Université Lumière Lyon2
Année 2003

Thèse
pour obtenir le grade de
Docteur
en
Informatique

présentée et soutenue publiquement par

Pierre-Emmanuel JOUVE
le 10 décembre 2003

Apprentissage Non Supervisé et Extraction de Connaissances à partir de Données

préparée au sein du laboratoire ERIC
Equipe de Recherche en Ingénierie des Connaissances

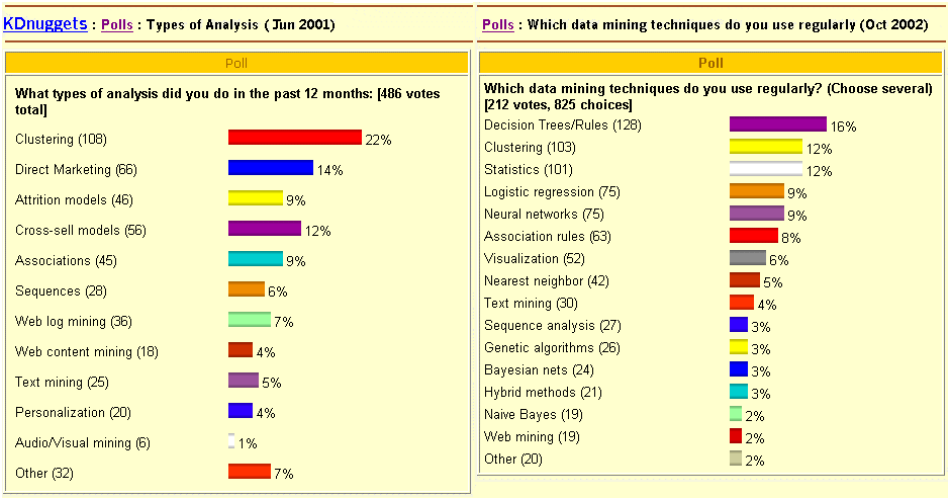
sous la direction de
Nicolas Nicoloyannis

devant le jury, composé de:

Jean-Paul Rasson, Rapporteur	Professeur, Facultés Universitaires N.D. de la Paix, Namur
Gilles Venturini, Rapporteur	Professeur, Université de Tours
Mohand-Saïd Hacid, Examinateur	Professeur, Université Claude Bernard-Lyon 1
Michel Lamure, Examinateur	Professeur, Université Claude Bernard-Lyon 1
Gilbert Ritschard, Examinateur	Professeur, Université de Genève
Nicolas Nicoloyannis, Directeur de thèse	Professeur, Université Lumière-Lyon 2

Table des matières

3 Classification Non Supervisée



Résultats de Sondages du Site Web KDNuggets (juin 2001 et octobre 2002) (KDNuggets Polls : www.kdnuggets.com/polls/)

3.1 Introduction

La classification non supervisée (cns) constitue l'un des outils les plus populaires de la fouille de données. Son utilisation permet la découverte de groupes, de motifs, d'éléments structurels à l'intérieur d'un jeu de données. La problématique sous-jacente à la cns consiste donc en le partitionnement de l'ensemble des objets du jeu de données considéré en des groupes tels que les objets d'un même groupe sont relativement similaires entre eux alors que ces objets sont relativement différents des objets des autres groupes [GRS98] [JMF99]. Ainsi, l'idée est de mettre à jour la structure interne des données en révélant l'organisation des objets par l'intermédiaire de groupes. L'analyse de ces groupes permettra alors de dériver un ensemble de connaissances sur le jeu de données.

La cns est utilisée dans divers domaines tels la biologie, la médecine, l'ingénierie... D'ailleurs, à l'appellation cns peut être substitué un ensemble d'autres

termes selon le contexte d'utilisation : apprentissage non supervisé (en reconnaissance de formes), taxonomie (en sciences de la vie), typologie (en sciences humaines)...

Nous proposons ici une rapide et bien évidemment non exhaustive présentation du processus de classification non supervisée. Le domaine est si vaste qu'envisager une présentation intégrale du domaine semble non réaliste, toutefois nous recommandons la lecture des travaux de [JMF99] [Ber02] [JD88] [HBV01] qui permettent une approche relativement complète du domaine.

3.1.1 Méthodologie Générale de la Classification Non Supervisée

La méthodologie générale de la cns est largement semblable à celle de l'ECD : elle est constituée d'un ensemble de tâches s'enchaînant séquentiellement et pouvant être intégrées au sein d'un processus itératif. Ces étapes sont les suivantes [FPSSU96] (voir figure 3.1) :

1. **Sélection de Variables** (SdV) : l'objectif est essentiellement ici de procéder à la sélection des variables encodant le mieux l'information que l'on veut soumettre à l'analyse. Plus rarement, la SdV pour la cns à un objectif identique à celui de la SdV dans le cadre de l'apprentissage supervisé : l'élimination de l'information non pertinente. Notons que dans ce dernier cas la nature non supervisée et l'utilisation à but exploratoire de la cns confère à la SdV un aspect paradoxal : "Dans quelle mesure puis-je conjecturer que cette information n'est pas pertinente alors que je n'ai aucune connaissance, idée, sur ce que je désire découvrir?". Nous proposons toutefois au chapitre 5 une méthode apparemment efficiente pour la SdV dans ce cadre particulier.
2. **Application de l'algorithme de cns** : cette étape centrale concerne le choix de la méthode de cns à utiliser, sa paramétrisation, le choix des mesures de similarités/dissimilarités à employer.
3. **Validation des Résultats** : de par la nature non supervisée de la cns, l'évaluation de la validité des structures mises à jour est essentielle mais relativement ardue. Nous abordons en détail ce point au chapitre 4.
4. **Interprétation des Résultats** : Dans de nombreux cas des experts doivent intégrer les résultats de la cns dans un ensemble plus vaste d'analyses afin d'aboutir à des conclusions intéressantes et valides.

3.1.2 Applications de la Classification Non Supervisée

Comme le montre les sondages réalisés sur le site KDNUGGETS[1] la cns est un outil majeur en ECD (voir images page 15). Nous résumons ici les applications de la cns en nous référant à [HBV01] :

– *Réalisation de Taxonomies* : il s'agit évidemment de l'application principale de la cns, les taxonomies générées pouvant alors être utilisées telles

1. www.kdnuggets.com

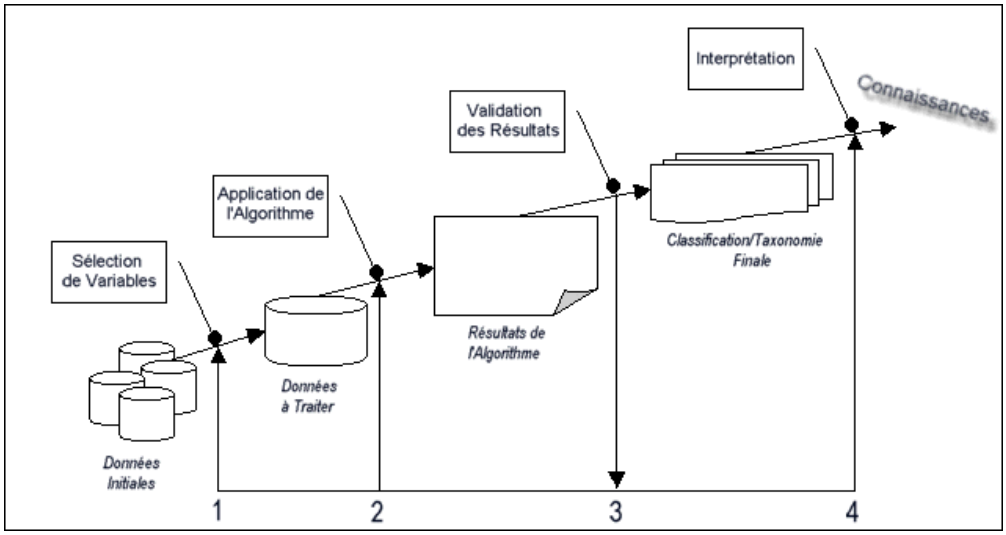


FIG. 3.1 –: *Etapes du Processus de Classification Non Supervisée*

qu'elles ou au sein d'analyses plus élaborées (la cns peut, par exemple, constituer une étape de pré-traitement de l'information préalable à l'utilisation d'autres algorithmes comme ceux d'apprentissage supervisé).

– *Résumer l'information* : la cns peut contribuer à la compression de l'information comprise dans un jeu de données, et permettre une présentation synthétique des éléments informationnels les plus marquants d'un jeu de données.

– *Génération/Test d'hypothèses* : La cns peut être utilisée afin d'inférer des hypothèses sur l'organisation, les relations existant parmi les objets d'un jeu de données ; de manière opposée, la cns peut être utilisée dans le but de confirmer/rejeter un ensemble d'hypothèses préalablement émises au sujet des objets d'un jeu de données.

3.1.3 Taxonomies des Méthodes de Classification Non Supervisée

Une multitude de méthodes de cns sont proposées dans la littérature, on classifie traditionnellement ces méthodes selon :

– Le type de données qu'elles peuvent traiter

– La mesure employée pour rendre compte de la similarité / dissimilarité entre objets

– Les éléments théoriques et les concepts fondamentaux sous-jacents à la méthode.

Les méthodes de cns sont ainsi généralement classées selon les groupes suivants [JMF99] :

- les *méthodes "partitionnelles"* déterminent une partition des objets du jeu de données ; pour cela elles procèdent à une recherche itérative de la partition optimisant un critère particulier.
  Méthode(s) Représentative(s) : K-Means [Mac67], Fuzzy C-Means [BEF84], K-Modes [Hua97], PAM [NH94], CLARA [NH94], CLARANS [NH94]....
- les *méthodes hiérarchiques* procèdent, quant à elles, par fusion ou segmentation de classes d'une partition en initialisant la partition initiale soit à la partition grossière (une unique classe) soit à la partition la plus fine (chaque objet constitue alors une classe). Dans le premier cas, la méthode procède par segmentation de classes, on parle alors de méthode divisive ou encore descendante ; dans le second, cas la méthode procède par fusion de classes et l'on parle cette fois de méthode agglomérative ou ascendante. Le résultat est alors un arbre appelé dendrogramme dont chaque niveau correspond à une partition particulière du jeu de données.
  Méthode(s) Représentative(s) : BIRCH [ZRL96], CURE [GRS98], ROCK [GRS00], Méthode de Williams et Lambert [WL59], Méthode de Fages....
- les *méthodes basées sur la densité* pour lesquelles l'idée clé est de procéder à des regroupements d'objets en groupes selon le voisinage des objets et son niveau de densité.
  Méthode(s) Représentative(s) : DBSCAN [EKSX96] [EKS+98], DENCLUE [EKS+98]...
- les *méthodes à base de grilles* sont essentiellement utilisées pour des données spatiales, elles procèdent par discrétisation de l'espace en un certains nombre de cellules et effectuent ensuite l'ensemble des traitements sur l'espace discrétisé.
  Méthode(s) Représentative(s) : Wave-Cluster [SCZ98], STING [HK01] [WYM97] [WYM99].

D'autres auteurs [Hua97] [GRS00] [RLR98a] proposent également une classification selon le type de données traitées par la méthode :

- les *méthodes dites statistiques* basées sur des concepts statistiques et traitant les données quantitatives qui utilisent des mesures de similarité pour partitionner les objets,
- les *méthodes conceptuelles* utilisées pour les données catégorielles classifient quant à elles les objets selon le concept qu'ils véhiculent.

On peut également classer les méthodes de cns selon leur façon d'intégrer l'incertitude :

- les *méthodes floues* utilisent la théorie des ensembles flous afin de mettre à jour, non pas une partition de l'ensemble des objets, mais un recouvrement de cet ensemble. Ainsi, un objet peut appartenir à plusieurs classes

et un degré d'appartenance à chacune des classes auxquelles il peut appartenir lui est asssigné. Ce type de méthodes peut donc permettre de transcrire les cas pratiques où l'incertitude doit être intégrée.
Méthode(s) Représentative(s) : Fuzzy C-Means [BEF84]...

– à l'opposé les méthodes classiques, les plus nombreuses, considèrent des partitions de l'ensemble des objets sans aucun recouvrement.

D'autres éléments permettent d'effectuer des distinctions entre méthodes, ainsi on peut opposer les méthodes par :

– leurs approches d'optimisation qui peuvent être *déterministes* ou *stochastiques*,
– leur *incrémentalité* ou *non-incrémentalité*,
– leur aspect *monothétique* ou *polythétique*
– l'utilisation de paradigmes particuliers, on peut citer :
  – les *méthodes neuronales* comme les cartes de Kohonen [Koh89] qui présentent de manière graphique des partitions de l'ensemble des objets ce qui permet d'obtenir des connaissances sur la proximité entre classes de la partition,
  – *les méthodes évolutionnaires* utilisant les algorithmes génétiques (telle la méthode proposée par Cristofor et Simovici [CS02]) ou la programmation génétique...
– ...

3.1.4 Méthodes de Classification Non Supervisée pour Données Catégorielles

Les travaux présentés ultérieurement concernant la cns pour données catégorielles, nous donnons maintenant les informations majeures (type de méthode, complexité algorithmique, géométrie des groupes extraits, faculté à gérer les outliers, paramètres, forme des résultats, critère sous-jacent à la méthode, publication de référence) concernant quelques une des méthodes de cns pour données catégorielles les plus récentes. Nous abordons ainsi successivement les méthodes **K-Modes**, **ROCK**, **RDA/AREVOMS**, une méthode proposée par **Cristofor et Simovici**, et enfin **STIRR**.

La plus connue et la plus populaire des méthodes de cns est certainement la méthode des K-Means [Mac67] (qui possède plusieurs évolutions comme les méthodes de nuées dynamiques). Cette méthode vise à déterminer la partition de l'ensemble des objets du jeu de données considéré telle qu'elle optimise (minimise) la fonction objectif $E = \sum_{i=1..nc} \sum_{x \in C_i} d(x,m_i)$. Ici, m_i est le centre de la classe C_i, nc est le nombre de classes de la partition, et $d(x,m_i)$ est le carré de la distance euclidienne entre l'objet x et le centre de la classe à laquelle il appartient. L'objectif de cet algorithme est donc de déterminer la partition (possédant un nombre de classes fixé a priori) telle que l'homogénéité des classes

soit la plus forte possible. Plus spécifiquement, l'algorithme débute par l'initialisation du centre des nc classes, puis assigne séquentiellement chaque objet à la classe dont le centre est le plus proche, ce dernier processus étant réitéré tant que des mouvements d'objets d'une classe à une autre ont lieu.

L'algorithme des **K-Modes** [Hua97] constitue l'adaptation des K-Means au cadre des données catégorielles. Pour ce faire, la notion de mode d'une classe (voir chapitre 2) est substituée à celle de centre d'une classe. Le critère à optimiser (à minimiser plus précisément) est donc le suivant :
$QKM = \sum_{i=1..nc} \sum_{x \in C_i} d(x,mode^{C_i})$, $d(x,mode^{C_i}) = dissim(x,mode^{C_i})$.

Ses principales caractéristiques sont :

- **Complexité :** $O(n)$,
- **Géométrie des Classes :** Forme non-concave,
- **Gestion des Outliers :** Non,
- **Paramètres :** Nombre de Classes,
- **Résultats :** Composition et Mode de chaque classe,
- **Critère :** Minimisation du critère QKM

La méthode **ROCK** [GRS00] est, elle, à classer dans la catégorie des méthodes hiérarchiques. La particularité principale de cette méthode est d'employer deux nouveaux concepts :

- Le concept de voisins d'un objet, que l'on peut décrire grossièrement comme les objets significativement similaires à cet objet. En fait, une mesure de similarité/proximité (qui peut être ou non une métrique) est utilisée afin de rendre compte de la proximité d'un couple d'objets, et si la valeur de cette mesure remplit certaines conditions alors les objets seront considérés comme voisins.
- Le concept de liens entre deux objets est utilisé pour mesurer la similarité/proximité entre deux objets. Le nombre de liens entre deux objets est défini comme le nombre de voisins en commun que possèdent ces deux objets.

Ainsi, l'algorithme hiérarchique agglomératif ROCK emploie ces concepts de voisinage et de liens (et non le concept de distance) afin de procéder aux fusions de classes. Notons que des étapes d'échantillonnage des objets peuvent être inclues dans le processus afin d'en réduire le temps d'exécution.

Ses principales caractéristiques sont :

- **Complexité :** $O(n^2 + nm_m m_a + n^2 logn)$ avec m_m le nombre maximal de voisins pour un objet, m_a le nombre moyen de voisins pour un objet,
- **Géométrie des Classes :** Pas d'a priori particulier pour la forme des classes,
- **Gestion des Outliers :** Oui
- **Paramètres :** Nombre de classes et θ une valeur de seuil pour la définition des voisins,
- **Résultats :** Composition de chaque classe,

– **Critère :** Maximisation du critère g ($g = sum_{i=1..nc} ni \times \sum_{o_a,o_b \in C_i} \frac{link(o_a,o_b)}{n_i^{1+2f(\theta)}}$, avec n_i le nombre d'objets de la classe C_i, $link(o_a,o_b)$ le nombre de liens entre o_a et o_b, $n_i^{1+2f(\theta)}$ une fonction visant à "estimer le nombre de liens que doit comprendre C_i").

Michaud décrit dans [Mic97] une méthode dénommée **RDA/AREVOMS** (Relational Data Analysis/Analyse du REsultat d'un VOte Majoritaire et S-théorie). L'originalité principale de cette méthode est l'utilisation d'un critère particulier : le Nouveau Critère de Condorcet (NCC) (voir chapitre 2). Ce critère issu de l'analyse des préférences permet une détermination automatique du nombre de classes de la partition résultant du processus de cns. Initialement, le problème d'optimisation sous-jacent à cette méthode était résolu par la programmation en nombres entiers ce qui conférait à cette méthode une non applicabilité sur des jeux de données de taille importante. Une première évolution de la méthode a alors consisté à substituer à la méthode d'optimisation par programmation en nombres entiers une méthode d'optimisation par programmation linéaire [MM81]. Le coût calculatoire est dans ce cas réduit mais l'optimalité du résultat n'est plus assurée. D'autres tentatives pour la mise au point d'une méthode plus rapide ont ensuite débouché sur la mise au point d'une méthode utilisant une heuristique de complexité en $O(n^2)$ [Mic91]. Finalement, une nouvelle heuristique de complexité en $O(n)$ a été développée par Michaud en s'appuyant sur une transformation du problème par le biais de sa S-théorie. Notons la remarquable accélération du processus due à cette méthode, ainsi que le fait que les solutions apportées par cette méthode sont très proches de l'optimalité et qu'il est également possible de fixer, si on le désire, le nombre de classes que doit posséder la partition résultat.

Ses principales caractéristiques sont :

– **Complexité :** $O(n)$,
– **Géométrie des Classes :** Pas d'a priori particulier pour la forme des classes,
– **Gestion des Outliers :** oui
– **Paramètres :** Aucun ou nombre de classes,
– **Résultats :** Composition de chaque classe,
– **Critère :** Minimiser le Nouveau Critère de Condorcet.

Cristofor et Simovici présentent quant à eux dans [CS02] une méthode visant à surpasser le problème difficile de la définition d'une mesure de similarité naturelle entre objets catégoriels. Pour cela, ils utilisent une mesure de dissimilarités entre partitions d'objets catégoriels (ces partitions étant en définitive les partitions déterminées par les variables catégorielles du jeu de données considéré). Cette mesure est basée sur la théorie de l'information et plus particulièrement sur l'entropie généralisée. Le processus de cns correspond

alors à un problème d'optimisation résolu par l'intermédiaire d'un algorithme génétique. Ses principales caractéristiques sont :

- **Complexité :** $O(gpn)$ avec g le nombre de générations de l'algorithme génétique, p le nombre de chromosomes d'une génération de l'algorithme génétique,
- **Géométrie des Classes :** Pas d'a priori particulier pour la forme des classes,
- **Gestion des Outliers :** Oui
- **Paramètres :** paramètres classiques pour un algorithme génétique, nombre maximal de générations sans amélioration de la fonction objectif,
- **Résultats :** Composition de chaque classe, poids décrivant l'influence de chacune des variables catégorielles dans la constitution du la cns,
- **Critère :** voir l'article de référence [CS02].

La méthode **STIRR** [GKR00] proposée par Gibson, Kleinberg et Raghavan est, elle, basée sur une approche itérative visant à associer des poids aux variables catégorielles d'un jeu de données afin de faciliter l'utilisation d'un certain type de mesures de similarité basées sur la cooccurrence de valeurs. Cette méthode peut ainsi être vue comme utilisant un type particulier de système dynamique non linéaire et généralisant les méthodes de partitionnement spectral de graphes. Notons que les résultats nécessitent ici une interprétation relativement ardue.

- **Complexité :** non évoquée par les auteurs, les évaluations expérimentales qu'ils proposent semblent montrer une complexité en $O(n)$,
- **Géométrie des Classes :** Pas d'a priori particulier pour la forme des classes[2],
- **Gestion des Outliers :** Oui[3]
- **Paramètres :** nombre d'itérations du processus itératif, pour le reste voir l'article de référence [GKR00]
- **Résultats :** Un graphique à interpréter,
- **Critère :** voir l'article de référence [GKR00].

3.1.5 Challenges Actuels en Classification Non Supervisée

De nombreux algorithmes classiques de cns ne permettent pas un traitement réellement satisfaisant des données pour de multiples raisons. Ces raisons peuvent être classées selon qu'elles sont inhérentes aux données traitées ou liées à des contraintes dues au domaine dans lequel est utilisé l'algorithme de cns :

- **Problèmes inhérents aux données traitées :**
  - **Très grand nombre d'objets.** Si le nombre d'objets à traiter est très élevé, les algorithmes utilisés se doivent de posséder une complexité

2. dans la mesure où les classes sont déterminées par interprétation d'un graphique

algorithmique théorique relativement faible, ou plutôt, exhiber une bonne scalabilité. En effet, comme de nombreux problèmes intéressants, la cns mène à la résolution de problèmes généralement NP-difficiles. Il est généralement admis que des algorithmes "valables" en terme de coût calculatoire doivent posséder une complexité algorithmique linéaire ou log-linéaire, dans quelques cas particuliers une complexité quadratique ou cubique peut être acceptée.

- **Dimensionnalité élevée.** Dans certains cas, le nombre de descripteurs (variables) est très élevé (parfois supérieur au nombre d'objets). Ainsi, un algorithme de cns doit pouvoir affronter "le fléau de la dimensionnalité"...[3]
- **Autres éléments problématiques.** Le problème de la *"vacuité"* ("sparsity") des données peut affecter la complexité algorithmique ainsi que le choix de la mesure de similarité à utiliser, la présence de *données manquantes* soulève également des questions pour le choix de cette mesure. La présence d'*outliers* et leur détection est un problème non trivial, ainsi il est parfois nécessaire de posséder un algorithme relativement insensible à leur présence. (Dans le cas de données catégorielles, la notion d'outliers fait référence aux objets présentant une description particulièrement différentes de l'ensemble des autres objets du jeu de données.)

- **Problèmes inhérents à des contraintes applicatives :**
  - **Necessité d'intégrer des connaissances, des contraintes dans le processus.** Le processus d'extraction des connaissances dans lequel s'insère le processus de cns est essentiellement anthropocentré et itératif. Donner la possibilité à l'utilisateur de "jouer" avec le processus de cns en y intégrant ses connaissances ou d'éventuelles contraintes est donc souvent nécessaire.
  - **Bonne utilisabilité.** Souvent l'utilisateur final d'un algorithme de cns ne s'avère pas un expert du domaine : simplicité d'utilisation (paramétrage facile et intelligible de l'algorithme), présentation explicite et intelligible des résultats et connaissances extraites par le processus de cns sont alors des éléments indispensables pour une utilisation profitable et pertinente d'un algorithme.
  - **Données distribuées.** Les grands entrepôts de données proposent le plus souvent des sources de données distribuées. Les modèles de cns que l'on peut bâtir localement nécessitent parfois d'être intégrés dans un modèle global/holistique.

Cette liste résume partiellement les problèmes en cns constituant les principales préoccupations actuelles des chercheurs dans ce domaine. Nous évalue-

3. curse of dimensionality [Fri94]

rons, en conclusion, dans quelle mesure la méthode de cns que nous proposons apporte ou non des solutions à chacun d'entre eux.

3.2 Une Nouvelle Méthode de Classification Non Supervisée "Orientée Utilisateur"

Nous proposons ici une méthode de classification non supervisée (pour données catégorielles) basée sur le concept de la comparaison par paires et plus spécifiquement sur le Nouveau Critère de Condorcet (*NCC*).

Les idées mises en œuvre pour la mise au point de cette méthode ne sont pas totalement nouvelles puisqu'il s'agit d'une généralisation (non hiérarchique) de la méthode d'analyse des associations de Williams et Lambert [WL59]. De plus, l'idée d'utiliser le concept de la comparaison par paires pour élargir la problématique de Williams et Lambert aux cas où les variables nominales ont plus de deux modalités est également évoquée dans [Mar84a] [Mar84b]. Enfin, l'utilisation du *NCC* au sein d'un algorithme de cns a été proposée par P.Michaud [Mic97] [Mic91] [Mic87] .

Nous pensons cependant que les travaux proposés ici présentent plusieurs intérêts :

- revivifier l'intérêt de la communauté ECD pour des travaux en agrégation des préférences et comparaison par paires à l'intérêt indéniable (les travaux de Michaud et Marcotorchino sont selon nous des sources de réflexions et d'inspirations immenses),
- proposer une méthode de cns possédant de multiples avantages pour l'utilisateur,
- proposer une méthode permettant de dériver une méthode d'apprentissage semi-supervisé...

Nous utilisons pour notre méthode de cns un critère dérivé du Nouveau Critère de Condorcet (NCC), la présentation de ce critère constitue le point de départ de la présentation de la méthode proposée, puis suivent la présentation de l'algorithme et son évaluation expérimentale. Nous concluons par l'introduction de plusieurs éléments additionnels conférant notamment à la méthode la possibilité d'être "transformée" en une méthode d'apprentissage semi-supervisée dont la présentation et l'évaluation achève ce chapitre.

3.2.1 Critère d'Évaluation de l'Aspect Naturel d'une Partition d'Objets

Le problème sous-jacent à la cns est : "étant donné un ensemble d'objets O, déterminer une partition P_{nat} de O que l'on dénommera naturelle et qui reflète la structure interne des données". Cette partition doit être telle que ses classes sont constituées d'objets présentant une relative forte similarité et que les objets de classes différentes présentent une relative forte dissimilarité. Pour

ce faire, on doit disposer d'un critère permettant de capturer l'aspect naturel d'une partition, nous utilisons ici $NCC^\star$ une version modifiée du Nouveau Critère de Condorcet. Le principal avantage de ce critère est, tout comme pour le critère NCC, une détermination automatique du nombre final de classes pour la cns. De plus, l'introduction d'un nouvel élément dit "facteur de granularité" permet également à l'utilisateur d'influer, de "piloter" la résolution de la partition résultant du processus de cns (i.e. d'influer sur le nombre de classes de cette partition). Voici la définition formelle de $NCC^\star(P_h)$ la mesure de l'aspect naturel d'une partition $P_h = \{C_1,...,C_z\}$:

$$NCC^\star(P_h) = \sum_{i=1..z,j=1..z,j>i} Sim(C_i,C_j) + \alpha \times \sum_{i=1}^{z} Dissim(C_i) \tag{3.1}$$

α scalaire appelé facteur de granularité ($\alpha \geq 0$), si $\alpha = 1$ alors $NCC^\star(P_h) = NCC(P_h)$

Nous rappelons ici quelques définitions données au chapitre précédent :

$$\begin{aligned} Sim(C_i,C_j) &= \sum_{o_a \in C_i, o_b \in C_j} sim(o_a,o_b) \\ Dissim(C_i) &= \sum_{o_a \in C_i, o_b \in C_i} dissim(o_a,o_b) \\ sim(o_a,o_b) &= \sum_{i=1}^{p} \delta_{sim}(o_{a_i},o_{b_i}) \\ dissim(o_a,o_b) &= \sum_{i=1}^{p} \delta_{dissim}(o_{a_i},o_{b_i}) \end{aligned} \tag{3.2}$$

$$\delta_{sim}(o_{a_i},o_{b_i}) = 1 - \delta_{dissim}(o_{a_i},o_{b_i}) = \begin{cases} 1 \text{ si } o_{a_i} = o_{b_i} \\ 0 \text{ si } o_{a_i} \neq o_{b_i} \end{cases} \tag{3.3}$$

Ainsi, $NCC^\star(P_h)$ mesure simultanément les dissimilarités entre objets de même classe de la partition P_h, et les similarités entre objets de classes différentes (on peut donc dire que $NCC^\star(P_h)$ est défini comme une fonction de l'homogénéité interne des classes et de l'hétérogénéité entre classes). Donc, les partitions présentant une forte homogénéité intra-classe et une forte disparité inter-classes posséderont une faible valeur pour $NCC^\star$ et constitueront les partitions apparaissant comme les plus naturelles.

Définition 5 *Une partition P_1 est dite plus naturelle qu'une partition P_2 (ou encore représentant mieux la structure interne des données) si $NCC^\star(P_1) < NCC^\star(P_2)$.*

Définition 6 *Une partition d'un ensemble d'objets O est nommée Partition Naturelle de O et est notée P_{nat} si elle minimise $NCC^\star$:*
$\wp$ ensemble des partitions de O, $\forall P_i \in \wp, NCC^\star(P_{nat}) \leq NCC^\star(P_i)$.

REMARQUES :

- Il peut y avoir plusieurs partitions naturelles différentes pour un ensemble d'objets; par exemple si $\alpha = 1$,
  $O = \{ICEL,FRAN,SWED,NORW,DENM,USA,UK,NETH,BELG\}$,
  $P_1 = \{\{ICEL,FRAN,SWED,NORW,DENM\},\{USA,UK,NETH,BELG\}\}$,
  $P_2 = \{\{FRAN,SWED,NORW,DENM\},\{ICEL,USA,UK,NETH,BELG\}\}$,
  $NCC^\star(P_1) = NCC^\star(P_2) = 28$, et $\wp$ ensemble des partitions de O,
  $\forall P_i \in \wp, 28 \leq NCC^\star(P_i)$
- Notons le rôle du facteur de granularité α (non présent dans la définition initiale du NCC) : celui ci permet soit de privilégier l'influence de l'homogénéité intra-classe ou de la disparité inter-classes pour la détermination de l'aspect naturel d'une partition. En effet, plus α est élevé (resp. faible) plus une partition doit présenter une forte homogénéité intra-classe (resp. une forte disparité inter-classes) pour apparaître naturelle. Par exemple, si
  $O = \{ICEL,FRAN,SWED,NORW,DENM,USA,UK,NETH,BELG\}$,
  $P_1 = \{\{ICEL,FRAN,SWED,NORW,DENM\},\{USA,UK,NETH,BELG\}\}$,
  $P_3 = \{\{FRAN,SWED,NORW,DENM\},\{ICEL\},\{USA,UK,NETH,BELG\}\}$,
  $NCC^\star(P_1) = 24 + \alpha \times 4$, $NCC^\star(P_3) = 32 + \alpha \times 0$, ainsi
  pour $\alpha = 1$ $NCC^\star(P_1) < NCC^\star(P_3)$,
  pour $\alpha = 2$ $NCC^\star(P_1) = NCC^\star(P_3)$,
  pour $\alpha = 3$ $NCC^\star(P_1) > NCC^\star(P_3)$

3.2.2 La Méthode de Classification Non Supervisée "Orientée Utilisateur"

3.2.2.1 Travaux Liés et Spécificités du Travail

La cns pour données catégorielles a été l'objet de multiples travaux ces dernières années, les principaux challenges associés à ces recherches sont :

- **la définition de critères bien adaptés à ce cadre particulier** :
  - dans ROCK [GRS00] les auteurs utilisent une mesure basée sur le nombre de voisins communs que possèdent deux objets,
  - Cristofor et Simovici [CS02] utilisent quant à eux une mesure basée sur l'entropie généralisée,
  - Michaud [Mic97] utilise quant à lui le NCC ;
- **la mise au point d'algorithmes au coût calculatoire relativement faible** :
  - Huang [Hua97] propose les K-Modes une adaption de la méthode des K-Means dans le cas de données catégorielles,
  - Gibson et al. [GKR00] utilisent les systèmes dynamiques pour STIRR,
  - Michaud s'appuie sur la S-théorie [Mic87] [Mic91] pour dériver un algorithme efficace [Mic97].

Notre propos n'est pas ici de poursuivre dans ces directions de recherche mais plutôt de s'appuyer sur ces travaux afin de proposer une méthode efficace exhibant de nombreux avantages pour l'utilisateur et utilisable par un non spécialiste. En effet, l'aspect utilisabilité est trop peu abordé dans la littérature. On peut ainsi lister un ensemble de qualités pratiques désirables par un utilisateur et qui distinguerait largement une méthode les possédant des approches existantes :

- générer une description des classes définies aisément compréhensible et ne nécessitant aucun post-traitement contrairement à [GRS00], [CS02], [GKR00] [Mic97]
- coût calculatoire relativement faible contrairement à [GRS00]
- ne pas nécessiter de fixer a priori le nombre final de classes contrairement à [Hua97] mais toutefois permettre à l'utilisateur de jouer sur la finesse de la partition s'il considère que le nombre de classes obtenues est trop élevé ou trop faible.
- permettre la découverte de structures ne présentant pas une régularité dans le nombre d'objets par classe contrairement à [Hua97]
- permettre la gestion de données manquantes, l'introduction de contraintes et l'intervention de l'utilisateur au cours du processus de cns et ce de manière aisée.

L'objectif du travail mené est donc la mise au point d'une méthode de cns permettant la mise à jour de partitions pertinentes et possédant ces qualités pratiques. La méthode ainsi mise au point a été nommée KEROUAC, pour sa présentation au Workshop International DATA MINING FOR ACTIONABLE KNOWLEDGE ("Fouille de Données pour Connaissances Utilisables") mené conjointement à la conférence PAKDD2003 (Pacific-Asia Conference on Knowledge Discovery and Data Mining)[4]. KEROUAC correspond à un acronyme pour les termes anglais **K**nowledge **E**xplicit, **R**apid, **O**ff-beat, **U**ser-centered, **A**lgorithm for **C**lustering, ces termes faisant référence aux principales qualités de la méthode : caractérisation explicite des classes mises à jour (**K**nowledge **E**xplicit), coût calculatoire relativement faible (**R**apid), bonne utilisabilté (**U**ser-centered).

3.2.2.2 L'Algorithme de Classification Non Supervisée

L'algorithme que nous proposons consiste en une mise en œuvre astucieuse et nouvelle de principes et techniques existants afin d'atteindre les objectifs détaillés précédemment. Pour cela nous utilisons le critère $NCC^{\star}$, et une technique de type graphes d'induction pour découvrir une partition $P_{\sim nat}$ proche ou égale à P_{nat} ce qui conférera un aspect non hiérarchique à la méthode

4. Ce travail a également fait l'objets de deux autres publications [JN03a] et [JN03b]

([WL59] avaient proposé une technique de type arbre d'induction ce qui conférait un aspect hiérarchique à la méthode et utilisaient un critère de type khi2).

REMARQUES :

- La découverte de P_{nat} (problème combinatoire) peut être résolue par des méthodes au coût calculatoire élevée : par une approche de type Programmation en nombre entier, par une approche basée sur les méthodes de Plans de Coupe et Branch and Bound [GW89].
- La découverte de $P_{\sim nat}$ peut être effectuée grâce à des heuristiques de coût calculatoire en $O(n^2)$: Nicoloyannis a proposé une approche itérative basée sur la prétopologie [Nic88], Nicoloyannis et al. utilisent une méthode itérative utilisant le recuit-simulé permettant la découverte de $P_{\sim nat}$ [NTT98], Michaud et Marcotorchino [MM81] utilisent une approche de type programmation linéaire, plus tard Michaud propose une méthode au coût calculatoire en $O(n)$ [Mic97].
- Ces méthodes ne possèdent pas les éléments d'utilisabillité désirés.

Nous avons donc adopté une heuristique gloutonne de type graphe d'induction (on procède par segmentation/fusion successives de classes de partitions). En définitive, à partir de la partition grossière de O, l'algorithme va évoluer itérativement de partition en partition en suivant le principe d'évolution suivant : on passe d'une partition P_i vers la partition P_{i+1} appartenant à son voisinage (cf. Définition 4 page 13) telle qu'elle réduise au plus le $NCC^\star$. L'algorithme s'achève lorsque $P_i = P_{i+1}$. Le pseudo-code de l'algorithme est donc :

Algorithme 1 Algorithme de Classification Non Supervisée

Paramètres : α (le facteur de granularité)

1. soit P_0 la partition grossière de O

2. i:=0

3. Déterminer $Vois(P_i)$

4. Déterminer $P_{i+1} \in Vois(P_i)$ la meilleure partition de $Vois(P_i)$ selon le $NCC^\star$

5. Si $P_{i+1} = P_i$ aller en 6), sinon i:=i+1 et aller en 3)

6. $P_{\sim nat} = P_i$

REMARQUES : Pour éviter des minima locaux, on peut autoriser dans le cas $P_{i+1} = P_i$, un nombre fixé a priori d'évolutions vers la partition du voisinage de P_i obtenue par segmentation selon une variable et impliquant la plus faible augmentation du $NCC^\star$.

3.2.2.3 Complexité de l'Algorithme

La complexité de l'algorithme est équivalente à celle des graphes d'induction. En effet, bien que la définition du critère $NCC^{\star}$ semble impliquer une comparaison de l'ensemble des couples d'objets (soit une complexité quadratique selon le nombre d'objets du jeu de données pour l'évaluation de la valeur du critère $NCC^{\star}$ pour une partition donnée), des astuces calculatoires (illustrées par la suite) permettent de réduire le coût calculatoire associé à l'évaluation de la valeur du critère $NCC^{\star}$ pour une partition. Ce coût n'est alors que linéaire selon le nombre d'objets du jeu de données.

En effet, la seule connaissance de la composition de chacune des classes d'une partition permet de déterminer sa valeur pour $NCC^{\star}$.

Considérons par exemple la partition P_h :
$P_h = \{C_1,C_2,C_3\} = \{\{ICEL,FRAN,SWED\},\{USA,UK\},\{POLA,USSR,CUBA\}\}$.
Soient :

- $n_{C_{i_{k,j}}}$ le nombre d'objets de la classe C_i ayant la valeur v_{jk} pour la variable $V_j \in EV$
- $n_{O_{k,j}}$ le nombre d'objets de O ayant la valeur v_{jk} pour la variable $V_j \in EV$.

On peut pour chaque valeur v_{jk} de chacune des variables $V_j \in EV$ déterminer $n_{O_{k,j}}$; pour chaque valeur v_{jk} de chacune des variables $V_j \in EV$ et chaque classe C_i de P_h déterminer $n_{C_{i_{k,j}}}$. Ces valeurs sont données dans le tableau 3.1.

	$M1$				$M2$				$M3$		
	A	B	C	D	A	B	C	D	A	B	C
C_1	0	0	3	0	0	3	0	0	1	0	2
C_2	0	0	2	0	0	0	2	0	2	0	0
C_3	3	0	0	0	2	0	0	1	0	0	3
O	3	0	5	0	2	3	2	1	3	0	5

TAB. 3.1 –: *Elements d'illustration de la complexité algorithmique*

On peut montrer que :

$$Sim(C_i,C_j) = \sum_{l=1}^{p} \sum_{k=1}^{card(Dom(V_l))} n_{C_{i_{k,l}}} \times n_{C_{j_{k,l}}} \quad (3.4)$$

$$Dissim(C_i) = \sum_{l=1}^{p} \sum_{k=1}^{card(Dom(V_l))} \frac{n_{C_{i_{k,l}}} \times (card(C_i) - n_{C_{i_{k,l}}})}{2} \quad (3.5)$$

Ce qui permet de calculer la valeur du critère $NCC^{\star}$. Ainsi,
$Sim(C_1,C_2) = (0\times0+0\times0+3\times2+0\times0)+(0\times0+3\times0+0\times2+0\times0)+(1\times2+0\times0+2\times0) = 8$
$Sim(C_1,C_3) = (0\times3+0\times0+3\times0+0\times0)+(0\times2+3\times0+0\times0+0\times0)+(1\times0+0\times0+2\times3) = 6$

$Sim(C_2,C_3) = (0\times3+0\times0+2\times0+0\times0)+(0\times2+0\times0+2\times0+0\times1)+(2\times0+0\times0+0\times3) = 0$
$Dissim(C_1) = (\frac{0\times3}{2}+\frac{0\times3}{2}+\frac{3\times0}{2}+\frac{0\times3}{2})+(\frac{0\times3}{2}+\frac{3\times0}{2}+\frac{0\times3}{2}+\frac{0\times3}{2})+(\frac{1\times2}{2}+\frac{0\times3}{2}+\frac{2\times1}{2}) = 2$
$Dissim(C_2) = (\frac{0\times2}{2}+\frac{0\times2}{2}+\frac{2\times0}{2}+\frac{0\times2}{2})+(\frac{0\times2}{2}+\frac{0\times2}{2}+\frac{2\times0}{2}+\frac{0\times2}{2})+\frac{2\times0}{2}+\frac{0\times2}{2}+\frac{0\times2}{2}) = 2$
$Dissim(C_3) = (\frac{3\times0}{2}+\frac{0\times3}{2}+\frac{0\times3}{2}+\frac{0\times3}{2})+(\frac{2\times1}{2}+\frac{0\times3}{2}+\frac{0\times3}{2}+\frac{1\times2}{2})+(\frac{0\times3}{2}+\frac{0\times3}{2}+\frac{3\times0}{2}) = 2$
d'où
$NCC(P_h) = 8+6+0+2+2+2$

Le calcul de la valeur du critère $NCC^\star$ possédant une complexité seulement linéaire dans le nombre d'objets du jeu de données, notre méthode possède donc bien une complexité équivalente à celle des graphes d'induction.

3.2.2.4 Qualités de la Méthode pour l'Utilisateur

- Chacune des classes de la partition résultat $(P_{\sim nat})$ est caractérisée par une règle logique (règle formée de disjonctions de conjonctions de sélecteurs) correspondant à la suite de mécanismes (fusion / segmentation) lui ayant donné le jour. Cette règle correspond alors pour un objet de O à une condition nécessaire et suffisante pour appartenir à la classe qu'elle décrit. (voir l'exemple page suivante)
- On associera également à chaque classe de $P_{\sim nat}$ son mode, celui-ci correspondant en définitive à une sorte de profil ou individu type de la classe.
- Ces deux premiers points simplifient largement la compréhension et l'interprétation des résultats.
- La description de chaque classe par une règle logique peut également permettre l'assignation d'un nouvel objet à l'une des classes sans pour autant connaître l'ensemble de ses caractéristiques.
- le nombre de classes est déterminé automatiquement par la méthode, toutefois l'utilisateur peut influer sur la finesse de la partition finalement produite par l'intermédiaire du facteur de granularité. (Si le nombre de classes apparaît trop faible (resp. trop fort) à l'utilisateur, ce dernier peut procéder à une nouvelle cns en augmentant (resp. en diminuant) la valeur du facteur de granularité).

3.2.2.5 Illustration du Fonctionnement de l'Algorithme

Considérons un jeu de données classique : les données mushrooms[MM96]. Ce jeu de données est composé de 8124 objets (en l'occurrence des champignons), chacun décrit par 23 variables. Chaque objet est, de plus, identifié par sa comestibilité (voir page 217 pour une présentation complète du jeu de données). Le jeu de données est ainsi composé de champignons comestibles et de champignons vénéneux.

La figure 3.2 présente le processus de cns sur le jeu de données "Mushrooms" pour un facteur de granularité α valant 1. Elle détaille ainsi l'ensemble des processus de segmentation/fusion, permettant l'obtention de la cns. Voici pour exemple les règles logiques caractérisant les classes C8 et C10 (rappelons que

chacune de ces règles détermine l'appartenance ou la non appartenance de tout objet à la classe à laquelle elle est associée) :

- C8 [Ring_Type = evanescent] ET [Gill_Size = broad] ET [Bruises? = bruises]
- C10 [Ring_Type = pendant] ET [[Bruises? = bruises] OU [[Bruises? = no] ET [stalk-color-above-ring = white] ET [Gill_Size = narrow]]]

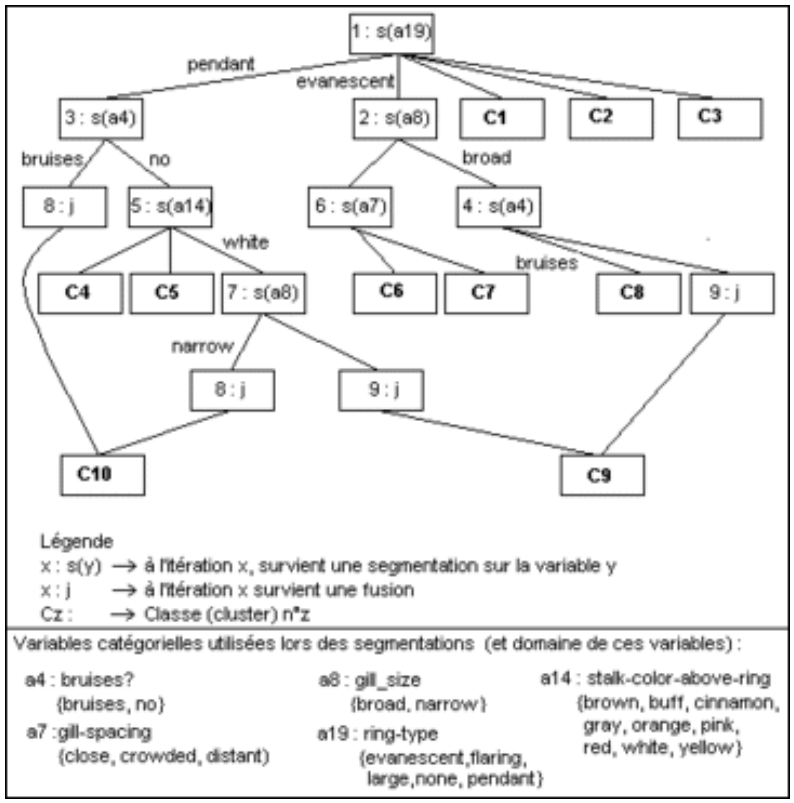


FIG. 3.2 –: *Illustration du Fonctionnement de l'Algorithme*

3.2.3 Evaluation de l'Algorithme de Classification non Supervisée

Classiquement, une méthode de cns s'évalue selon :

- la validité des classifications qu'elle propose,
- la stabilité des classifications qu'elle propose,
- son efficacité algorithmique.

Nous proposons maintenant une évaluation expérimentale de chacun de ces points.

3.2.3.1 Evaluation de la Validité des Classifications

La nature non supervisée de la cns n'autorise pas une définition claire et directe de ce que sont des structures/organisations valides. Ainsi, les mul-

tiples algorithmes de cns se caractérisent par l'ensemble d'hypothèses qu'ils emploient afin de définir les propriétés devant être satisfaites par une structure valide. L'ensemble d'hypothèses déterminant la validité d'une structure n'étant pas universel et différant selon les méthodes, les résultats varient selon la méthode utilisée. Conséquemment, il est essentiel de définir une méthode d'évaluation des structures resultant d'un processus de cns. Ce type d'évaluation est nommé évaluation de la validité d'une cns.

L'évaluation de la validité d'une cns est généralement réalisée par l'utilisation de mesures de validité de cns (voir [HBV01] et le chapitre 4 pour des présentations de l'évaluation de la validité des cns). Ces mesures sont de deux types : externes et internes (les modes d'évaluation habituels correspondant en définitive à l'utilisation implicite d'une mesure externe). Les critères externes de validité évaluent dans quelle mesure le résultat du processus de cns correspond à des connaissances avérées sur les données. De manière assez générale, on admet que ces informations ne sont pas calculables à partir des données. La forme la plus commune de données de ce type est un ensemble d'étiquettes que l'on associe à chacun des objets (ce dernier type d'information peut éventuellement être obtenu par une classification manuelle). Les critères internes de validité consistent quant à eux en une mesure basée uniquement sur le traitement des données servant au processus de cns. Le choix de l'utilisation de l'un ou l'autre type de mesures (pour procéder à l'évaluation de la validité) est essentiellement pragmatique : si l'on dispose d'informations permettant de caractériser la structure devant être extraite, l'utilisation de mesures externes est alors privilégiée tandis qu'en cas d'absence de ce type d'informations l'unique moyen disponible pour l'évaluation de la validité est l'utilisation de mesure internes.

REMARQUES : Si l'utilisation d'une mesure externe n'est pas envisageable en pratique, i.e. lorsqu'il s'agit de traiter un jeu de données dont on ne connaît pas la structure sous-jacente, l'utilisation de ce type de mesure sur un jeux de de données dont on connaît la structure constitue par contre une solution pour l'évaluation de la capacité d'un algorithme donné à découvrir cette structure ou encore pour déterminer la validité des cns obtenues par un algorithme donné.)

Expérience #1 Nous avons utilisé ici une évaluation de type mesure externe largement utilisée dans la littérature. Nous avons considéré le jeu de données mushrooms (composé de champignons comestibles et de champignons vénéneux) et avons réalisé plusieurs cns, pour des valeurs de α différentes, enfin nous avons utilisé le concept "comestibilité" et le taux de correction (T.C.) des cns par rapport à ce concept afin de caractériser la qualité de la cns résultant (la variable définissant la "comestibilité" n'étant évidemment pas introduite dans le processus de cns) (le taux de correction d'une cns par rapport à un concept donné correspond à la pureté de cette cns par rapport à ce concept).

Les résultats obtenus pour 3 valeurs différentes de α ($\alpha = 1,2,3$) sont présentés dans la figure 3.3 ; ils montrent que :

- les différentes classifications réalisées permettent bien d'obtenir des partitions reflétant correctement la structure impliquée par le concept comestibilité,
- la capacité de l'algorithme à déterminer une structure présentant des irrégularités dans le nombre d'objets par classe (voir figures 3.3, 3.4).

(Notons de plus que pour des valeurs de α supérieures à 3, les classifications obtenues présentaient un nombre de classes strictement supérieur à 24, chacune étant homogène du point de vue de la comestibilité des champignons la constituant.)

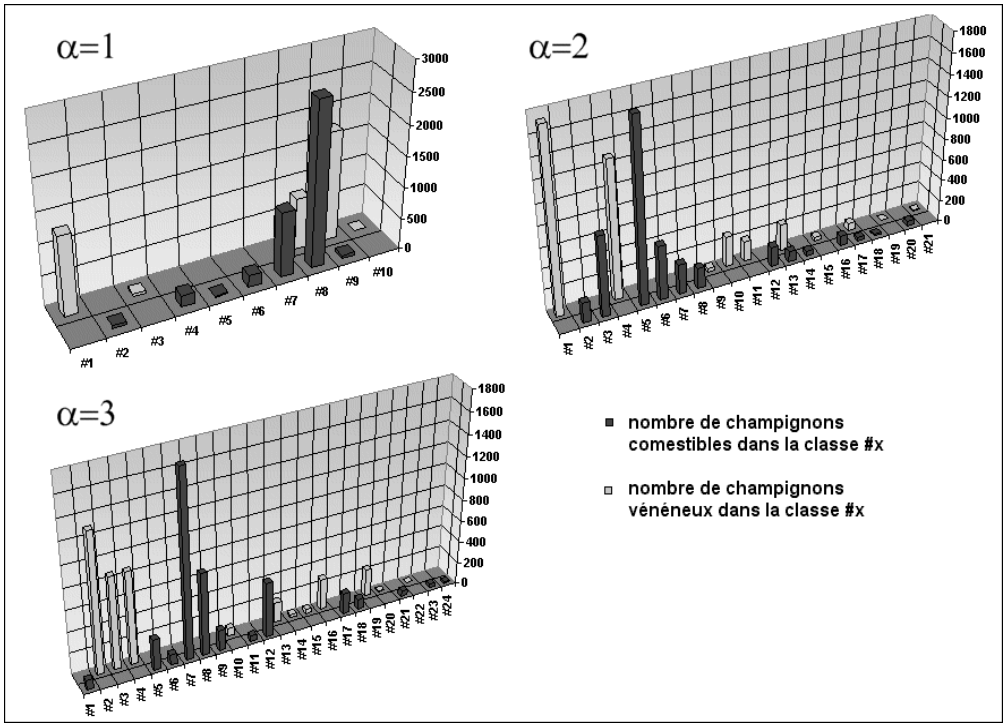


FIG. 3.3 –: *Composition en terme de comestibilité des classes de 3 cns différentes ($\alpha = 1$, $\alpha = 2$, $\alpha = 3$) du jeu de données Mushrooms*

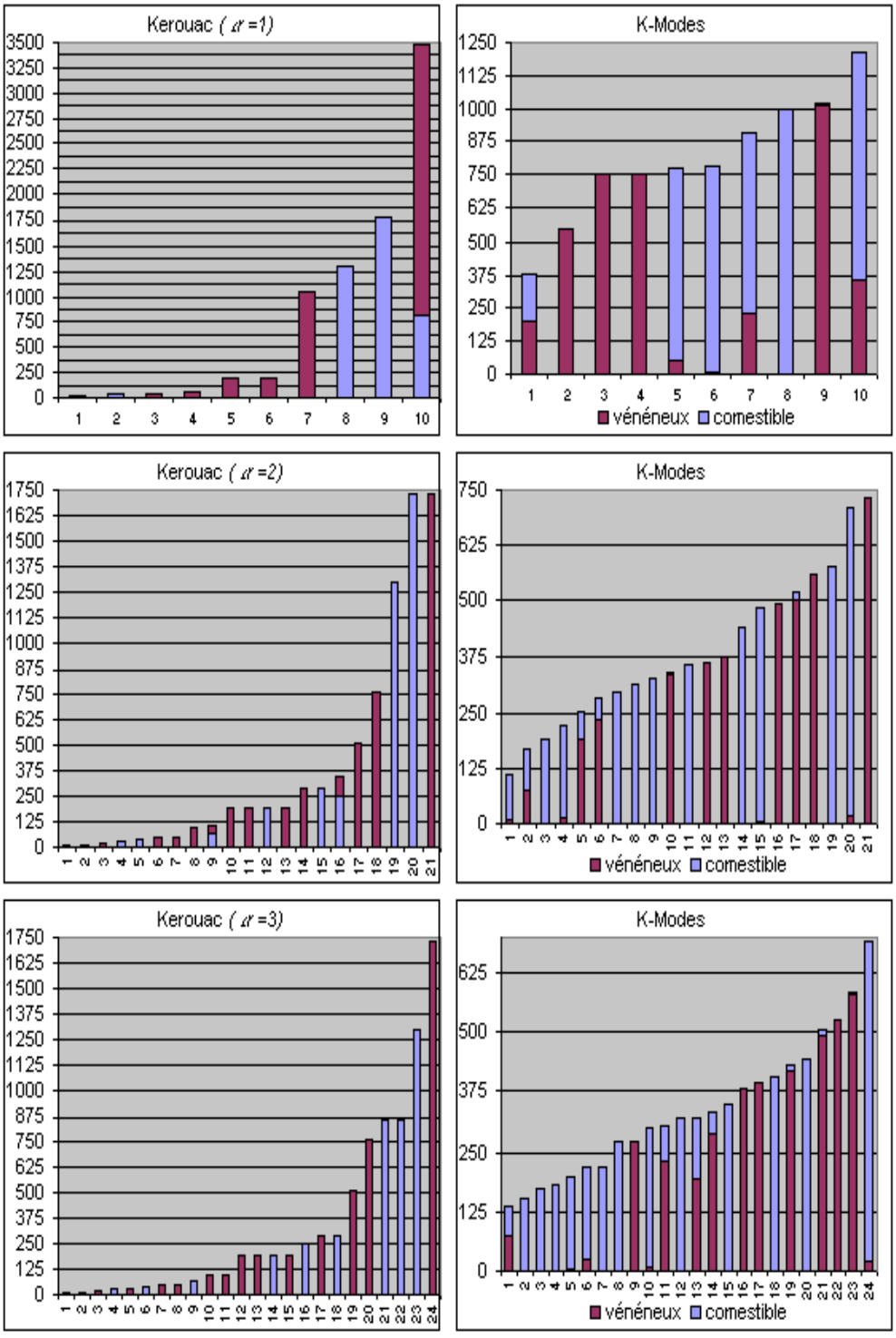


FIG. 3.4 –: *Composition en terme de comestibilité des classes de 6 cns différentes du jeu de données Mushrooms ($\alpha = 1$, $\alpha = 2$, $\alpha = 3$ pour* KEROUAC, *nombre de classes = 10, 21, 24 pour les K-Modes)*

Expérience #2 Nous présentons également une comparaison des résultats obtenus par notre méthode et ceux obtenus par les k-modes pour différentes cns. Nous avons, pour cela, lancé plusieurs processus de cns avec notre méthode en utilisant des facteurs de granularité différents. Cela nous a permis d'obtenir des cns en 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 58, 141 et 276 classes. Les taux de correction de ces cns par rapport au concept "comestibilité" sont ainsi reportés sur la figure 3.5.

Ensuite nous avons lancé des séries de 10 cns en utilisant les k-modes paramétrés de manière telle qu'on obtienne des cns en 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 58, 141 et 276 classes. Les taux de corrections concernant chacune de ces cns sont détaillés sur la figure 3.5. (Notons que nous donnons le taux de correction pour chacune des cns possédant entre 2 et 25 classes et que nous n'indiquons que la moyenne du taux de correction des séries de 10 cns possédant un nombre de classes strictement supérieur à 25. De plus, nous indiquons pour chaque série de 10 cns possédant entre 2 et 25 classes le taux de correction de la "meilleure" cns de la série (i.e. la cns possédant la plus faible valeur pour le critère QKM, voir page 20) pour la définition du critère QKM).

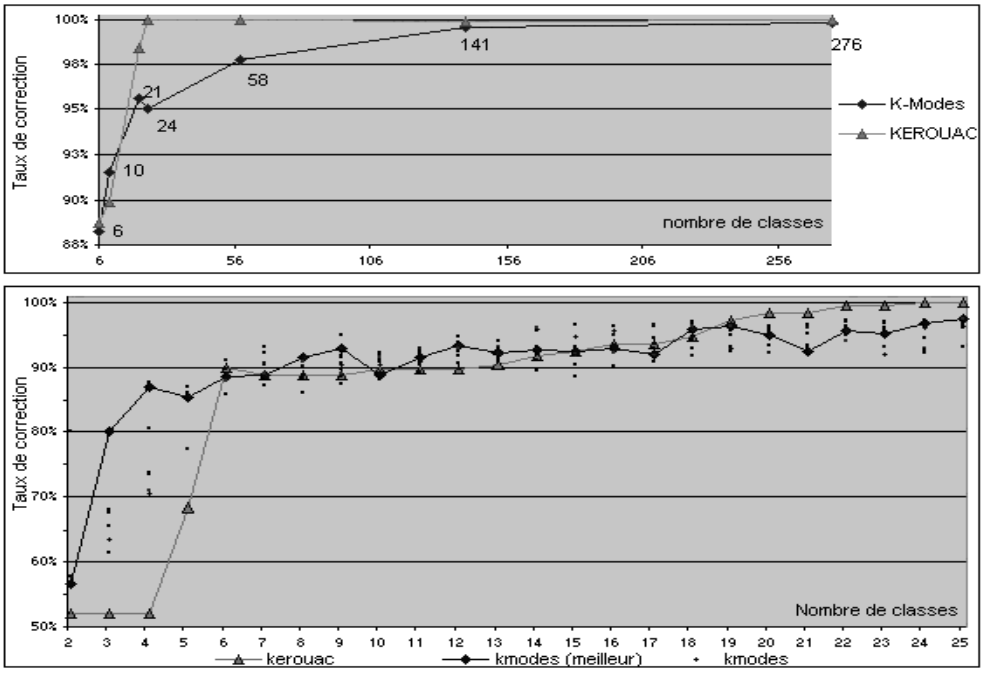


FIG. 3.5 –: *Taux de correction par rapport au concept "comestibilité" pour différentes cns obtenues par application des K-Modes, ou de* KEROUAC

L'ensemble de ces tests montrent une qualité légèrement supérieure pour les cns obtenues par l'intermédiaire de notre méthode. (Ils mettent également en évidence la sensibilité de la méthode K-Modes à l'initialisation : le taux de

correction d'une cns en un nombre donné de classes est susceptible de varier plus ou moins fortement selon l'initialisation de l'algorithme.)

Expérience #3 Des tests ont également été menés sur le jeu de données Soybean Disease [MM96]. Soybean Disease est un jeu de données standard en apprentissage symbolique (machine learning) composé de 47 objets, chacun étant décrit par 35 variables catégorielles. Chaque objet est caractérisé par une des 4 pathologies suivantes : Diaporthe Stem Canker (D1), Charcoal Rot (D2), Rhizoctonia Root Rot(D3), and Phytophthora Rot(D4). A l'exception de D4 qui est représentée par 17 objets, toutes les autres pathologies sont représentées par 10 objets chacune (voir page 217 pour une présentation complète du jeu de données).

Nous avons mené plusieurs cns pour différentes valeurs de α et utilisé le concept "pathologie" pour caractériser la qualité des cns obtenues (la variable "pathologie" n'étant évidemment pas introduite dans le processus de cns). Les résultats obtenus pour 4 valeurs différentes de α ($\alpha = 1$,1.5,2,3) présentés dans la figure 3.6 montrent que les cns obtenues reflètent correctement le concept "pathologie".

Pour $\alpha \geq 3$, les cns ont un nombre de classes strictement supérieur à 4, chaque classe étant homogène du point de vue du concept "pathologie". Nos résultats pour une cns en 4 classes (taux de correction égal à 100%) sont meilleurs que ceux des k-modes reportés dans [Hua97], (taux de correction à peu près égal à 96% pour les k-modes) et à ceux des expérimentations que nous avons menées dont les résultats sont reportés dans la figure 3.7.

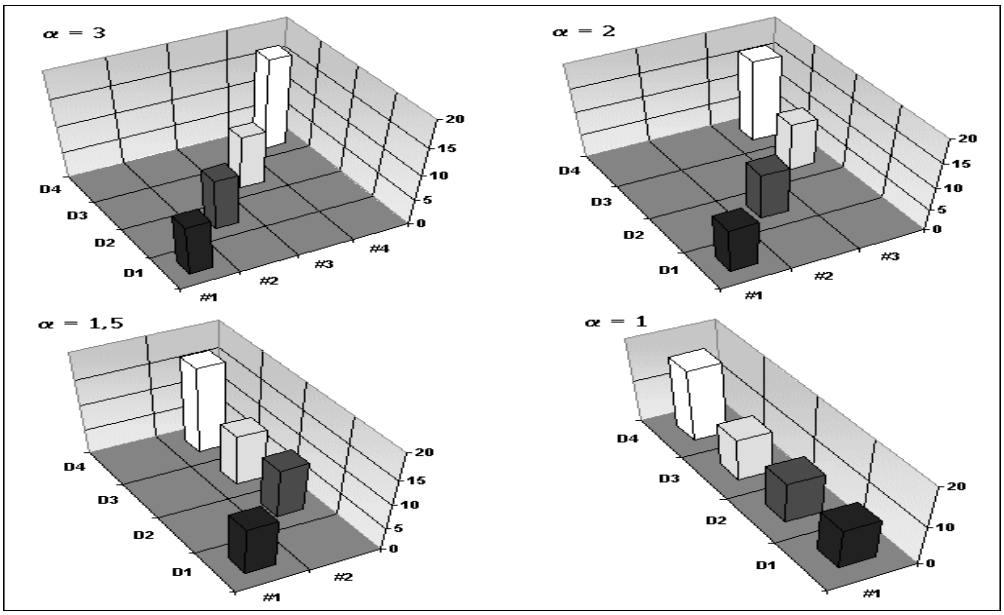


FIG. 3.6 –: *Composition en terme de pathologie des classes de 4 cns différentes ($\alpha = 1$, $\alpha = 1.5$, $\alpha = 2$, $\alpha = 3$) du jeu de données Soybean Diseases*

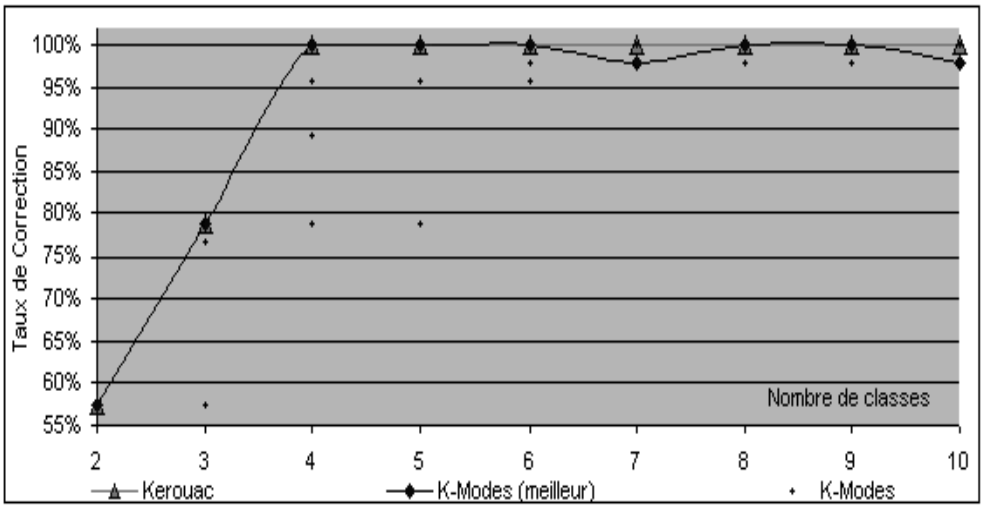


FIG. 3.7 –: *Taux de correction par rapport au concept "pathologie" pour différentes cns obtenues par application des K-Modes, ou de* KEROUAC

Expérience #4 Des expériences impliquant une méthode d'évaluation que nous avons mise au point et présentée au chapitre suivant nous ont permis d'évaluer et de comparer de manière globale la validité de cns obtenues par notre méthode et par la méthode K-Modes. Ces résultats (présentés page 95) semblent indiquer que notre méthode fournit généralement des cns possédant un niveau de validité supérieur à celles obtenues par application des K-Modes[5].

3.2.3.2 Evaluation de la Stabilité

Un autre point d'évaluation d'un algorithme de cns, est l'évaluation de sa stabilité (plus précisément de la stabilité de ses résultats), ce qui signifie à répondre à la question "aurai je obtenu une organisation des objets similaire ou très proche si l'ensemble d'objets que j'avais utilisé avait été légèrement différent (quelques objets supplémentaires ou en moins)?".

Afin de répondre à cette question, des méthodes d'échantillonnage et de comparaison des partitions obtenues ont été présentées, nous utiliserons ici celle présentée dans [LD01], son mode de fonctionnement est le suivant :

– on considère le jeu de données dans son intégralité et l'on réalise une première cns qui constituera la classification de référence P_{Ref} (le nombre d'objets du jeu de données est noté n).

5. Nous ne détaillons pas ici l'analyse des résultats de ces expériences, car ceux-ci sont présentés ultérieurement et nécessitent la présentation de la méthodologie utilisée pour l'évaluation/comparaison de validité de cns

- On réalise ensuite un ensemble EC de p cns (P_i,$i = 1..p$) sur des échantillons (tirés au hasard) du jeu de données de taille $\mu \times n$ ($\mu \in]0,1]$, μ est appelé facteur de dilution).
- Pour chaque cns P_i, $i = 1..p$ on procède à une comparaison avec P_{Ref} afin de calculer la proportion de paires d'objets (notée $prop_i$) traitées différemment par P_{Ref}. On dit qu'une paire d'objets est traitée différemment par P_{Ref} et P_i, si les deux objets sont présents dans l'échantillon ayant permis de bâtir P_i et si ces deux objets sont regroupés au sein d'une même classe dans P_i alors qu'ils ne l'étaient pas pour P_{Ref} ou si ces deux objets ne sont pas regroupés au sein d'une même classe dans P_i alors qu'ils l'étaient pour P_{Ref}. ($prop_i \in [0,1]$)
- On calcule la valeur d'un indicateur de stabilité de la cns $Stab$ ($Stab \in [0,1]$)qui correspond à la moyenne des $prop_i$.

Ainsi, une valeur élevée de $Stab$ (relativement proche de 1) correspondra à une forte différence entre les cns de EC et P_{Ref}, et donc une valeur faible de $Stab$ (proche de 0) correspondra à une faible différence entre les cns de EC et P_{Ref}. La valeur de $Stab$ permet alors de savoir si les résultats de l'algorithme de cns peuvent être considérés comme stables et l'utilisation de l'algorithme valable (la non stabilité impliquant la non utilisabilité de la méthode ou une recherche en profondeur des causes de la non stabilité).

Les tests de stabilité de l'algorithme réalisés sur le jeu de données "Mushrooms" sont présentés dans la figure 3.8 et permettent d'une part d'évaluer la stabilité des cns proposées par KEROUAC et d'autre part d'évaluer la stabilité des cns proposées par la méthode K-Modes. Plus précisément, ces tests correspondent aux expérimentations suivantes :

Pour KEROUAC :

- 11 séries de 25 cns ont été réalisées sur des échantillons aléatoires du jeu de données (un échantillon différent pour chaque cns de chaque série). Chacune des séries correspond à une taille d'échantillon particulière (et donc à une valeur du facteur de dilution). Les différentes tailles d'échantillons sont respectivement 90%, 80%, 70%, 60%, 50%, 40%, 30%, 20%, 10%, 5%, 1% du jeu de données (correspondant respectivement à une valeur de 0.9, 0.8, 0.7, 0.6, 0.5, 0.4, 0.3, 0.2, 0.1, 0.05, 0.01 pour le facteur de dilution). Pour chaque cns le facteur de granularité α a été fixé à 1.
- Une cns de référence a été obtenue en appliquant Kerouac au jeu de données "complet" , la cns ainsi obtenue possède 10 classes. (Notons que, pour un même jeu de données et une même valeur du facteur de granularité, KEROUAC fournit toujours la même cns. Ainsi, pour un facteur de dilution valant 1, la valeur de l'indice de stabilité est donc 0.).
- Puis la valeur de l'indice de stabilité a été évaluée pour chacune de ces cns. Ainsi, pour chaque série de 25 cns, la valeur moyenne de l'indice de stabilité ainsi que son écart-type sont calculés.
- De plus, pour chaque série de 25 cns, le nombre moyen de classes par cns ainsi que son écart-type sont calculés.

Pour la méthode des K-Modes :

- 12 séries de 25 cns ont été réalisées sur des échantillons aléatoires du jeu de données (un échantillon différent pour chaque cns de chaque série). Chacune des séries correspond à une taille d'échantillon particulière (et donc à une valeur du facteur de dilution). Les différentes tailles d'échantillons sont respectivement 100%, 90%, 80%, 70%, 60%, 50%, 40%, 30%, 20%, 10%, 5%, 1% du jeu de données (correspondant respectivement à une valeur de 1, 0.9, 0.8, 0.7, 0.6, 0.5, 0.4, 0.3, 0.2, 0.1, 0.05, 0.01 pour le facteur de dilution). Pour chaque cns le nombre de classes a été fixé à 10 de manière à obtenir des cns possédant le même nombre de classes que la cns de référence obtenue pour les K-Modes. (Le facteur de granularité a, en fait, été fixé à 1 pour Kerouac afin d'obtenir une cns en 10 classes, car pour ce jeu de données les auteurs utilisant les K-Modes indiquent qu'il s'agit du "meilleur" nombre de classes.)
- Une cns de référence a été obtenue en sélectionnant parmi les 25 cns de la série de cns sur le jeu de données "complet" (taille de l'échantillon = 100% du jeu de données), la cns possédant la plus faible valeur du critère QKM (critère à minimiser sous-jacent à la méthode des K-Modes) (Cette cns peut donc être considérée comme la meilleure cns obtenue, par application des K-Modes, sur le jeu de données complet).
- Puis la valeur de l'indice de stabilité a été évaluée pour chacune de ces cns. Ainsi, pour chaque série de 25 cns, la valeur moyenne de l'indice de stabilité ainsi que son écart-type est calculée.
- De plus, pour chaque série de 25 cns, nous avons recherché la cns correspondant à la "meilleure" cns (i.e. celle possédant la plus faible valeur pour le critère QKM) et indiqué sa valeur pour l'indice de stabilité.

Les résultats de ces expériences sont présentés sur la figure 3.8. Notons que la valeur de l'indice de stabilité est indiquée en pourcentage. Ainsi exprimée, elle indique le pourcentage de couples d'objets traités différemment par la cns de référence et les cns pour lesquelles la stabilité est évaluée.

Ces tests montrent une excellente stabilité des cns obtenues par notre méthode : quel que soit le niveau d'échantillonnage ($\geq 1\%$), moins de 0.5% des couples d'objets sont, en moyenne, traités différemment par la cns de référence et par les cns évaluées. Comparativement, la méthode des K-Modes est, quant à elle, à peu près 20 fois moins efficace...

La moindre efficacité des K-Modes peut s'expliquer d'une part par sa sensibilité au processus d'initialisation et d'autre part par une efficacité intrinsèque moins bonne due à la mesure de similarité qu'elle utilise. Cette dernière raison est sans doute la principale : si l'on étudie la stabilité associée aux "meilleures" cns de chaque série on peut observer que bien que l'on observe un accroissement global du niveau de stabilité, celui-ci reste toutefois à peu près 16 fois moins bon que le niveau de stabilité de notre méthode.

Notons également que, naturellement le niveau de stabilité est sensible au niveau de dilution mais qu'il apparaît toutefois très stable. Enfin, la stabilité du

point de vue du nombre de classes de chacune des cns obtenues par KEROUAC est également excellente :

- pour des niveaux de dilution supérieurs à 0.2 le nombre de classes de chacune des cns obtenues est quasiment toujours égal au nombre de classes de la cns de référence ;
- pour des niveaux de dilution inférieurs à 0.2 le nombre de classes de chacune des cns obtenues est en général quelque peu plus faible que le nombre de classes de la cns de référence, cela pouvant s'expliquer par l'éventuelle quasi-absence dans l'échantillon traité d'objets appartenant à l'une des 10 classes de la cns de référence (en effet, pour ces niveaux de dilution la taille des échantillons d'objets traités est inférieure à 20% du jeu de données complet, or 6 classes de la cns de référence comporte moins de 2.5% des individus).

Ce dernier point (la stabilité du nombre de classes par cns) montre la réelle supériorité de notre méthode par rapport à celle des K-Modes : en effet si l'on avait pu penser que la différence de stabilité entre les méthodes pouvait peut-être s'expliquer par la forme de la cns de référence (qui dans le cas de KEROUAC présente 4 classes regroupant la quasi-totalité des individus, voir figure 3.4), la stabilité du nombre de classes par cns permet de rejeter cet argument.

REMARQUES : Il est clair que les tests de stabilité des algorithmes ne donnent pas uniquement lieu à une évaluation de la stabilité des algorithmes et qu'ils prennent également en compte la nature intrinsèque des données à permettre la mise à jour de structure stable. Cependant, la différence de stabilité des algorithmes KEROUAC et K-Modes sur un même jeu de données (Mushrooms) (voir figure 3.4) montre bien que ce type de test permet véritablement de caractériser la plus grande d'un stabilité d'un algorithme par rapport à un autre...

3.2.3.3 Evaluation de l'Efficacité Algorithmique

Nous ne présentons pas ici une étude poussée du coût calculatoire de la méthode, nous nous contentons de préciser que la complexité algorithmique de notre méthode est équivalente à celle des graphes d'induction largement utilisés en E.C.D.[6] et présentons le temps de calcul associé à différentes cns pour le jeu de données "mushrooms". En fait, nous ne présentons pas explicitement les temps de calcul associés aux cns mais le rapport suivant :
$R = \frac{temps\ de\ calcul\ associé\ à\ la\ cns}{temps\ de\ calcul\ associé\ à\ la\ cns\ en\ 6\ classes\ par\ les\ K-modes}$.

Les rapports présentés pour les K-modes sont les valeurs moyennes obtenues pour des séries de 10 cns. Ces résultats montrent que les K-Modes (qui sont reconnus comme une méthode rapide et possédant une bonne scalabilité) sont plus rapides pour de faibles nombres de classes mais que les 2 méthodes semblent se comporter de manière similaire pour des nombres de classes plus

6. la méthode possède donc une complexité log-linéaire selon le nombre d'objets du jeu de données traité, et linéaire selon le nombre de variables du jeu de données

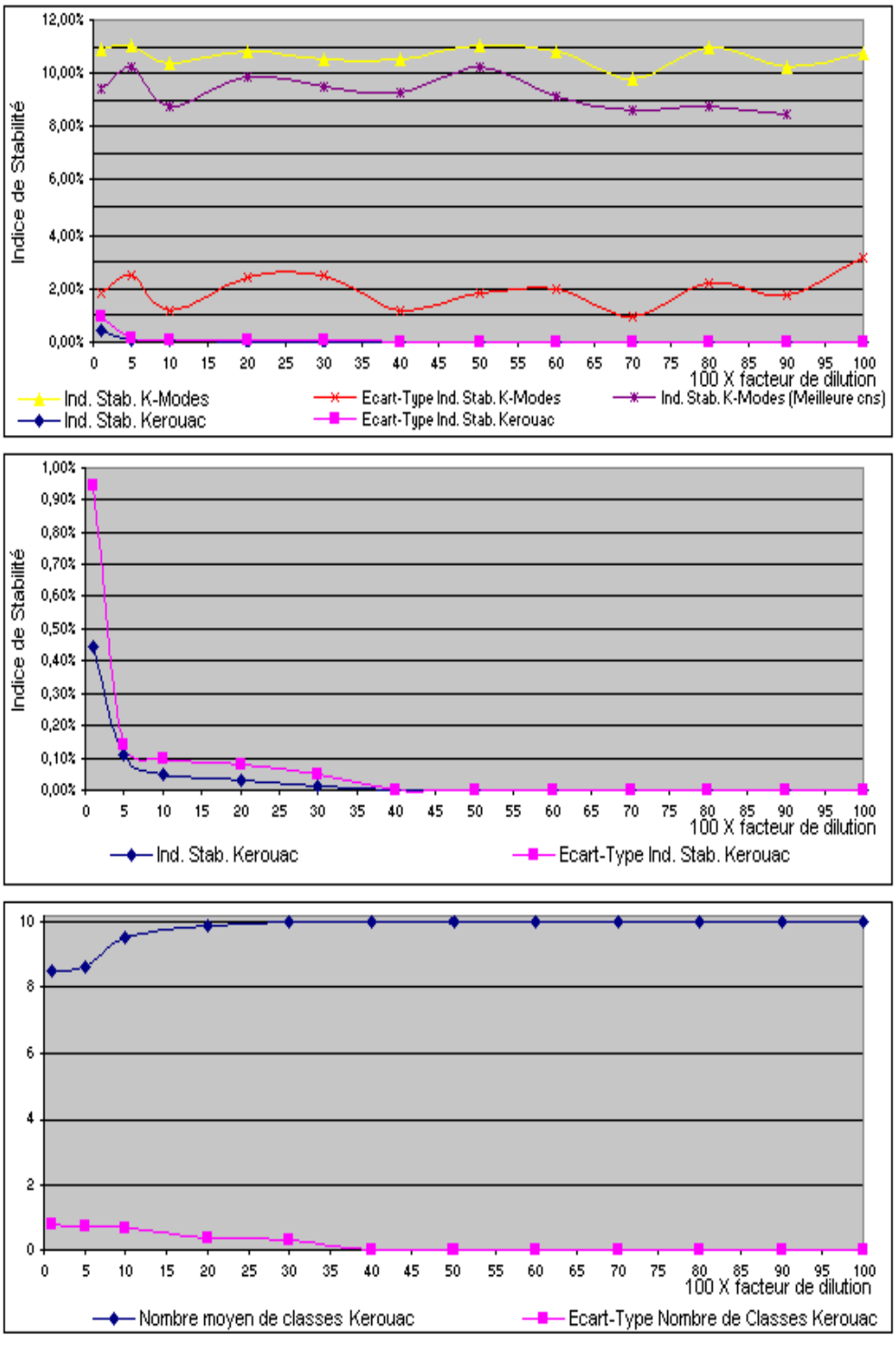


FIG. 3.8 –: *Evaluation de la stabilité pour le jeu de données Mushrooms*

élevés. En définitive, en observant les résultats expérimentaux et en y associant la complexité algorithmique théorique, on peut conclure que KEROUAC présente une efficacité algorithmique tout à fait correcte.

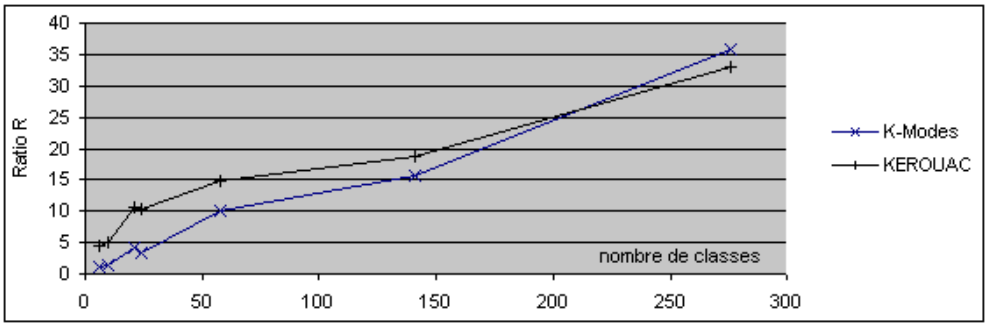


FIG. 3.9 –: *Rapports R associés à différentes cns*

3.2.4 Eléments Additionnels

Nous venons de procéder à l'évaluation de notre méthode de cns qui nous a permis de mettre en avant la validité des structures découvertes, la très bonne stabilité et le coût calculatoire relativement faible de la méthode. Ces différents points constituent plusieurs points forts, tout comme les avantages concernant l'utilisabilité cité à la section 3.2.2.4. Nous listons un ensemble d'autres points que nous allons maintenant aborder et qui participent également à rendre cette méthode très attrayante du point de vue de l'utilisateur :

- la présence de données manquantes n'est pas gênante : leurs conséquences sur la classification est complètement paramétrable en codant l'implication particulière de la présence de données manquantes sur l'aspect naturel d'une partition,
- l'introduction de contraintes est possible par l'intermédiaire de l'utilisation de variables supplémentaires sur lesquelles on n'autorisera pas de segmentation lors du processus de recherche de la partition naturelle approchée (Ainsi, l'interactivité entre utilisateur et processus de cns est possible, par introduction de contraintes).

Nous explicitons maintenant comment les traitements des données manquantes et contraintes sont facilités grâce à notre méthode et par l'intermédiaire de l'introduction de valeurs spécifiques dans les domaines des variables catégorielles.

3.2.4.1 Valeurs Spécifiques pour le Domaine des Variables Catégorielles

Dans certaines situations, il peut apparaître nécessaire de modifier l'implication sur l'aspect naturel d'une partition que peut avoir une valeur particulière

du domaine d'une variable catégorielle (i.e. il peut être parfois intéressant que les opérateurs δ_{sim} et δ_{dissim} aient un comportement spécifique en cas de présence de valeurs particulières du domaine d'une variable catégorielle).

Considérons par exemple le cas de valeurs manquantes (notée ?) pour une variable catégorielle V_i. On peut déterminer plusieurs types d'implication sur l'aspect naturel d'une partition pour cette valeur particulière du domaine de V_i selon "le type d'information qu'elle véhicule", par exemple :

- Considérons qu'une variable décrive le nombre de roues d'un véhicule. Une valeur manquante pour cette variable signifie alors l'absence de roue, ce qui est vecteur d'information, si bien que cette valeur manquante peut être considérée comme une valeur classique. Dès lors si $o_{l_i} =?$ et $o_{m_i} =?$ alors $\delta_{sim}(o_{a_i},o_{b_i}) = 1 - \delta_{dissim}(o_{a_i},o_{b_i}) = 1$; par contre si $o_{l_i} =?$ et $o_{m_i} \neq?$ alors $\delta_{sim}(o_{a_i},o_{b_i}) = 1 - \delta_{dissim}(o_{a_i},o_{b_i}) = 0$. Ainsi l'implication sur l'aspect naturelle de cette valeur particulière est identique au cas classique.
- Considérons maintenant une variable quelconque et que la présence d'une valeur manquante soit la conséquence d'une erreur de saisie. Pourquoi considérerions nous alors que un objet présentant une valeur manquante pour cette variable est similaire à un autre objet présentant également une valeur manquante pour cette variable... ou encore, quelle bonne raison pourrait nous pousser à le considérer dissimilaire d'un objet pour lequel il n'y a pas eu d'erreur de saisie ? La valeur manquante ne peut alors être considérée comme une valeur classique, on peut ainsi vouloir qu'un objet présentant cette valeur pour V_i, ne soit considéré ni similaire ni dissimilaire des autres objets du point de vue de V_i. Dans ce cas, si deux objets o_a et o_b sont tels que $o_{a_i} =?$ et $o_{b_i} \neq?$ on obtient : $\delta_{sim}(o_{a_i},o_{b_i}) = \delta_{dissim}(o_{a_i},o_{b_i}) = 0$; de même si deux objets o_l et o_m sont tels que $o_{a_i} =?$ et $o_{b_i} =?$ on obtient : $\delta_{sim}(o_{a_i},o_{b_i}) = \delta_{dissim}(o_{a_i},o_{b_i}) = 0$. Ainsi l'implication sur l'aspect naturel est particulière dans la mesure où la présence de valeurs manquantes aura pour conséquence de ne pas impliquer le regroupement ou la séparation des objets présentant cette valeur particulière et de ne pas impliquer la séparation ou le les regroupement des objets ne la présentant pas.
- ... (on peut imaginer d'autres cas)

Nous avons défini (au chapitre 2 un ensemble de valeurs additionnelles particulières pouvant être ajoutées aux domaines des variables catégorielles de manière à "coder" des cas particuliers comme celui de la présence de valeurs manquantes. Ces valeurs particulières $\varepsilon_1,\varepsilon_2,\varepsilon_3,\varepsilon_4,\varepsilon_5,\varepsilon_6,\varepsilon_7$ sont inclues dans l'ensemble E_ε, elles permettent de coder un ensemble relativement vaste de cas particuliers dont nous allons donner quelques exemples dans le cas de leur utilisation pour le codage de données manquantes ou de contraintes (il est toutefois tout à fait possible d'étendre l'ensemble avec d'autres valeurs).

Nous présentons dans les tableaux 3.2, 3.3, 3.4, les valeurs pour $\delta_{sim}(o_{a_i},o_{b_i})$, $(1-\delta_{dissim}(o_{a_i},o_{b_i}))$, $\delta_{dissim}(o_{a_i},o_{b_i})$ dans différents cas. Tout d'abord dans le cas

classique où les valeurs prises par o_{a_i} et o_{b_i} sont des valeurs ne correspondant pas à des valeurs spécifiques mais aux différentes modalités (A,B,C,D) d'une variable V_i (voir tableau 3.2). Puis dans le cas où les valeurs prises par o_{a_i} et o_{b_i} sont soit l'une des 4 modalités possibles de V_i que l'on note λ ou bien l'une des valeurs spécifiques que nous introduisons (voir tableaux 3.3, 3.4).

Le tableau 3.2 expose le "comportement" classique de $\delta_{dissim}(o_{a_i},o_{b_i})$ et $\delta_{sim}(o_{a_i},o_{b_i})$ pour une variable V_i c'est à dire lorsque les valeurs de cette variable sont disponibles à la fois pour o_a et o_b (i.e. $\forall i$, $o_{a_i} \neq \varepsilon_i$ et $\forall i$, $o_{b_i} \neq \varepsilon_i$; les tableaux 3.3 et 3.4) montrent, quant à eux, le comportement de $\delta_{dissim}(o_{a_i},o_{b_i})$ et $\delta_{sim}(o_{a_i},o_{b_i})$ pour une variable V_i lorsque l'une ou les 2 valeurs de V_i est inclue dans E_ε.

3.2.4.2 Gestion des Valeurs Manquantes :

Nous l'avons illustré, il peut être intéressant de réserver un traitement particulier aux valeurs manquantes et cela peut être réalisé grâce à l'utilisation des valeurs spécifiques introduites précédemment (ε_1,ε_2,ε_3,ε_4,ε_5,$\varepsilon 6$,ε_7). Ainsi, si l'on veut, par exemple, qu'un objet o_a présentant une valeur manquante pour une variable V_i, ne soit considéré ni similaire ni dissimilaire des autres objets du point de vue de V_i (i.e. quel que soit un autre objet quelconque et quelle que soit sa valeur pour V_i, cet objet ne sera considéré ni comme similaire ni comme dissimilaire de o_a) on peut utiliser la valeur spécifique ε_4 pour coder la valeur manquante. En effet dans ce cas, si $o_{a_i} = \varepsilon_4$, alors quelle que soit la valeur o_{b_i} on obtient : $\delta_{sim}(o_{a_i},o_{b_i}) = \delta_{dissim}(o_{a_i},o_{b_i}) = 0$. Ainsi l'implication sur l'aspect naturel est particulière dans la mesure où la présence de valeurs manquantes aura pour conséquence de ne pas impliquer le regroupement ou la séparation des objets présentant cette valeur particulière et de ne pas impliquer la séparation ou le regroupement des objets ne la présentant pas.

3.2.4.3 Introduction de Contraintes :

La cns est une tâche subjective par nature : le même jeu de données peut nécessiter différents partitionnements afin de répondre à diverses attentes de l'utilisateur. Considérons par exemple les animaux suivants : éléphants, baleines et thons [Wat85]. Les éléphants tout comme les baleines sont des mammifères et peuvent par conséquents former une classe. Cependant, si l'utilisateur est intéressé par une classification basée sur l'habitat naturel de ces animaux, baleines et thons formeront alors une classe. Il peut donc parfois être utile d'introduire cette notion de subjectivité dans le processus de cns.

Ainsi, si lorsque l'on procède à une cns on ne sait pas par avance le résultat que l'on désire obtenir, il peut arriver que l'on ait toutefois des idées sur la forme du résultat voulu. On peut en effet posséder un ensemble de connaissances qui nous poussent par exemple à vouloir que plusieurs objets soient regroupés au sein d'une même classe, ou, a contrario, que certains objets appartiennent à des classes différentes. Afin d'obtenir un résultat conforme à

$\delta_{sim}(o_{a_i},o_{b_i})$

$o_a \backslash o_b$	A	B	C	D
A	1	0	0	0
B	0	1	0	0
C	0	0	1	0
D	0	0	0	1

$\delta_{dissim}(o_{a_i},o_{b_i})$

$o_a \backslash o_b$	A	B	C	D
A	0	1	1	1
B	1	0	1	1
C	1	1	0	1
D	1	1	1	0

$1 - \delta_{dissim}(o_{a_i},o_{b_i})$

$o_a \backslash o_b$	A	B	C	D
A	1	0	0	0
B	0	1	0	0
C	0	0	1	0
D	0	0	0	1

TAB. 3.2 –: *Comportement des opérateurs $\delta_{sim}(o_{a_i},o_{b_i})$, $\delta_{dissim}(o_{a_i},o_{b_i})$, $1 - \delta_{dissim}(o_{a_i},o_{b_i})$ pour des valeurs classiques*

$o_a \backslash o_b$	λ	ε_1	ε_2	ε_3	ε_4	ε_5	ε_6	ε_7
λ	1	0	0	0	0	0	0	0
ε_1	0	1	0	0	0	0	0	0
ε_2	0	0	$0^\star$	0	0	0	0	0
ε_3	0	0	0	1	0	0	0	0
ε_4	0	0	0	0	$0^\star$	0	0	0
ε_5	0	0	0	0	0	$0^\star$	0	0
ε_6	0	0	0	0	0	0	$0^\star$	0
ε_7	0	0	0	0	0	0	0	$0^\star$

TAB. 3.3 –: *Comportement de l'opérateur $\delta_{sim}(o_{a_i},o_{b_i})$ pour des valeurs particulières*

o_a o_b	λ	ε_1	ε_2	ε_3	ε_4	ε_5	ε_6	ε_7
λ	0	1	1	$0^\star$	$0^\star$	1	$0^\star$	1
ε_1	1	0	1	$0^\star$	$0^\star$	1	$0^\star$	1
ε_2	1	1	0	1	$0^\star$	1	$0^\star$	$0^\star$
ε_3	$0^\star$	$0^\star$	1	0	$0^\star$	1	$0^\star$	1
ε_4	$0^\star$	$0^\star$	$0^\star$	$0^\star$	0	1	$0^\star$	$0^\star$
ε_5	1	1	1	1	1	$1^\star$	$0^\star$	1
ε_6	$0^\star$	$0^\star$	$0^\star$	$0^\star$	$0^\star$	$0^\star$	$1^\star$	$0^\star$
ε_7	1	$0^\star$	1	1	$0^\star$	1	$0^\star$	0

TAB. 3.4 –: *Comp. de l'opérateur $\delta_{dissim}(o_{a_i},o_{b_i})$ pour des valeurs particulières*

	V_E	V_1	V_2	V_3	V_4	O_{app}	O_{cont}	O_{cont_1}	O_{cont_2}	V_{cont_1}	V_{cont_2}
1	A	A	A	A	A	x	x	x	-	ε_2	ε_7
2	A	A	A	B	A	x	-	-	-	ε_4	ε_4
3	A	A	C	B	B	-	-	-	-	ε_4	ε_4
4	A	A	A	A	A	-	-	-	-	ε_4	ε_4
5	A	A	C	A	B	x	x	x	-	ε_2	ε_7
6	A	A	C	B	B	-	-	-	-	ε_4	ε_4
7	A	B	B	A	A	x	x	x	-	ε_2	ε_7
8	A	B	A	A	A	-	-	-	-	ε_4	ε_4
9	A	B	C	A	A	x	x	x	-	ε_2	ε_7
10	A	B	B	A	A	x	-	-	-	ε_4	ε_4
11	B	A	B	A	B	x	x	-	x	ε_7	ε_2
12	B	A	B	B	A	-	-	-	-	ε_4	ε_4
13	B	A	B	A	B	-	-	-	-	ε_4	ε_4
14	B	B	C	B	A	x	x	-	x	ε_7	ε_2
15	B	B	C	B	B	-	-	-	-	ε_4	ε_4
16	B	B	C	B	B	-	-	-	-	ε_4	ε_4
17	B	B	A	B	A	-	-	-	-	ε_4	ε_4
18	B	B	B	A	B	x	x	-	x	ε_7	ε_2
19	B	B	B	A	B	-	-	-	-	ε_4	ε_4
20	B	B	B	A	B	x	x	-	x	ε_7	ε_2

TAB. 3.5 –: *Jeu de données synthétique*

ces attentes imposer des contraintes que doit respecter la cns résultat constitue une bonne solution. Toutefois, nombre des méthodes existantes s'adaptent mal à l'adjonction de contraintes. Nous montrons que la méthode que nous proposons peut, elle, utiliser la notion de contraintes sans grande difficulté.

En fait, nous allons montrer qu'une contrainte peut être modélisée sous la forme d'une variable dont on pondère l'influence sur la valeur de l'aspect naturel d'une partition. Nous commençons par présenter un exemple illustrant comment nous pourrons introduire des contraintes dans le processus de clustering, puis formalisons cette notion de contrainte.

Illustration Considérons le jeu de données du tableau 3.5, (ce jeu de données est constitué de 20 objets, de 4 variables exogènes (V_1,V_2,V_3,V_4) et d'une variable endogène (V_E). Admettons que l'on veuille réaliser une cns et que nous désirions que cette cns présente la particularité de ne pas réunir d'objets de $O_{cont1} = \{o_1,o_5,o_7,o_9\}$ avec des objets de $O_{cont2} = \{o_{11},o_{14},o_{18},o_{20}\}$ (voir tableau 3.5). Pour cela, nous allons introduire une contrainte, et ce, sous la forme d'une nouvelle variable V_{cont_1} (voir tableau 3.5) à laquelle on associera une pondération p_1 afin de mettre en adéquation son effet sur la valeur de l'aspect naturel d'une partition et son objectif de séparation des objets de O_{cont1}

et de ceux de O_{cont2}. Cette variable possède 3 modalités différentes ε_4, ε_2, ε_7, on assigne aux objets de O_{cont1} la valeur de modalité ε_2, on assigne aux objets de O_{cont2} la valeur de modalité ε_7, on assigne aux objets restants la valeur de modalité ε_4 (voir tableau 3.5). Ainsi, lors du calcul du critère NCC :

- les partitions présentant des objets de O_{cont1} et de O_{cont2} au sein d'une même classe seront pénalisées du point de vue de cette nouvelle variable (par exemple, si o_1 et o_{11} sont réunis au sein d'une même classe, l'homogénéité interne de la classe est pénalisée puisque :
  $1 - \delta_{dissim}(o1_{V_{cont_1}}, o11_{V_{cont_1}}) = 1)$;
- par contre mélanger des objets de O_{cont1} avec des objets n'appartenant ni à O_{cont1} ni à O_{cont2} ne sera pas pénalisant , enfin séparer les objets de O_{cont1} (resp. O_{cont2}) (resp. les objets restants) n'est absolument pas pénalisant du point de vue de cette variable (par exemple si o_1 et o_5, appartiennent à des classes différentes l'hétérogénéité entre ces deux classes n'est pas pénalisée puisque :
  $\delta_{sim}(o1_{V_{cont_1}}, o5_{V_{cont_1}}) = \delta_{sim}(\varepsilon_2, \varepsilon_2) = 0)$
  (resp. si o_{11} et o_{14} appartiennent à des classes différentes l'hétérogénéité entre ces deux classes n'est pas pénalisée puisque :
  $\delta_{sim}(o11_{V_{cont_1}}, o14_{V_{cont_1}}) = \delta_{sim}(\varepsilon_7, \varepsilon_7) = 0)$
  (resp. si o_2 et o_{12} appartiennent à des classes différentes l'hétérogénéité entre ces deux classes n'est pas pénalisée puisque :
  $\delta_{sim}(o2_{V_{cont_1}}, o12_{V_{cont_1}}) = \delta_{sim}(\varepsilon_4, \varepsilon_4) = 0)$).

La pondération p_1 correspond à un facteur multiplicateur de l'effet de la variable sur le critère NCC : son effet sera alors équivalent à celui de p_1 variables identiques à elle. Notons enfin que cette variable ne sera pas utilisée pour les processus de segmentation ayant lieu lors du processus de cns. (nous fixons ici $p_1 = 64$).

Le résultat du processus de cns avec adjonction des contraintes (et $\alpha = 1$) est présenté sur la figure 3.11, le résultat du processus de cns sans contrainte (et $\alpha = 1$) est présenté sur la figure 3.10.

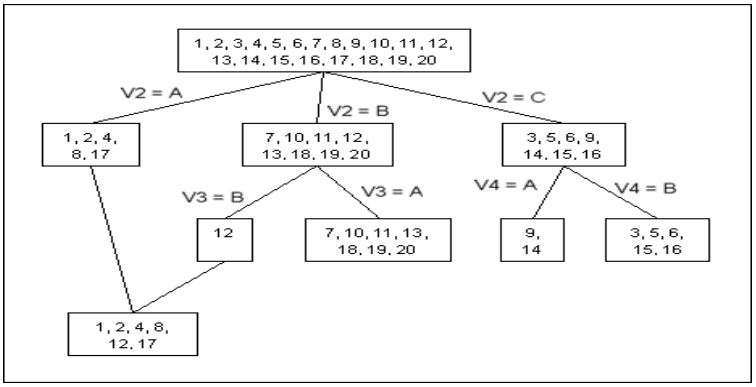


FIG. 3.10 –: *Illustration du processus de cns sans contrainte*

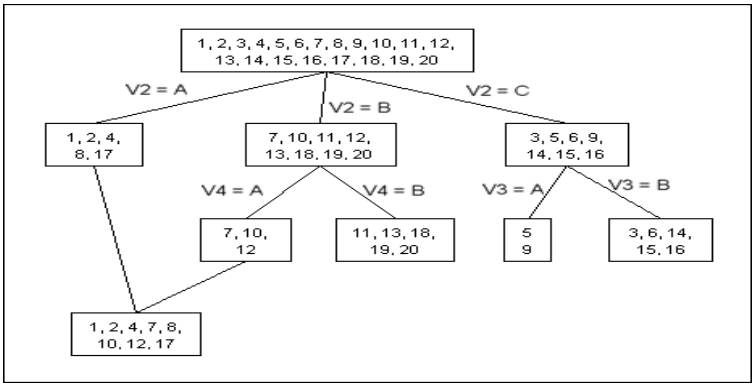


FIG. 3.11 –: *Illustration du processus de cns avec contrainte*

Formalisation Nous formalisons maintenant l'ensemble des éléments qui permettent l'introduction de contraintes dans un processus de cns.

Définition 7 *Contrainte : Une contrainte est modélisée par une variable supplémentaire, qui ne peut servir pour la segmentation de classes lors du processus de segmentation et dont l'influence dans la valeur de l'aspect naturel est pondérée. Ainsi, le calcul de l'aspect naturel d'une partition est le suivant dans le cas de la présence de contraintes.*

Soient

$O = \{o_i, i = 1..n\}$ *l'ensemble des objets du jeu de données*

$EV = \{V_j, j = 1..p\}$ *l'ensemble des variables du jeu de données*

$EV_C = \{Vc_j, j = 1..pc\}$ *l'ensemble des contraintes auxquelles sont associées une pondération* p_j *pour chacune d'entre elles*

$o_i = \{o_{ij}, j = 1,..,p,p+1,...,p+pc\}$ *un objet de O (les p premières valeurs d'attributs correspondant aux variables les pc suivantes aux contraintes)*

C_k *un ensemble d'objets de O*

$P_h = \{C_1,...,C_z\}$ *une partition de O en z groupes*

$Q(P_h)$ *la mesure de l'aspect naturel d'une partition* P_h

$$Q(P_h) = \sum_{i=1..z, j=1..z, j<i} Sim(C_i, C_j) + \alpha \times \sum_{i=1}^{z} Dissim(C_i) \tag{3.6}$$

α *un scalaire (fixé par l'utilisateur)*

$$Sim(C_i, C_j) = \sum_{o_a \in C_i, o_b \in C_j} sim(o_a, o_b) \tag{3.7}$$

$$Dissim(P_i) = \sum_{o_a \in C_i, o_b \in C_i} dissim(o_a, o_b) \tag{3.8}$$

$$sim(o_a,o_b) = \sum_{i=1}^{p} \delta_{sim}(o_{a_i},o_{b_i}) + \sum_{i=p+1}^{p+pc} p_i \delta_{sim}(o_{a_i},o_{b_i}) \tag{3.9}$$

$$dissim(o_a,o_b) = \sum_{i=1}^{p} \delta_{dissim}(o_{a_i},o_{b_i}) + \sum_{i=p+1}^{p+pc} p_i(\delta_{dissim}(o_{a_i},o_{b_i})) \tag{3.10}$$

$$\delta_{sim}(o_{a_i},o_{b_i}) = \begin{cases} 1 \textit{ si } o_{a_i} = o_{b_i} \\ 0 \textit{ si } o_{a_i} \neq o_{b_i} \\ \textit{cas particuliers si } o_{a_i} \in E_\varepsilon \textit{ et/ou } o_{b_i} \in E_\varepsilon \end{cases} \tag{3.11}$$

$$\delta_{dissim}(o_{a_i},o_{b_i}) = \begin{cases} 0 \textit{ si } o_{a_i} = o_{b_i} \\ 1 \textit{ si } o_{a_i} \neq o_{b_i} \\ \textit{cas particuliers si } o_{a_i} \in E_\varepsilon \textit{ et/ou } o_{b_i} \in E_\varepsilon \end{cases} \tag{3.12}$$

Définition 8 *Pondération d'une contrainte : lors de l'établissement d'une contrainte on peut désirer que celle ci soit toujours respectée, parfois respectée, ou si possible respectée ; en clair, on peut définir une sorte de niveau d'exigence du respect d'une contrainte. Ce niveau d'exigence du respect d'une contrainte sera modélisé par ce que nous appelons pondération d'une contrainte. Cette pondération est un scalaire* $p \in [0,+\propto [$*. Plus le scalaire est grand plus l'exigence de respect de la contrainte est forte.*

Exemples de Contraintes Nous donnons ici quelques exemples de type de contraintes ainsi que leur modélisation :

- *Contraintes du type "Aucun objet de l'ensemble* O_1 *ne doit se retrouver dans la même classe qu'un objet de l'ensemble* O_2 *"*. Ces contraintes se modélisent de la manière suivantes : soit O l'ensemble des objets du jeu de données $O = O_1 \bigcup O_2 \bigcup O_r$ ($O_r = \{o_i \in O, o_i \notin O_1, o_i \notin O_2\}$), on ajoute au jeu de données une variable contrainte V_c associée à une pondération p_c telle que :
  - les objets de O_1 aient ε_2 pour valeur pour V_c,
  - que les objets de O_2 aient ε_7 pour valeur pour V_c
  - les objets de O_r aient ε_4 pour valeur pour V_c.
- *Contraintes du type "Les objets de l'ensemble* O_1 *doivent se retrouver dans la même classe que les objets de l'ensemble* O_2 *"*. Ces contraintes se modélisent de la manière suivantes :soit O l'ensemble des objets du jeu de données $O = O_1 \bigcup O_2 \bigcup O_r$, on ajoute au jeu de données une variable contrainte V_c associée à une pondération p_c telle que :
  - les objets de O_1 et O_2 aient une valeur classique commune pour V_c,
  - les objets de O_r aient ε_4 pour valeur pour V_c.

3.2.4.4 De l'Apprentissage Non Supervisé à l'Apprentissage Supervisé : l'Apprentissage Non Supervisé sous Contraintes

L'intégration de contraintes au processus d'apprentissage non supervisé peut mener à l'établissement d'une méthode d'apprentissage supervisé ou plutôt semi-supervisé [CCM00] [DBE99] dont l'intérêt est notamment de pouvoir permettre la réalisation d'apprentissage supervisé lorsque l'on dispose uniquement d'un faible nombre d'exemples étiquetés mais également d'assurer dans certains cas une plus grande stabilité au modèle d'apprentissage bâti [CCM00].

Nous exposons rapidement comment la méthode d'apprentissage non supervisé que nous venons de proposer peut être transformée en une méthode d'apprentissage semi-supervisé relativement efficace[7].

L'idée sous jacente est simple : "pourquoi ne pas réaliser un apprentissage non supervisé par l'intermédiaire de notre méthode en lui adjoignant un certain nombre de contraintes basées sur la connaissance de l'étiquette d'un certain nombre d'individus". Le processus d'apprentissage ainsi réalisé mènera à la découverte d'une partition (et de la structure du graphe d'induction ayant mené à cette partition) telle qu'elle respecte les contraintes et qu'elle soit la plus "naturelle" possible.

Ainsi, par réalisation d'un apprentissage non supervisé sous contraintes nous obtenons une partition pour laquelle chaque classe est caractérisée par une règle logique, ces règles permettent donc d'associer à tout individu une classe de la partition. De plus, pour chaque classe de la partition on peut déterminer la modalité (de la variable endogène) la plus représentée parmi les individus dont on connaît l'étiquette et ainsi l'associer à la classe. On dispose alors de règles permettant non seulement d'associer à tout individu une classe de la partition découverte mais aussi une modalité de la variable endogène, cela correspond bien à un modèle d'apprentissage supervisé.

Notons que nous avons utilisé ici uniquement des contraintes visant à séparer des individus.

L'algorithme que nous proposons est présenté en page 51. Nous ne détaillons pas plus en détail la description formelle de cette méthode qui s'apparente aux graphes d'induction, toutefois nous proposons un exemple illustratif sur le jeu de données du tableau 3.5.

7. notons qu'il ne s'agit ici que d'un prototype de méthode d'apprentissage...

Algorithme 2 Algorithme d'Apprentissage Semi-Supervisé

Données :
l'ensemble des objets : $O = \{o_1,..,o_n\}$;
les variables exogènes $EV = \{V_i, i = 1..p\}$
la variable endogène V_E (qui possède k modalités ($v_{E1},..,v_{Ek}$))

1. Séparer l'ensemble des objets O en 2 échantillons : l'échantillon d'apprentissage O_{app} et l'échantillon de validation O_{val} ($O = O_{app} \bigcup O_{val}$).

2. Fixer le pourcentage d'individus ($pourc$) de l'échantillon d'apprentissage sur lesquels on impose des contraintes.

3. Séparer l'échantillon d'apprentissage O_{app} en 2 ensembles, l'un (O_{cont}) comprenant les individus sur lesquels des contraintes sont imposées ($card(O_{cont}) = pourc \times card(O_{app})$) et l'autre ($O_{rest}$) les objets restants ($O_{app} = O_{cont} \bigcup O_{rest}$).

4. Séparer l'échantillon O_{cont} en autant d'échantillons qu'il existe de modalités pour la variable endogène (c'est à dire séparer O_{cont} en k échantillons, chacun de ces échantillons est noté O_{cont_i} et est pur du point de vue de la variable endogène (il ne contient que des objets ayant la modalité v_{Ei} pour la variable endogène)) ($O_{cont} = \bigcup_{i=1..k} O_{cont_i}$).

5. Créer les contraintes telles qu'elles imposent aux individus de chacun des échantillons O_{cont_i} à ne pas être regroupés au sein d'une même classe qu'un individu appartenant à l'un des échantillons O_{cont_j}, $j = 1..k$, $j \neq i$. Cela mène à la création de k contraintes : $V_{cont_1},..,V_{cont_k}$ dont les pondération $p_1,..,p_k$ doivent être très forte, $\forall i = 1..k,\ p_i \rightarrow \infty$ (nous fixons $\forall i = 1..k\ p_i = p^3$). Chaque contrainte possède 3 modalités : ε_2, ε_4, ε_7, on assigne ces valeurs de modalités aux objets de O selon les règles suivantes : la valeur prise par l'objet o_a pour la variable V_{cont_i} est : ε_4 si $o_a \notin O_{cont}$; ε_2 si $o_a \in O_{cont_i}$; ε_7 si $o_a \in O_{cont_j}$, $j \in 1,..,k$, $j \neq i$

6. Lancer l'algorithme d'apprentissage non supervisé KEROUAC (avec $\alpha = 1$) sur le jeu de données considéré dans son intégralité (i.e. l'ensemble des objets considérés est bien O et non pas O_{app} en lui ayant au préalable intégré les contraintes (i.e. l'ensemble des variables considérées est constitué de V et des k contraintes modélisées sous la forme des variables $V_{cont_1},..,V_{cont_k}$; nous rappelons toutefois qu'aucun processus de segmentation ne peut être obtenue sur la base de ces dernières variables).

7. Obtention d'une partition $P_h = \{C_i, i = 1..z\}$ et de la structure du graphe d'induction lui ayant donné le jour.

8. Pour chaque classe de P_h on peut associer une modalité de la variable endogène, cette modalité étant celle la plus représentée parmi les objets de la classe appartenant à l'échantillon d'apprentissage, si toutefois aucun objet de l'échantillon d'apprentissage n'est présent dans la classe alors aucune modalité de la variable endogène n'est associé à la classe.

9. Obtention d'un ensemble de règles permettant éventuellement d'associer à un objet une modalité de la variable endogène. Si une règle ne permet pas d'associer une modalité de la variable endogène à un individu, cet individu est alors dit indéterminé.

EXEMPLE : *Considérons donc le jeu de données du tableau 3.5. Admettons que l'on veuille réaliser un apprentissage sur 50% des objets et que l'échantillon d'apprentissage (resp. de validation) soit composé de la manière suivante :*
$O_{app} = \{o_1,o_2,o_5,o_7,o_9,o_{10},o_{11},o_{14},o_{18},o_{20}\}$
(resp. $O_{val} = \{o_3,o_4,o_6,o_8,o_{12},o_{13},o_{15},o_{16},o_{17},o_{19}\}$*).*

Admettons maintenant que l'on fixe le paramètre $pourc$ *à 80%, et que* O_{cont} *(resp.* O_{rest}*) soit composé comme suit* $O_{cont} = \{o_1,o_5,o_7,o_9,o_{11},o_{14},o_{18},o_{20}\}$ *(resp.* $O_{rest} = \{o_2,o_{10}\}$*). On obtient donc* $O_{cont1} = \{o_1,o_5,o_7,o_9\}$, $O_{cont2} = \{o_{11},o_{14},o_{18},o_{20}\}$*. Cela permet de générer les variables* V_{cont_1} *et* V_{cont_2} *qui modélisent les contraintes impliquant que les objets de* O_{cont1} *et ceux de* O_{cont2} *ne doivent pas être réunis au sein d'une même classe. Notons que les pondérations (*p_1*,* p_2*) de ces deux variables doivent être très fortes (*$p_1 \to \infty$*,* $p_2 \to \infty$*). Si l'on lance l'algorithme* KEROUAC *(avec* $\alpha = 1$*) on obtient alors la partition suivante*
$\{\{o_1,o_2,o_4,o_7,o_8,o_{10},o_{12},o_{17}\}, \{o_5,o_9\}, \{o_3,o_6,o_{14},o_{15},o_{16}\}, \{o_{11},o_{13},o_{18},o_{19},o_{20}\}\}$*, le graphe d'induction de la figure 3.11, ainsi que les règles logiques suivantes* :

- $[V_2 = A] OU [V_2 = B\ ET\ V_4 = A] \Rightarrow Classe1$
- $[V_2 = C\ ET\ V_3 = A] \Rightarrow Classe2$
- $[V_2 = C\ ET\ V_3 = B] \Rightarrow Classe3$
- $[V_2 = B\ ET\ V_4 = B] \Rightarrow Classe4$

Par comptabilisation dans chaque classe des valeurs pour la variable endogène V_E *des objets de* O_{app} *appartenant à la classe, on peut associer la modalité la plus représentée à la classe et ainsi obtenir les règles logiques suivantes :*

- $[V_2 = A] OU [V_2 = B\ ET\ V_4 = A] \Rightarrow V_E = A$
- $[V_2 = C\ ET\ V_3 = A] \Rightarrow V_E = A$
- $[V_2 = C\ ET\ V_3 = B] \Rightarrow V_E = B$
- $[V_2 = B\ ET\ V_4 = B] \Rightarrow V_E = B$

On peut par la suite utiliser ces règles pour assigner une modalité de la variable endogène aux divers objets du jeu de données ce qui implique les associations suivantes : $o_1 \to V_E = A$, $o_2 \to V_E = A$, $o_3 \to V_E = B$, $o_4 \to V_E = A$, $o_5 \to V_E = A$, $o_6 \to V_E = B$, $o_7 \to V_E = A$, $o_8 \to V_E = A$, $o_9 \to V_E = A$, $o_{10} \to V_E = A$, $o_{11} \to V_E = B$, $o_{12} \to V_E = A$, $o_{13} \to V_E = B$, $o_{14} \to V_E = B$, $o_{15} \to V_E = B$, $o_{16} \to V_E = B$, $o_{17} \to V_E = A$, $o_{18} \to V_E = B$, $o_{19} \to V_E = B$, $o_{20} \to V_E = B$, .

Ici le taux de correction sur l'échantillon d'apprentissage (resp. de validation) est donc de 100% (resp. 60%).

Evaluation Expérimentale L'évaluation expérimentale a été réalisée sur 14 jeux de données (issus de la collection de l'université de Californie à Irvine [MM96], voir page 217 pour plus d'informations sur ces jeux de données) sur lesquels ont été menés divers apprentissages mettant en œuvre 6 méthodes d'apprentissage différentes : ID3, C4.5, Sipina et notre méthode pour les méthodes de type arbres/graphes d'induction, 1-plus proche voisins et bayésiens naïfs pour les autres méthodes d'apprentissage. Ces divers apprentissages ont

permis la réalisation d'une étude comparative concernant le taux de correction selon la méthode employée. L'évaluation du taux de correction est réalisée pour une 10-cross-validation ainsi que pour cinq 2-cross-validations. Pour ces évaluations, les paramètres suivants ont été adoptés : $pourc = 90\%$, $\alpha = 1$.

	ID3	C4.5	SIPINA	B.Naïfs	1-PPV	KEROUAC
GERMAN	$31.2^{3.37}$	$29^{4.1}$	$29.9^{4.5}$	$24^{3.6}$	$31.6^{6.07}$	$33.7^{3.74}$
MUSH.	$0.07^{0.13}$	0^{0}	$0.62^{0.17}$	$0.31^{0.19}$	0^{0}	0^{0}
SICK	$2.36^{1.11}$	$2.14^{0.81}$	$2.64^{0.99}$	$2.82^{1.45}$	$2.79^{0.91}$	0^{0}
VEHICLE	$33.7^{4.64}$	$32.4^{4.67}$	$43.51^{1.81}$	$33.32^{4.56}$	$33.21^{2.73}$	$35.11^{5.5}$
MONKS 3	0^{0}	0^{0}	0^{0}	$2.79^{2.71}$	$13.66^{4.68}$	0^{0}
FLAGS	$30.89^{7.23}$	$34.03^{10.6}$	$46.37^{11.4}$	$34.73^{11.8}$	$46.95^{10.2}$	$35.13^{9.55}$
BREAST	$9.3^{2.23}$	$5.43^{2.28}$	$6.87^{3.07}$	$3^{2.16}$	$5.58^{3.1}$	$4.44^{2.25}$
ZOO	$26.64^{10.62}$	$7^{6.4}$	$13.82^{7.88}$	$14^{11.14}$	$3.91^{6.56}$	$3.91^{4.58}$
WINE	$8.46^{3.83}$	$6.14^{6.31}$	$7.22^{7.88}$	$3.4^{4.56}$	$4.48^{4.17}$	$2.23^{2.72}$
CANCER	$7.47^{2.73}$	$5.27^{2.64}$	$4.68^{3.44}$	$2.49^{2.08}$	$5.42^{2.54}$	$5.42^{2.62}$
PIMA	$25.26^{3.22}$	$26.3^{5.07}$	$26.05^{5.41}$	$22.26^{4.13}$	$31.65^{4.64}$	$33.95^{2.68}$
CONTRA.	$52^{3.83}$	$50.71^{3.6}$	$56.29^{5.52}$	$50.16^{4.49}$	$56.82^{4.49}$	$43.74^{2.79}$
ION	$10.83^{6.98}$	$8.55^{3.13}$	$12.26^{5.14}$	$5.42^{2.99}$	$13.97^{5.35}$	$11.38^{4.89}$
HVOTES	$4.37^{3.62}$	$6.22^{3.88}$	$4.38^{2.18}$	$10.34^{4.15}$	$13.76^{4.37}$	$6.21^{3.34}$

Légende : Taux d'erreur Moyen $^{Ecart\ Type\ pour\ le\ taux\ d'erreur}$

TAB. 3.6 –: *Taux d'Erreur Moyen en Validation pour une 10-Cross-Validation*

Dans la mesure où la méthode présentée ici n'est en défintive qu'un prototype et que, nous le verrons, un certains nombre de questions restent à aborder nous ne détaillons pas ici les résultats de ces évaluations. Cependant, il apparaît clairement que les modèles d'apprentissage bâtis par le biais de cette méthode présentent une qualité très intéressante puisqu'elle est relativement similaire à celle des méthodes de références utilisées...

Ces résultats tendent à étayer la proposition de création de modèles d'apprentissage supervisé par des techniques d'apprentissage semi-supervisé et, plus précisément, dans le cas présent, de techniques d'apprentissage non supervisé sous contraintes.

Notons que plusieurs points mériteraient d'être approfondis tels que le type et le nombre de contraintes à intégrer. Ainsi, si les contraintes employées ici visent à séparer un certain nombre d'objets sélectionnés aléatoirement, on pourrait envisager une étude poussée concernant le nombre d'objets à assujetir à des contraintes ainsi qu'une étude concernant une sélection "intelligente" et non aléatoire de ces mêmes objets. De même, l'intégration d'autres types de contraintes doit être vecteur d'amélioration de qualité des modèles élaborés dans certaines situations.

Nous pensons donc que ces différents points soulignent plus encore l'intérêt de l'approche apprentissage non supervisé sous contraintes en démontrant

	ID3	C4.5	SIPINA	B.Naïfs	1-PPV	KEROUAC
GERMAN	$30.54^{0.47}$	$28.92^{1.09}$	$29.86^{0.71}$	$24.44^{0.74}$	$32.92^{1.49}$	$32.28^{1.69}$
MUSH.	$0.26^{0.05}$	$0.02^{0.03}$	$0.48^{0.16}$	$0.32^{0.02}$	0^{0}	0^{0}
SICK	$3.19^{0.17}$	$2.48^{0.18}$	$2.41^{0.04}$	$3.28^{0.24}$	$3.09^{0.43}$	$2.98^{0.34}$
VEHICLE	$37.64^{0.62}$	$34.18^{1.97}$	$46.48^{0.96}$	$34.54^{1.01}$	$36.41^{0.91}$	$35.62^{1.85}$
MONKS 3	$5^{0.06}$	0^{0}	$3.19^{1.13}$	2.78^{0}	$12.13^{1.45}$	0^{0}
FLAGS	$52.99^{1.71}$	$38.04^{4.81}$	$54.85^{1.06}$	$44.43^{2.77}$	$48.14^{3.29}$	$38.45^{4.66}$
BREAST	$8.56^{0.48}$	$5.41^{0.7}$	$6.32^{0.6}$	$3^{0.37}$	$5.58^{0.6}$	$5.81^{1.31}$
ZOO	$19^{1.47}$	$13.85^{2.44}$	$18.23^{0.8}$	$25.7^{4.39}$	$7.13^{1.6}$	$12.68^{7.19}$
WINE	$14.49^{1.15}$	$8.43^{2.59}$	$18.76^{2.09}$	$9.55^{1.88}$	$8.88^{1.96}$	$7.8^{3.3}$
CANCER	$8.37^{0.55}$	$6.76^{0.3}$	$6.53^{0.73}$	$2.9^{0.32}$	$5.04^{0.48}$	$5.27^{0.81}$
PIMA	$26.43^{1.44}$	$25.96^{0.92}$	$26.12^{1.09}$	$22.5^{0.1}$	$31.87^{1.23}$	$34.09^{2.85}$
CONTRA.	$53.22^{0.39}$	$51.54^{1.39}$	$58.38^{0.62}$	$49.75^{0.87}$	$59.02^{0.34}$	$55.12^{2.71}$
ION	$18.52^{1.39}$	$9.57^{1.07}$	$12.25^{1.62}$	$9.91^{1.78}$	$13.62^{0.75}$	$19.25^{3.84}$
HVOTES	$4.64^{0.55}$	$5.38^{0.63}$	$7.22^{3.54}$	$10.12^{1.21}$	$13.89^{1.12}$	$6.39^{1.24}$

Légende : Taux d'erreur Moyen $^{Ecart\ Type\ pour\ le\ taux\ d'erreur}$

TAB. 3.7 –: *Taux d'Erreur Moyen en Validation pour cinq 2-Cross-Validation*

qu'il est possible d'accroître la qualité des modèles ainsi que de créer des modèles performants dans des situations où l'on dispose de nombre d'objets non étiquetés.

3.3 Conclusion

L'évaluation de notre méthode de cns a permis de mettre en avant la qualité des cns produites, la très bonne stabilité et le coût calculatoire relativement faible de la méthode. De plus, les différentes comparaisons avec la méthode de référence que constitue les K-Modes montrent tout l'intérêt de KEROUAC [8].

Nous utilisons maintenant "la grille de lecture" utilisée plus tôt pour présenter les méthodes de cns classiques afin de résumer les caractéristiques principales de notre méthode :

- **Complexité :** identique à celle des graphes d'induction (log-linéaire selon le nombre d'objets et linéaire selon le nombre de variables), scalabilté apparemment bonne,
- **Géométrie des Classes :** pas d'a priori particulier pour la forme des classes [9],

8. Cependant, bien que les efforts en terme d'expérimentations aient été importants, mener un ensemble plus vaste encore d'expériences pourrait permettre d'obtenir un panorama plus vaste de résultats pour comparer ces deux méthodes...

9. Aucune évaluation précise menée sur ce point, mais dans la mesure où l'on utilise une mesure proche de celle employée dans RDA/AREVOMS on peut penser que le comportement de notre méthode est identique sur ce point à celui de RDA/AREVOMS (et ce bien que la méthode d'optimisation sous-jacente à notre méthode soit différente à celle de RDA/AREVOMS).

- **Gestion des Outliers :** oui[10]
- **Paramètres :** facteur de granularité
- **Résultats :** composition des classes, mode de chacune des classes et graphe permettant d'associer une règle caractérisant chacune des classes,
- **Critère :** $NCC^{\star}$.

Enfin, nous évaluons maintenant notre méthode au regard des challenges actuels en cns :

- **Problèmes inhérents aux données traitées :**
  - **Très grand nombre d'objets.** Complexité algorithmique théorique relativement faible et exhibe une bonne scalabilité.
  - **Dimensionnalité élevée.** La complexité de notre méthode est linéaire selon le nombre de variables ce qui de prîme abord semble très correct. Cependant KEROUAC est également sensible aux nombres de modalités des variables catégorielles traitées, ainsi si les variables possèdent de nombreuses modalités le coût calculatoire de notre méthode peut s'accroître sensiblement. Cela constitue selon nous le principal problème lié à notre méthode.
  - **Autres éléments problématiques.** La présence de *données manquantes* n'est pas gênante car leur traitement est aisément et rapidement paramétrable comme cela a été montré précédemment. La présence d'*outliers* n'apparaît pas vraiment problématique dans la mesure où notre méthode est relativement insensible à leur présence puisqu'elle peut générer des classes possédant peu d'objets voire un unique objet.
- **Problèmes inhérents à des contraintes applicatives :**
  - **Nécessité d'intégrer des connaissances, des contraintes dans le processus.** L'intégration de connaissances ou de contraintes est aisée comme cela est montré dans la section précédente.
  - **Bonne utilisabilité.** L'utilisabilité de notre méthode semble très bonne : le paramétrage de l'algorithme est facile, intelligible et intuitif, la présentation des résultats et des connaissances extraites par le processus de cns est explicite et intelligible.
  - **Données distribuées.** En soi, KEROUAC ne permet pas vraiment un traitement direct de données distribuées, cependant KEROUAC peut constituer une méthode d'agrégation de cns distribuées ce qui fait l'objet du chapitre 6.

Enfin, notons que cette méthode peut être utilisée comme base pour la mise au point de méthode d'apprentissage semi-supervisé (comme le montre la section précédente) ou encore de méthode visant à la complétion de données manquantes, d'apprentissage généralisé, ou de "profilage" (voir [JN03f])...