \quad | \quad 1 \leq j \leq n-1\}$$

En fait, diverses alternatives ont été envisagées et testées pour réaliser ce partitionnement sur une dimension. Ainsi, nous avons d'abord envisagé l'utilisation de l'algorithme K-means (Diday *et al.*, 1982b) et de l'algorithme optimal de Fisher (Diday *et al.*, 1982a). Nous avons également envisagé différentes mesures à optimiser, parmi lesquelles des mesures mixant distances inter-clusters et distances intra-clusters, au lieu de simplement optimiser la distance inter-clusters. Mais les tests ont mis en avant le fait qu'en plus d'être bien plus rapide, l'algorithme élémentaire utilisé est suffisamment pertinent dans notre cadre.

1.3 Mesures et sélection de la division à effectuer

Nous disposons d'un algorithme de partitionnement sur une dimension. Cet algorithme peut être utilisé sur n'importe quel cluster déjà formé et, pour chacun d'eux, sur n'importe quelle dimension sur laquelle on projette les coordonnées de ses objets.

Nous devons donc maintenant être capable de choisir la meilleure division parmi toutes les candidates. Pour cela, deux considérations sont à prendre en compte :

1. est-ce que l'espace laissé entre les deux nouveaux clusters est significatif ?
2. les nouveaux clusters formés sont-ils plus intéressants que le cluster de départ ?

On retrouve le compromis classique entre l'optimisation des distances inter-clusters (critère 1) et celle des distances intra-clusters (critère 2), mais simplifié dans notre cas puisque les distances sont ramenées à une seule dimension.

Soit $P_d(C) = \{c_1, c_2\}$ la partition obtenue par la méthode `Clustering1D` à laquelle on a fourni en entrée l'ensemble $C[d]$ des coordonnées sur la dimension d des objets appartenant au cluster initial C ; c_1 et c_2 sont les deux intervalles créés sur la dimension d ; enfin, nous notons $\texttt{Inter}_d(C) = Min(c_2) - Max(c_1)$ la distance séparant ces deux intervalles.

Pour évaluer le premier critère, la mesure utilisée est tout simplement la distance $\texttt{Inter}_d(C)$, normalisée par $|D_d(C)|$[1] pour permettre de comparer des divisions sur des dimensions de tailles différentes, et pondérée par le nombre d'objets du cluster initial C pour favoriser la division des clusters contenant le plus d'objets.

$$\texttt{Interet}(P_d(C)) = N(C) \times \frac{\texttt{Inter}_d(C)}{|D_d(C)|}$$

Plus cette quantité est grande, plus la division correspondante est jugée intéressante.

[1] on ne considère que l'espace dans lequel C se trouve, et non pas l'espace complet de description S.

L'idée du second critère est de refuser une division si celle-ci marque une baisse de l'homogénéité du cluster sur ses dimensions caractéristiques. Cette homogénéité est évaluée par l'inertie moyenne du cluster sur ses dimensions caractéristiques.

Nous nommons `AcceptDivision` la fonction retournant s'il faut accepter ou non une division donnée $P_d(C)$ (algorithme 1).

Algorithme 1 AcceptDivision

Entrée : $P_d(C) = \{c_1, c_2\}$ la division proposée du cluster C sur la dimension d.
  for all $j \in DimSet(C)$ **do**
    $G_j(C) = \frac{\sum_{\{i|\overrightarrow{X_i} \in E(C)\}} X_{ij}}{N(C)}$ {centre de gravité du cluster C sur la dimension j}
    $\texttt{Inertie}_j(C) = \frac{\sum_{\{i|\overrightarrow{X_i} \in E(C)\}} (X_{ij} - G_j(C))^2}{|D_j(C)|}$ {inertie normalisée de C sur j}
  end for
  $\texttt{Inertie}(C) = \frac{\sum_{j \in DimSet(C)} \texttt{Inertie}_j(C)}{ND(C)}$ {inertie de C sur $DimSet(C)$}
  for $i = 1$ to 2 **do**
    $G_i = \frac{\sum_{\{j|x_j \in c_i\}} x_j}{|\{j|x_j \in c_i\}|}$ {centre de gravité de l'ensemble des valeurs de c_i}
    $I_i = \sum_{\{j|x_j \in c_i\}} (x_j - G_i)^2$ {inertie de c_i}
  end for
  $\texttt{minInertie} = \frac{Min(I_1, I_2)}{|D_d(C)|}$ {inertie normalisée minimale entre c_1 et c_2}
  Return ($\texttt{minInertie} < \texttt{Inertie}(C)$)
Sortie : 1 si la division est acceptée ; 0 sinon.

Ainsi, on choisit de ne considérer que les divisions pour lesquelles la plus petite inertie sur la dimension de division est inférieure à l'inertie du cluster considéré. On peut aussi choisir de ne considérer que les divisions pour lesquelles la plus grande inertie sur la dimension de division est inférieure à l'inertie du cluster considéré (ce qui revient à remplacer Min par Max dans l'algorithme 1). Nous verrons dans la partie expérimentation les différences que ce choix entraîne quant au nombre de clusters fi naux générés.

Finalement, la méthode de sélection de la dimension de division suivante à considérer, étant donnée la partition courante, est fournie par l'algorithme 2.

1.4 Algorithme de base : divisions successives

À ce niveau, nous disposons d'un algorithme qui, étant donnée une partition courante, désigne la division suivante à effectuer ou indique que plus aucune division n'est pertinente. Nous en déduisons naturellement la méthode de génération de nouveaux clusters à partir d'une division sélectionnée $P_d(C) = \{c_1, c_2\}$ du cluster C sur la dimension d.

Soit $\texttt{SubCluster}(C, d, c_i)$ la fonction qui retourne le cluster C'_i :

- dont la vue $E(C'_i)$ sur les objets de E inclut les éléments de $E(C)$ dont les coordonnées sur la dimension d appartiennent à c_i : $E(C'_i) = \{\overrightarrow{X_j} \in E(C) | X_{jd} \in c_i\}$,
- et dont la deuxième vue sur les dimensions de S comprend les dimensions caractéristiques de C plus la dimension d : $DimSet(C'_i) = DimSet(C) \cup d$. Mais il peut arriver qu'en

Algorithme 2 DivSelect

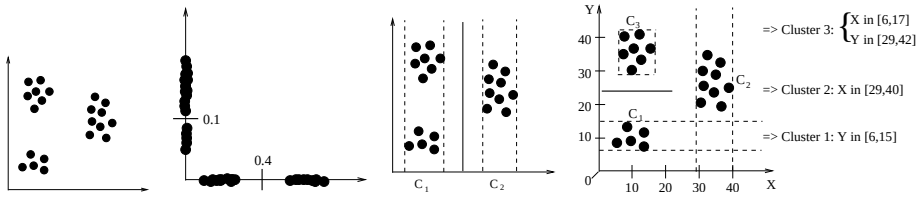
Entrée : $P = (C_1, ..., C_K)$ une partition composée de K clusters.
  `Best` $=$ `(0,0,'none')` {la meilleure division courante}
  `BestNote` $= 0$ {la mesure d'intérêt associée à la meilleure division}
  for $k = 1$ to K **do**
    for $d = 1$ to p **do**
      $\pi =$ `Clustering1D`$(C_k[d])$; `Note` $=$ `Interet`(π)
      if (`Note` $>$ `BestNote`) **and** (`AcceptDivision`(π)) **then**
        `BestNote` $=$ `Note`; `Best` $= (k, d, \pi)$
      end if
    end for
  end for
Sortie : `Best` : k le numéro du cluster à diviser, d la dimension sur laquelle effectuer la division, et $\pi = P_d(C_k)$ la division à effectuer sur cette dimension.

ajoutant à un cluster C'_i une nouvelle dimension caractéristique, une autre dimension caractéristique de ce cluster $e \in DimSet(C'_i)$ ne soit plus pertinente : ainsi, si aucun objet $\overrightarrow{X_j} \in E \setminus E(C'_i)$ n'est tel que pour toute dimension $d \in DimSet(C'_i) \setminus e$, X_{jd} appartient à $D_d(C'_i)$, alors on supprime la dimension caractéristique e de $DimSet(C'_i)$. On obtient ainsi une *description minimale* du cluster C'_i, en ciblant l'ensemble minimal des dimensions supports des intervalles qui caractérisent l'appartenance d'un objet à ce cluster.

Finalement, la méthode itérative de base est présentée à l'algorithme 3. Le seul paramètre de cet algorithme est `MAX_CLUSTER`, et correspond au nombre maximum de clusters générés. Il ne s'agit pas du nombre attendu de clusters mais d'une borne supérieure qui n'est pas nécessairement atteinte. Le résultat de cet algorithme sur un exemple élémentaire est donné par la figure 2.

Algorithme 3 Divisif

Entrée : $E = (\overrightarrow{X_1}, ..., \overrightarrow{X_N})$ un ensemble d'objets.
  $K = 1$; $P[1] = (E, \emptyset)$ {P la partition courante composée de K clusters et initialisée avec un cluster contenant tous les objets de E et aucune dimension caractéristique}
  $(k, d, \pi) =$ `DivSelect(P)`
  while ($K <$ `MAX_CLUSTER`) **and** ($\pi \neq \emptyset$) **do**
    $K = K + 1$
    $P[K] =$ `SubCluster`$(P[k], d, c_1)$
    $P[k] =$ `SubCluster`$(P[k], d, c_2)$
    $(k, d, \pi) =$ `DivSelect(P)`
  end while
Sortie : P la partition créée de E, composée de K clusters.



(a) Base d'exemples

(b) Phase 1 : Clustering1D sur chaque dimension et mesures d'intérêt associées

(c) Phase 2 : division sur la dimension d'intérêt maximum

(d) Itérations : divisions tant que possible, et description minimale des clusters

FIG. 2 – Exemple d'exécution de l'algorithme `Divisif`

1.5 Algorithme final : Tuareg

La version de base de l'algorithme 2 se contente de rechercher le meilleur découpage possible sur une dimension de l'un des clusters courants. Or, il est souvent plus effi cace de conserver les k meilleurs choix possibles lors d'une recherche descendante (principe de la recherche en faisceau ou *beam search*). De plus, nous voulons introduire une certaine souplesse dans l'algorithme, en ne requérant pas qu'il effectue un partitionnement strict de l'ensemble des données de départ, mais qu'il constitue plutôt un ensemble de clusters possibles homogènes (qui peuvent éventuellement avoir des intersections non vides). Pour satisfaire ces deux exigences tout en conservant un algorithme effi cace, nous proposons d'introduire une dimension stochastique dans l'algorithme 2. L'idée est donc de remplacer la sélection du meilleur découpage possible à chaque étape par un tirage au sort (avec des probabilités proportionnelles à leur mesure d'intérêt) parmi les k (nommé `ALEA_DIV` par la suite) meilleurs découpages possibles. Cette idée conduit à remplacer l'algorithme 2 par l'algorithme 4.

De manière générale, la dimension stochastique de la méthode a pour but de limiter les biais possibles dûs à la simplicité de notre méthode de partitionnement sur une dimension et dûs aux choix des mesures d'intérêts effectués. Elle permet ainsi de simuler, pour un coût algorithmique raisonnable, la recherche parallèle de solutions alternatives à chaque fois qu'un choix de partitionnement est effectué. Après plusieurs exécutions de l'algorithme de base, l'idée est ensuite de sélectionner parmi tous les clusters générés ceux qui sont les plus denses sur leurs dimensions caractéristiques, et couvrant le plus d'objets non encore couverts (fonction `FinalSelect` formalisée par l'algorithme 5).

Finalement, **Tuareg**, notre algorithme de clustering contextualisé est présenté à l'algorithme 6 et consiste :

1. à exécuter l'algorithme de base `Divisif` un certain nombre de fois (`NB_RUNS`), en faisant un tirage aléatoire parmi les meilleures divisions proposées ;
2. à sélectionner dans l'ensemble des clusters résultants le sous-ensemble de ceux qui sont

Algorithme 4 DivSelect (version 2, stochastique)

Entrée : $P = (C_1, ..., C_K)$ une partition composée de K clusters.
`Bests`$[1..$`ALEA_DIV`$]$ $\leftarrow$ initialiser à $(0, 0,$ `'none'`$, 0)$ {tableau stockant les `ALEA_DIV` divisions dont la mesure d'intérêt associée est optimale}
`MinBest` $= 0$ {note minimale associée aux divisions stockées dans `Bests`}
for $k = 1$ to K **do**
  for $d = 1$ to p **do**
    $\pi =$ `Clustering1D`$(C_k[d])$; `Note` $=$ `Interet`(π)
    if (`Note` $>$ `MinBest`) **then**
      Cibler l'indice i de l'emplacement, dans `Bests`, de la division de note minimale ; `Bests`$[i] = (k, d, \pi,$ `Note`$)$; Mettre à jour `MinBest`
    end if
  end for
end for
for $i = 1$ to `ALEA_DIV` **do**
  if (`Not AcceptDivision(Bests[i])`) **then**
    Supprimer le i^{eme} élément du tableau `Bests`
  end if
end for
Retourner aléatoirement l'une des divisions stockées dans `Bests`, avec la probabilité de tirage d'une division proportionnelle à sa mesure d'intérêt associée

Sortie : k le numéro du cluster à diviser, d la dimension sur laquelle effectuer la division, et $\pi = P_d(C_k)$ la division à effectuer sur cette dimension.

les plus denses sur leurs dimensions caractéristiques et couvrent l'ensemble des objets de la base (algorithme 5).

Il nécessite la donnée de trois paramètres en entrée : `ALEA_DIV` et `NB_RUNS`, utilisés pour gérer la composante stochastique de notre algorithme, et `MAX_CLUSTER`, borne supérieure sur le nombre total de clusters générés pour une itération de la méthode. Pour ce qui concerne les deux premiers paramètres, nous verrons que nous pouvons leur fixer expérimentalement une valeur par défaut indépendante du problème.

La complexité dans le pire des cas de **Tuareg** est en $O(p.N.log(N))$. La complexité en $O(N.log(N))$ selon le nombre d'objets N vient de l'algorithme `Clustering1D`, nécessitant l'ordonnancement des valeurs sur une dimension pour cibler la distance maximale entre valeurs consécutives. Et la complexité linéaire en nombre de dimensions p provient de l'algorithme `DivSelect`, qui nécessite d'exécuter l'algorithme `Clustering1D` sur chaque dimension de l'espace. Quant aux autres algorithmes, leur complexité est inférieure, car bornée par des constantes dépendantes des paramètres `MAX_CLUSTER` et `NB_RUNS`, supposés négligeables devant p et N.

Algorithme 5 FinalSelect

Entrée : $E = (\overrightarrow{X_1}, ..., \overrightarrow{X_N})$ un ensemble d'objets, et SET, un ensemble de clusters candidats pouvant se recouvrir.
NbUncover $= N$ {nombre d'objets non couverts par les clusters sélectionnés}
Uncover $= \{\overrightarrow{X_1}, ..., \overrightarrow{X_N}\}$ {liste des objets non couverts}
$P = \emptyset$ {la partition résultante}
while (NbUncover > 0) **do**
Best = 'none' {cluster courant le plus dense}
BestDensite $= 0$ {meilleure densité courante}
NbCover $= 0$ {nombre d'objets de Best non couverts}
for all $C \in$ SET **do**
$Ncover = |E(C) \cap$ Uncover$|$ {nombre d'objets de C non couverts}
$Volume = \sqrt[ND(C)]{\prod_{d \in DimSet(C)} |D_d(C)|}$ {moyenne géométrique des tailles d'intervalles de défi nition de C sur ses dimensions caractéristiques (moyenne utilisée pour permettre que les volumes de clusters défi nis sur différents nombres de dimensions caractéristiques soient comparables)}
$Densite = Ncover/Volume$
if ($Densite >$ BestDensite) **then**
Best $= C$; BestDensite $= Densite$; NbCover $= Ncover$
end if
end for
$P = P \cup$ Best ; SET $=$ SET $\setminus$ Best
NbUncover $=$ NbUncover $-$ NbCover ; Uncover $=$ Uncover $\setminus E($Best$)$
end while
for all $C_i \in P$ **do**
for all $C_j \in P$ **do**
if $E(C_i) \subset E(C_j)$ **then**
Supprimer C_i de P {cluster inclus dans un autre}
end if
end for
end for
Sortie : $P = (C_1, ..., C_K)$ la partition sélectionnée, composée de K clusters.

Algorithme 6 Tuareg

Entrée : $E = (\overrightarrow{X_1}, ..., \overrightarrow{X_N})$ un ensemble d'objets.
SET $= \emptyset$
for $n = 1$ to NB_RUNS **do**
$P =$ Divisif(E)
SET $=$ SET $\cup P$
end for
Return FinalSelect(SET)
Sortie : $P = (C_1, ..., C_K)$ la partition créée de E, composée de K clusters.

2 Expérimentations

Des expériences ont été menées sur un problème classique en apprentissage non supervisé, issu des bases de données de l'UCI (Blake & Merz, 1998), ainsi que sur des données générées artificiellement. La première base permet d'illustrer les résultats de **Tuareg** en classification non supervisée, les classes à trouver étant connues. Les bases artificielles serviront à évaluer **Tuareg** sur différents aspects :

1. quelle est l'influence des paramètres de **Tuareg** ?
2. **Tuareg** est-il robuste relativement au nombres d'objets, au nombre de dimensions, et au nombre et caractéristiques des clusters de la base de données ?
3. **Tuareg** résiste-t-il à l'ajout de *dimensions non pertinentes* sur lesquelles les coordonnées des objets sont générées aléatoirement ?

Dans un premier temps, nous avons choisi de nous comparer à l'algorithme *K-means* (Diday *et al.*, 1982b) : dans chaque test, le nombre de clusters à trouver lui est fourni, les centroïdes sont initialisés de façon aléatoire, la méthode est lancée dix fois, et le résultat minimisant l'inertie globale de la partition est conservé.

2.1 Base des quadrupèdes

Des tests ont été effectués sur une base connue en classification non supervisée, issue des bases de données de l'UCI : les *quadrupèdes*. Il s'agit d'une base dans laquelle sont mélangés des chevaux, girafes, chiens et chats, tous décrits par les mesures de leurs membres (tête, cou, torse, queue, et 4 pattes), chaque membre étant vu comme un cylindre caractérisé par 9 attributs : au total, un exemple est donc décrit par 72 attributs.

Lorsque celle-ci contient plus de 70 animaux, **Tuareg** obtient les résultats suivants : en plus de retrouver les quatre groupes pertinents (et ce, sans connaître le nombre de groupes recherchés), **Tuareg** caractérise chaque groupe par des dimensions qui lui sont propres : les girafes sont caractérisées par leurs très longues jambes, les chevaux par leurs jambes de taille moyenne, les chiens la petite taille de leur queue, et les chats la petite taille de leur torse. Nous obtenons bien un ensemble de *groupes aux dimensions spécifiques*, et ce résultat reste stable à l'ajout de dimensions sur lesquelles les coordonnées de tous les objets sont tirées aléatoirement dans [0..100]. Ainsi, alors que K-means se dégrade dès l'ajout d'une dimension aléatoire, **Tuareg** continue à retrouver les quatre groupes pertinents même après l'ajout de 1000 dimensions aléatoires.

À titre indicatif, le temps d'exécution moyen de **Tuareg** sur une base de 200 quadrupèdes est de 10 secondes sur un pentium 4 à 2.5 Ghz, et de 25 secondes pour K-means.

2.2 Données artificielles

Pour tester plus avant **Tuareg**, nous produisons automatiquement des problèmes, un problème étant caractérisé par les paramètres suivants : N le nombre d'objets de la base, p le nombre de dimensions décrivant les objets, K le nombre de clusters, c le nombre moyen de dimensions caractéristiques des clusters, $ecar$ l'écart type des coordonnées des objets appartenant à un même cluster par rapport à son centre de gravité et sur ses dimensions caractéristiques, $space$ l'espace

minimum isolant les clusters sur leurs dimensions caractéristiques, et d le nombre de dimensions aléatoires supplémentaires.

K points d'ancrage $(\overrightarrow{O_1}, ..., \overrightarrow{O_K})$ sont tirés aléatoirement dans l'espace de description $[0..100]^p$, et sont utilisés comme centroïdes des clusters $(C_1, ..., C_K)$ à générer. À chacun de ces clusters est associée une partie des N objets, et un sous-ensemble des p dimensions constituant ses dimensions caractéristiques. Puis les coordonnées des objets appartenant à un cluster C_i sont générées selon une loi normale de centre O_{id} et d'écart type $ecar$, sur toute dimension caractéristique d de $DimSet(C_i)$; elles sont générées selon une loi uniforme sur ses dimensions non caractéristiques : dans l'intervalle $[0..100]$ si la dimension n'est caractéristique pour aucun cluster, et hors des *zones caractéristiques* des autres clusters sinon ; on appelle *zone caractéristique* d'un cluster de point d'ancrage $\overrightarrow{X_0}$ sur une dimension d l'intervalle $[X_{0d}-ecar-space, X_{0d}+ecar+space]$; enfi n, les coordonnées des objets sont générées selon une loi uniforme dans l'intervalle $[0..100]$ sur les d dimensions aléatoires supplémentaires.

2.3 Protocole d'évaluation

Nous utilisons ces bases artifi cielles pour évaluer les taux d'erreurs de **Tuareg** et K-means en classifi cation. Pour toute base de données disponible, 70% des données forment l'ensemble E utilisé pour l'apprentissage (le partitionnement de la base), et 30% l'ensemble T utilisé pour tester la classifi cation de nouveaux exemples (leur association à un cluster créé).

Pour chaque nouvel objet $\overrightarrow{X_i}$ à classer, la méthode adoptée est de l'associer au cluster dont il est le plus proche sur l'ensemble des dimensions caractéristiques de celui-ci, c'est-à-dire le cluster C qui minimise la mesure $Proximite$ suivante :

$$Proximite(\overrightarrow{X_i}, C) = \frac{\sum_{d \in DimSet(C)} proximite(X_{id}, D_d(C))}{ND(C)}$$

avec : $proximite(X_{id}, D_d(C)) = \begin{cases} 0 \text{ si } X_{id} \in D_d(C) \\ min(|X_{id} - Min_d(C)|, |X_{id} - Max_d(C)|) \text{ sinon.} \end{cases}$

Pour la méthode K-means, un nouvel objet est associé au cluster dont le centroïde est le plus proche, en utilisant la distance euclidienne.

Étant donnés, pour toute classe réelle i et tout cluster créé C_k, n_i le nombre d'éléments de la classe i dans T, Nb_k le nombre d'éléments du cluster C_k dans T, et N_{ik} le nombre d'éléments de la classe i appartenant au cluster C_k, deux mesures sont alors classiquement utilisées pour évaluer les taux d'erreurs en classifi cation des méthodes :

1. la F-mesure, qui nécessite de calculer les quantités suivantes :
   $Rappel(i,k) = N_{ik}/n_i$ et $Precision(i,k) = N_{ik}/Nb_k$, puis
   $F(i,k) = \frac{2 \times Rappel(i,k) \times Precision(i,k)}{Rappel(i,k)+Precision(i,k)}$; et donnée par $F = \sum_i \frac{n_i}{|T|} max_k\{F(i,k)\}$;
2. l'Entropie : $Ent_k = -\sum_i Precision(i,k) \times log(Precision(i,k))$,
   et $Ent = \sum_k \frac{Nb_k \times Ent_k}{|T|}$.

L'Entropie mesure la pureté des clusters produits. La F-mesure, elle, prend également en compte le nombre de clusters générés, comparé au nombre réel de classes de la base.

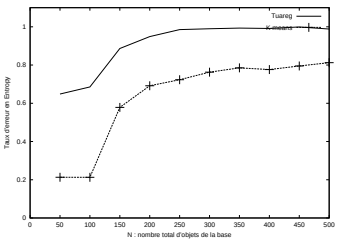
2.4 Résultats

Dans un premier temps, nous utilisons ces bases de données artificielles pour tester l'influence des paramètres de **Tuareg** sur ses performances. Les données sont générées avec comme paramètres : $N = 500$, $p = 10$, $K = 5$, $c = 2$, $ecar = 3$, $space = 10$, et $d = 10$, et chaque valeur mesurée correspond à une moyenne sur 100 tests.

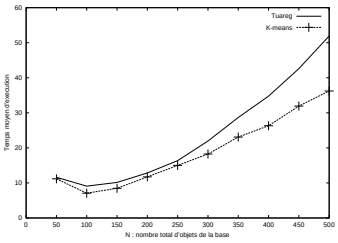
Les expériences ont mis en avant le fait que les performances de **Tuareg** sont améliorées lorsqu'`ALEA_DIV` passe de 1 à 2 (c'est-à-dire que la composante stochastique de **Tuareg** améliore ses résultats), puis qu'elles restent stables quand on continue à incrémenter sa valeur. Nous avons donc fixé `ALEA_DIV` à 2 par la suite. Nous avons de même fixé `NB_RUNS` à 7 expérimentalement.

Concernant `MAX_CLUSTER`, son influence dépend de la méthode choisie pour la fonction `AcceptDivision`. Si celle-ci est basée sur l'inertie minimale, l'algorithme produit un nombre de clusters proche de `MAX_CLUSTER`, ce qui donne de meilleurs résultats en classification. Si elle est basée sur l'inertie maximale, l'algorithme génère un nombre de clusters proche du nombre réel de clusters de la base (ce nombre plafonne à 7 dans nos expériences, même quand `MAX_CLUSTER` dépasse les 20) et a donc de meilleures propriétés explicatives, mais elle est moins performante en classification. Dans les deux cas, `MAX_CLUSTER` constitue une borne supérieure qui permet d'assurer de ne pas générer trop de clusters. Nous l'avons fixé à 10 dans la suite.

Les courbes de la figure 3 mettent en avant la robustesse de **Tuareg** face au nombre d'objets de la base fournie en entrée. On observe sur la première qu'à partir d'un certain nombre d'éléments (200), les performances de **Tuareg** se stabilisent au dessus de celles de K-means, et proches de 1. La seconde courbe montre que le temps d'exécution de **Tuareg** reste raisonnable, même pour des bases contenant beaucoup d'objets.



(a) Entropie : **Tuareg** surpasse K-means, et est proche de 1 quand $N > 200$



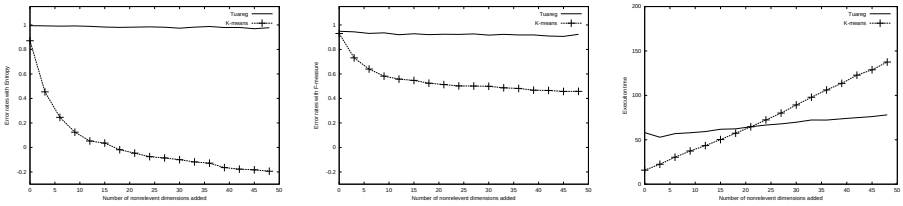
(b) Temps d'exécution : l'évolution pour **Tuareg** suit celle de K-means

FIG. 3 – Robustesse face au nombre d'objets de la base $N \in [50..500]$

D'autres tests ont été menés pour observer l'évolution des performances de **Tuareg** selon les caractéristiques de la base de données qu'on lui fournit en entrée. Ainsi, il a été mis en avant que **Tuareg** reste stable face à p le nombre de dimensions de la base, à K le nombre

de clusters (tant qu'il n'est pas trop proche de MAX_CLUSTER), et à c le nombre moyen de dimensions caractéristiques des clusters. Par contre, ses performances commencent à chuter lorsque $ecar$ devient trop important (à partir de 5), ou lorsque $space$ devient trop faible (à partir de 7). Elles restent cependant bonnes (autour de 0.8 quand $space = 0$ ou $ecar = 10$), parce qu'en produisant les exemples d'un cluster suivant une loi normale centrée sur son centroïde, on rend plus probable la présence d'écarts importants aux marges de ce cluster plutôt qu'en son sein.

En particulier, et contrairement à K-means, **Tuareg** reste stable face à des dimensions non pertinentes ajoutées aléatoirement. Ce résultat est illustré par les courbes de la fi gure 4, représentant les taux d'erreurs en classifi cation et le temps d'exécution des méthodes. De plus, il apparaît que lorsque le nombre de dimensions devient important, **Tuareg** devient plus rapide que K-means, pourtant connu pour son effi cacité.



(a) Entropie : **Tuareg** reste stable, contrairement à K-means

(b) F-mesure : **Tuareg** reste stable, contrairement à K-means

(c) Temps d'exécution : **Tuareg** devient plus rapide que K-means quand $d > 20$

FIG. 4 – Résistance aux dimensions non pertinentes $d \in [0..50]$

3 Conclusions et perspectives

Tuareg se distingue de ses concurrents par un certain nombre de caractéristiques. La principale est qu'il produit un ensemble de clusters hypercubiques défi nis par des intervalles sur un *nombre restreint de dimensions caractéristiques*. Il ne réalise pas une partition de l'ensemble des données puisque les clusters fi naux peuvent se chevaucher partiellement. Pour évaluer plus avant **Tuareg**, il nous faut maintenant poursuivre nos expérimentations et le comparer aux autres algorithmes existant de *subspace clustering*.

L'hypothèse fondamentale sur laquelle est basé **Tuareg** est que pour tout couple de clusters distincts, il existe toujours au moins une dimension de projection sur laquelle ces deux clusters sont bien séparés. Cette hypothèse est tout à fait comparable à celle qui fonde l'effi cacité de C4.5 en apprentissage supervisé.

De par ces caractéristiques, **Tuareg** ne peut traiter correctement des cas comme ceux de la fi gure 5. Dans tous ces exemples, même quand les clusters sont bien séparés dans l'espace complet de description, cette séparation ne se traduit par aucun « trou » dans les projections sur une

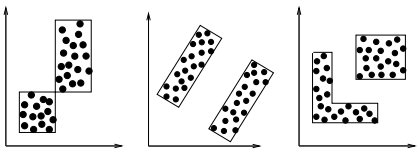


FIG. 5 – Exemples de cas que l'on ne peut capturer avec **Tuareg**.

seule dimension, ce qui fait échouer l'algorithme de partitionnement élémentaire. Ceci se traduit par exemple par de mauvais résultats sur une base de données bien connue en apprentissage supervisé : les *Iris*, où trois types d'iris (fleurs) sont décrits. L'une des classes est bien séparée des deux autres, mais ces dernières ne sont pas linéairement séparables, et **Tuareg** ne retourne donc que deux clusters.

Pour relâcher la contrainte que les « trous » séparant les clusters doivent se projeter sur au moins une dimension de description (ce qui signifie qu'ils doivent être parallèles aux axes), nous nous proposons d'étudier une variante de **Tuareg** mettant en oeuvre une première phase d'Analyse en Composantes Principales. Cette phase devrait permettre, via un changement de repère, de retrouver des clusters dont l'orientation générale est parallèle aux axes, alors que ce n'était pas le cas initialement. Cette phase préliminaire risque néanmoins d'entraîner des temps de calculs plus conséquents.

Par ailleurs, l'un des principaux intérêts de **Tuareg** est qu'il demande très peu de *connaissance a priori* de la part de l'utilisateur, contrairement à de nombreuses méthodes qui requièrent de spécifier le nombre exact de clusters recherchés, ou leur inertie maximale par exemple. Le seul paramètre à fixer en entrée est une borne supérieure sur le nombre de clusters recherchés. Or, si cette borne est fixée initialement à un niveau trop bas, il est très facile de relancer après coup **Tuareg** sur un cluster résultant considéré comme pas assez homogène, et ceci d'autant plus que **Tuareg** fournit un *résultat compréhensible,* spécifique pour chacun des clusters générés. Il effectue en effet de la *sélection d'attributs*, en même temps qu'il partitionne la base d'objets, et reste *stable face aux dimensions non pertinentes*.

Tuareg est d'une complexité très raisonnable (cf. fin de la section 1). De plus, une fois les clusters créés, la *classification est très rapide*, car il suffit alors de considérer les dimensions caractéristiques des clusters, et non plus toutes les dimensions de l'espace. Notons aussi que plusieurs étapes de calculs peuvent être réalisées indépendamment les unes des autres. Les performances de **Tuareg** pourront donc être encore améliorées par la mise en œuvre d'une version parallèle.

Ces bonnes propriétés permettent d'envisager divers domaines d'applications. De manière générale, les domaines d'application traditionnels des algorithmes de *subspace clustering* sont des contextes naturels d'utilisation pour **Tuareg**. Il pourrait par exemple être utilisé pour repérer les zones denses en données dans de grandes bases de données de type OLAP. Dans ces systèmes, l'optimisation de requêtes passe par une indexation adaptée à la répartition des données dans l'espace de description. La caractérisation de groupes de données par un sous-ensemble des dimensions de description peut être d'une aide précieuse dans ce cadre. **Tuareg** faisant de

la sélection d'attributs en même temps que du clustering, nous pensons qu'il peut aussi être intéressant de l'utiliser pour préparer des données destinées à l'apprentissage supervisé. Il devrait aussi être bien adapté au cas de bases de données avec des valeurs manquantes. En effet, si les valeurs manquantes correspondent à des dimensions de description non pertinentes, **Tuareg** ne sera pas perturbé par leur absence.

Enfi n, nous pensons pouvoir étendre **Tuareg** à des données autres que numériques.

Références

AGGARWAL C. C., WOLF J. L., YU P. S., PROCOPIUC C. & PARK J. S. (1999). Fast algorithms for projected clustering. In *ACM SIGMOD Int. Conf. on Management of Data*, p. 61–72.

AGGARWAL C. C. & YU P. S. (2000). Finding generalized projected clusters in high dimensional spaces. In *ACM SIGMOD Int. Conf. on Management of Data*, p. 70–81.

AGRAWAL R., GEHRKE J., GUNOPULOS D. & RAGHAVAN P. (1998). Automatic subspace clustering of high dimensional data for data mining applications. In *ACM SIGMOD Int. Conf. on Management of Data*, p. 94–105, Seattle, Washington.

ANKERST M., BREUNIG M., KRIEGEL H.-P. & SANDER J. (1999). Optics : Ordering points to identify the clustering structure. In *ACM SIGMOD Int. Conf. on Management of Data*, p. 49–60, Philadelphia, Pennsylvania.

BERKHIN P. (2002). *Survey Of Clustering Data Mining Techniques.* Rapport interne, Accrue Software, San Jose, California.

BLAKE C. & MERZ C. (1998). UCI repository of machine learning databases [http ://www.ics.uci.edu/∼mlearn/MLRepository.html].

BRÉZELLEC P. & DIDIER G. (2001). Gizmo : un algorithme de grille cherchant des clusters homogènes. *CAP 2001*, p. 101–116.

CAO Y. & WU J. (2002). Projective ART for clustering data sets in high dimensional spaces. In *Neural Networks 15*, p. 105–120.

CHENG C. H., FU A. W.-C. & ZHANG Y. (1999). Entropy-based subspace clustering for mining numerical data. In *Knowledge Discovery and Data Mining*, p. 84–93.

DIDAY E., LEMAIRE J., POUGET J. & TESTU F. (1982a). *Un algorithme de partitionnement optimal dans le cas d'une variable unique*, In *Eléments d'analyse de données*, p. 129–132. Dunod.

DIDAY E., LEMAIRE J., POUGET J. & TESTU F. (1982b). *Un algorithme de type nuées dynamiques*, In *Eléments d'analyse de données*, p. 117–123. Dunod.

ESTER M., KRIEGEL H.-P., SANDER J. & XU X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *2nd Int. Conf. on Knowledge Discovery and Data Mining*, p. 226–231, Portland, Oregon.

FISHER D. (1987). Knowledge acquisition via incremental conceptual clustering. In *Machine Learning*, p. 139–172.

FISHER D. (1995). Iterative optimization and simplification of hierarchical clusterings. *Artifi cial Intelligence Research*.

FRIEDMAN J. H. & MEULMAN J. J. (2004). Clustering objects on subsets of attributes. *Journal of the Royal Statistical Society : Series B (Statistical Methodology)*, **66**(4), 1–25.

GENNARI J. H., LANGLEY P. & FISHER D. H. (1989). Models of incremental concept formation. In *Artifi cial Intelligence*, p. 11–61.

GUHA S., RASTOGI R. & SHIM K. (1998). CURE : an efficient clustering algorithm for large databases. In *ACM SIGMOD Int. Conf. on Management of Data*, p. 73–84.

HINNEBURG A. & KEIM D. A. (1998). An efficient approach to clustering in large multimedia databases with noise. In *Knowledge Discovery and Data Mining*, p. 58–65.

HINNEBURG A. & KEIM D. A. (1999). Cluster discovery methods for large data bases - from the past to the future. In *ACM SIGMOD Int. Conf. on Management of Data*. Tutorial Session.

JAIN A., MURTY M. & FLYNN P. (1999). Data clustering : a review. *ACM Computing Surveys*, **31**(3), 264–322.

KARYPIS G., HAN E.-H. S. & KUMAR V. (1999). Chameleon : Hierarchical clustering using dynamic modeling. *Computer*, **32**(8), 68–75.

LIU B., XIA Y. & YU P. S. (2000). Clustering through decision tree construction. In *9th Int. Conf. on Information and Knowledge Managment*, p. 20–29.

NAGESH H., GOIL S. & CHOUDHARY A. (1999). *MAFIA : Effi cient and scalable subspace clustering for very large data sets*. Rapport interne, Northwestern University.

NG R. T. & HAN J. (1994). Efficient and effective clustering methods for spatial data mining. In J. BOCCA, M. JARKE & C. ZANIOLO, Eds., *20th Int. Conf. on Very Large Data Bases*, p. 144–155, Santiago, Chile : Morgan Kaufmann Publishers.

PARSONS L., HAQUE E. & LIU H. (2004). Evaluating subspace clustering algorithms. In *Workshop on Clustering High Dimensional Data and its Applications, SIAM Int. Conf. on Data Mining*, p. 48–56.

PROCOPIUC C. M., JONES M., AGARWALA P. K. & MURALI T. M. (2002). A monte carlo algorithm for fast projective clustering. In *ACM SIGMOD Int. Conf. on Management of Data*, p. 418–427.

SHEIKHOLESLAMI G., CHATTERJEE S. & ZHANG A. (2000). Wavecluster : a wavelet-based clustering approach for spatial data in very large databases. *VLDB Journal*, **8**(3-4), 289–304.

SU M.-C. & CHANG H.-T. (2000). Fast self-organizing feature map algorithm. *IEEE-NN*, **11**(3), 721–733.

WANG H., WANG W., YANG J. & YU P. S. (2002). Clustering by pattern similarity in large data sets. In *ACM SIGMOD Int. Conf. on Management of Data*, p. 394–405.

WANG W., YANG J. & MUNTZ R. R. (1997). STING : A statistical information grid approach to spatial data mining. In M. JARKE, M. J. CAREY, K. R. DITTRICH, F. H. LOCHOVSKY, P. LOUCOPOULOS & M. A. JEUSFELD, Eds., *23rd Int. Conf. on Very Large Data Bases*, p. 186–195, Athens, Greece : Morgan Kaufmann.

WOO K. & LEE J. (2002). *FINDIT : a fast and intelligent subspace clustering algorithm using dimension voting*. PhD thesis, Korea Advanced Institute of Science and Technology, Department of Electrical Engineering and Computer Science.

XU X., ESTER M., KRIEGEL H.-P. & SANDER J. (1998). A distribution-based clustering algorithm for mining in large spatial databases. In *14th Int. Conf. on Data Engineering*, p. 324–331, Orlando, Florida, USA.

YANG J., WANG W., WANG H. & YU P. S. (2002). δ-cluster : capturing subspace correlation in a large data set. In *18th Int. Conf. on Data Engineering*, p. 517–528.

YIP K. Y., CHEUNG D. W. & NG M. K. (2003). A highly-usable projected clustering algorithm for gene expression profiles. In *3rd ACM SIGKDD Workshop on Data Mining in Bioinformatics*, p. 41–48.

ZHANG T., RAMAKRISHNAN R. & LIVNY M. (1997). Birch : A new data clustering algorithm and its applications. *Data Mining and Knowledge Discovery*, **1**(2), 141–182.