

Algorithme génétique de pondération d'attributs pour une classification non supervisée d'objets complexes

Alexandre Blansché et Pierre Gancarski

LSIIT, UMR 7005 CNRS-ULP
Parc d'Innovation, Boulevard Sébastien Brant
67412 ILLKIRCH
{blanche, gancarski}@lsiit.u-strasbg.fr
<http://lsiit.u-strasbg.fr/afd/>

Résumé. La classification non supervisée d'objets complexes, composés d'un nombre important d'attributs est souvent problématique. En effet il existe très fréquemment des corrélations entre les attributs ainsi que des attributs bruités ou non pertinents. Pour résoudre ce problème nous proposons une méthode de pondération d'attributs non supervisée par approche *wrapper*, cherchant des pondérations continues et locales.

Dans cette méthode, un ensemble d'extracteurs (de classe) est défini. Chaque extracteur est muni d'une stratégie qui lui permet d'extraire une unique classe. Chaque extracteur utilise une pondération différente sur les attributs. Le résultat global est obtenu par l'ensemble des classes ainsi extraites. Afin de chercher l'ensemble de poids optimal permettant d'obtenir la meilleure classification possible, nous utilisons un apprentissage par coévolution coopérative.

Dans cet article, nous précisons notre méthode, en particulier comment sont définis les extracteurs, comment est évaluée la qualité de la classification, comment se déroule l'apprentissage et comment est construit le résultat final. Nous présenterons nos premiers résultats, sur des données artificielles et sur des images de télédétection.

1 Introduction

La classification non supervisée consiste à créer des classes à partir d'un ensemble d'objets, telles qu'une classe regroupe des objets similaires et deux classes différentes des objets dissemblables. Contrairement à une approche supervisée, la classification non supervisée ne se base pas sur un ensemble d'exemples, mais opère uniquement à partir des propriétés intrinsèques des objets. Un panorama très complet des méthodes existantes est donné dans [Grabmeier et Rudolph, 2002].

Ces méthodes ont prouvé leur pertinence dans bien des domaines. Néanmoins, les données à traiter sont de plus en plus complexes :

- de part leurs types (intervalles, distributions, histogrammes, etc.) ;
- de part l'hétérogénéité de leurs attributs (données multi-sources) ;
- de part la qualité de ces attributs.

Ces méthodes classiques ne sont alors plus assez efficaces. De nombreuses recherches se focalisent sur des approches permettant l'utilisation d'attributs non élémentaires [Bock et Diday, 2000, Chavent et Lechevallier, 2002] ou permettant de pallier l'hétérogénéité des sources, par des méthodes de fusion [Bloch *et al.*, 2001], de combinaison [Kittler, 1998] ou par une approche collaborative [Wemmert *et al.*, 2000].

Concernant le problème de la qualité des attributs trois types d'approches ont été proposées :

- l'extraction d'attributs, qui consiste à transformer l'ensemble d'attributs de départ en un nouvel ensemble d'attributs, généralement plus petit, qui contient autant d'informations que les attributs d'origine. Les méthodes les plus connues consistent à projeter les données dans un nouvel espace de dimension inférieure par des méthodes de projections linéaires (analyse en composantes principales, analyse en composantes indépendantes, etc.) ou non linéaires (analyse en composantes curvilignes) [Jolliffe *et al.*, 1966, Lennon, 2002] ;
- la construction d'attributs, qui consiste à créer de nouveaux attributs basés sur les relations entre les attributs déjà existants ;
- la sélection et la pondération d'attributs, qui consistent à choisir ou pondérer au mieux les attributs à utiliser lors de la classification.

De nombreuses méthodes de pondération d'attributs ont été proposées avec différents aspects [John *et al.*, 1994, Wettschereck et Aha, 1995, Blum et Langley, 1997], les principaux étant à notre avis :

Le lien entre l'algorithme de pondération et celui de classification. Trois approches existent :

- l'approche *filter* ou *ignorant*, qui consiste à déterminer une pondération en fonction uniquement des propriétés intrinsèques des objets à classifier ;
- l'approche *wrapper* ou *feedback*, qui consiste à évaluer un ensemble de poids en utilisant le résultats d'une classification utilisant cet ensemble de poids ;
- l'approche *embedded*, qui consiste à modifier l'algorithme de classification pour y intégrer un mécanisme de pondération d'attributs.

La première approche ne calcule que la pondération alors que les deux dernières génèrent en plus une classification à partir de celle-ci.

L'espace des poids. Les poids utilisés peuvent être booléens (sélection d'attributs) ou continus (pondération d'attributs).

Le niveau de généralité. Les poids peuvent avoir une portée globale (un même ensemble de poids est utilisé pour l'ensemble des objets à classifier) ou locale (une pondération spécifique pour chaque classe à extraire est utilisée). En effet, même dans le cas où tous les attributs sont importants, leurs importances relatives peuvent dépendre de la classe à mettre en évidence. Par exemple, un géographe désirant mettre en évidence les classes d'eau dans une image de télédétection utilisera principalement des informations radiométriques. D'un autre côté, pour discriminer deux types de zones urbaines, il utilisera avant tout des informations de texture.

Nous proposons une méthode de pondération d'attributs non supervisée par approche *wrapper*, cherchant des pondérations continues et locales. En effet, il a été montré dans le cadre de la classification supervisée que :

- l’approche *wrapper* permet d’obtenir de meilleurs r sultats que l’approche *filter*, gr ce au retour d’informations de la classification [Wettschereck et Aha, 1995];
- la pond ration d’attributs (poids continus) permet de meilleurs r sultats que la s lection d’attributs (poids binaires) [Wettschereck *et al.*, 1997];
- l’utilisation de pond rations locales est pertinente [Howe et Cardie, 1997].

Nous pensons que cela est valable  galement pour une approche non supervis e. D’ailleurs de telles m thodes ont d j   t  propos es, en particulier deux m thodes *embedded* semblables dans leur principe, construites sur les algorithmes *K*-means ou Fuzzy *C*-means [Chan *et al.*, 2004, Frigui et Nasraoui, 2004] :   chaque it ration de l’algorithme utilis , de nouvelles pond rations sont calcul es en fonction de la compacit  sur chaque attribut pour chacune des classes.

Notre approche est diff rente. Elle est bas e sur l’id e que les informations sur les objets offertes par diff rentes m thodes d’extraction de classe(s) sont compl mentaires. Et donc que la combinaison de ces diff rentes m thodes permet d’augmenter leur efficacit  [Kittler, 1998]. Une classification peut  tre produite   partir de diff rentes m thodes utilisant diff rents points de vue : chaque m thode est utilis e pour obtenir une d cision consensuelle. L’approche la plus courante consiste   utiliser des classifieurs classiques et   fusionner directement les r sultats, par exemple par une m thode de vote [Bauer et Kohavi, 1999].

Dans notre m thode, un ensemble d’extracteurs (de classe) est d fini. Chaque extracteur est muni d’une strat gie qui lui permet d’extraire une unique classe. Chaque extracteur utilise une pond ration diff rente sur les attributs. Le r sultat global est obtenu par l’ensemble des classes ainsi extraites.

Afin de chercher l’ensemble de poids optimal permettant d’obtenir la meilleure classification possible, nous utilisons un apprentissage par co volution co op rative. Pour cela, *K* populations d’extracteurs sont d finies, pour *K* classes cherch es. Le r sultat global est obtenu par l’union d’une seule classe par population. Chaque population  volue afin de permettre l’extraction de classes qui maximisent la qualit  globale, mais les individus n’ voluent (croisements, mutations, etc.) qu’au sein de leur propre population.

Dans la section 2, nous pr ciserons notre m thode, en particulier comment sont d finis les extracteurs, comment est  valu e la qualit  de la classification, comment se d roule l’apprentissage et comment est construit le r sultat final. Dans la section 3 nous pr senterons nos premiers r sultats, sur des donn es artificielles et sur des images de t l d tection. Enfin, nous concluerons par les perspectives de d veloppement.

2 M thode propos e

Nous proposons une m thode de classification non supervis e pour des donn es complexes dans lesquelles les attributs peuvent  tre tr s nombreux et de types vari s : simples (num riques, symboliques) ou structur s (histogrammes, etc.). De plus ces attributs peuvent  tre :

- bruit s : capteur d faillant, saisie erron e, etc. ;
- corr l s entre eux : les attributs corr l s entre eux auront un poids plus important que des attributs ind pendant (par exemple, si des donn es sont repr sent es par

trois attributs $f1$, $f2$ et $f3$ et que les valeurs sur $f2$ et $f3$ sont identiques pour tous les objets, cela reviendrait à avoir deux attributs $f1$ et $f2$, avec des pondérations respectives $\frac{1}{3}$ et $\frac{2}{3}$, ce qui risque de fausser la classification) ;

- non pertinents : en essayant d'amasser un maximum d'informations sur un objet, certaines de ces informations n'ont aucune pertinence pour une éventuelle classification.

La méthode que nous proposons pour pouvoir traiter ce type de données consiste à pondérer les attributs de façon à optimiser la qualité de la classification. Nous utilisons pour cela un algorithme de coévolution coopérative dans lequel les individus sont des extracteurs. Le chromosome utilisé pour représenter un extracteur est l'ensemble des poids utilisés par celui-ci pour extraire une classe. K populations d'extracteurs sont définies, où K est le nombre de classes à extraire. La classification finale est obtenue par l'union de classes extraites par les individus représentatifs de chaque population.

Le principe est alors le suivant. À chaque génération (étape de l'algorithme génétique) :

1. Chaque individu (qui est un extracteur) propose une classe, en fonction de la pondération qu'il utilise. Chaque individu est alors évalué en calculant la qualité de la classification obtenue à partir de la classe extraite et des classes des individus représentatifs des autres populations.
2. Au sein de chaque population, une étape d'évolution génétique (sélection, croisements, mutations) est réalisée en utilisant l'évaluation des individus.
3. Les individus représentatifs sont recalculés et le résultat (la classification, composée des classes extraites) de la génération courante est déterminé.

Ce processus est réitéré tant que la qualité du résultat de la génération courante s'améliore ou suivant un nombre d'étapes déterminé.

En conséquence, pour définir notre méthode de classification modulaire, et en particulier le processus de pondération d'attributs, il est nécessaire d'étudier les quatre points suivants :

- Comment extraire une classe (cf. 2.1) ?
- Quel est le critère de qualité à utiliser pour l'algorithme génétique (cf. 2.2) ?
- Comment faire évoluer les extracteurs pour que les pondérations soient les meilleures possibles (cf. 2.3) ?
- Comment unifier les résultats (cf. 2.4) ?

2.1 Extraction d'une classe

Formellement, toute fonction qui renvoie un sous-ensemble (une classe) $X(S)$ de l'ensemble des objets à classer $S = \{o_1, \dots, o_N\}$ est un extracteur. Une telle fonction peut être définie par différents procédés (par exemple, un seuillage).

Nous avons choisi de définir les extracteurs à partir d'algorithmes de classification classiques. Un extracteur est alors un triplet $X = (M, w, r)$, où M est une méthode de classification, w un ensemble de poids $\{w_1, \dots, w_n\}$, $0 < w_i < 1$ (n étant le nombre d'attributs des éléments de S), et r un critère de qualité d'une classe. Ainsi, pour calculer la classe extraite $X(S)$, on effectue tout d'abord une classification en utilisant

la méthode M et les poids w sur les attributs des éléments de S . On obtient ainsi un ensemble de K_M^w classes $\{C_1, \dots, C_{K_M^w}\}$. La classe $X(S) = C_e$ telle que $r(C_e) = \max\{r(C_k), k = 1, \dots, K_M^w\}$ est alors sélectionnée.

Par exemple, si on définit un extracteur à partir de la méthode de classification non supervisée K -means [MacQueen, 1965], les poids seront utilisés pour calculer la distance entre les objets (définition 2.1) et le critère de qualité pour déterminer la classe extraite pourra alors être une mesure de compacité de classe.

Définition 2.1 (Distance pondérée) *La distance pondérée entre deux objets o et o' composés de n attributs en utilisant les poids w peut être définie par :*

$$d_w(o, o') = w_1 \times d_1(o, o') + w_2 \times d_2(o, o') + \dots + w_n \times d_n(o, o')$$

où les d_i sont les distances sur chacun des n attributs.

De même, pour la méthode de formation de concepts Cobweb [Fisher, 1987], les poids serviront au calcul de la prédictivité, qui sera utilisée comme critère de qualité pour sélectionner la classe extraite.

Dans la suite, par facilité de langage, nous utiliserons le terme de répartition (définition 2.2) pour désigner un ensemble de classes extraites.

Définition 2.2 (Répartition) *Une répartition est un ensemble $D = \{D_1, \dots, D_K\}$ de K sous-ensembles de l'ensemble des N objets observés $S = \{o_1, \dots, o_N\}$. Les intersections entre les D_i ne sont pas forcément vides et $\bigcup_{1 \leq i \leq K} D_i$ n'est pas forcément égal à E .*

2.2 Qualité d'une classification

Dans une approche *wrapper*, afin de trouver le meilleur ensemble de poids à utiliser, il est nécessaire d'évaluer la qualité d'une classification obtenue en utilisant ces poids.

Dans le cas des méthodes supervisées, il est relativement aisé de définir la qualité d'une classification : il suffit par exemple de compter les objets placés dans la bonne classe dans l'ensemble d'apprentissage. Mais dans le cas non supervisé, il n'est possible d'utiliser que des critères statistiques.

De nombreux indices de qualité d'une classification non supervisée, comme la compacité ou la séparabilité entre les classes, ont ainsi été définis [Nicolayannis *et al.*, 1997, Halkidi *et al.*, 2001, Günter et Burke, 2001, Bolshakova et Azuaje, 2003].

Il existe aussi des méthodes plus complexes. Par exemple, certaines méthodes de validation [Bel Mufti et Bertrand, 1997, Levine et Domany, 2001, Tibshirani *et al.*, 2001] consistent à tester la stabilité de l'algorithme de classification en rééchantillonnant les données. De telles méthodes sont utilisées pour chercher les paramètres optimaux des algorithmes, en particulier le nombre de classes.

Mais les méthodes classiques de validation s'appliquent difficilement dans notre cas. En effet, il est impossible d'utiliser directement la distance entre les classes, chacune utilisant une métrique différente (les pondérations étant différentes). Pour résoudre

ce problème, dans [Dy et Brodley, 2000], les auteurs proposent une méthode de normalisation des indices pour des classifications utilisant des métriques différentes, mais celle-ci ne s'applique que pour la comparaison de deux classifications et ne donne pas une qualité dans l'absolu.

De plus dans notre cas, nous travaillons sur des répartition et non des partitions. Or la majorité des indices ne s'appliquent qu'à des partitions. En utilisant uniquement un critère de qualité interne (comme la compacité), il est alors possible d'obtenir une qualité correcte même s'il y a beaucoup d'objets non classifiés et des intersections entre les classes.

De fait, le critère de qualité d'une classification, critère qui sera utilisé dans l'algorithme génétique (fonction de fitness), doit dépendre de la qualité de la répartition que forment les classes obtenues : il doit donc en particulier tenir compte du degré du partitionnement obtenu (cf. 2.2.1) et de la qualité de chacune des classes qui le composent (cf. 2.2.2).

2.2.1 Degré de partitionnement

Définition 2.3 (Objets k -classifiés) *Un objet k -classifié dans une répartition D est défini comme un objet appartenant à k classes de D , donc qui a été extrait par k extracteurs. Nous définissons $S_k(D)$ comme l'ensemble de tous les objets k -classifiés d'une partition D .*

De façon évidente, D est une partition si et seulement si $S_1(D) = S$, S étant l'ensemble des objets à classifier. On donnera alors une qualité de partitionnement maximal à D . Réciproquement, moins il y a d'objets 1-classifiés dans D moins bonne sera la qualité de partition de D .

Ceci peut être traduit en définissant un degré de partitionnement de sorte que la valeur est maximale si la répartition est une partition et nulle si aucun objet n'est 1-classifié.

On peut définir le degré de partitionnement d'une répartition D par $Q_P(D) = \frac{|S_1(D)|}{|S|}$. Cette définition du degré de partitionnement donne le taux d'objets 1-classifiés par rapport au nombre d'objets à classifier. Mais ce critère n'est pas satisfaisant car aucune différence n'est faite entre des objets k -classifiés et k' -classifiés ($2 \leq k < k'$), alors qu'il serait naturel de trouver « moins grave » qu'il y ait un objet 2-classifié qu'un objet 4-classifié, par exemple.

Pour définir le degré de partitionnement, nous avons donc d'abord défini une qualité pour chaque objet (définition 2.4).

Définition 2.4 (Qualité relative à un objet) *On définit la qualité relative à un objet o dans une répartition D par :*

$$Q_o(o, D) = 1 - \left| \left| \{D_i \mid o \in D_i, D_i \in D\} \right| - 1 \right| = \begin{cases} 0 & \text{si } o \in S_0(D) \\ 1 & \text{si } o \in S_1(D) \\ 2 - k & \text{si } o \in S_k(D), \text{ avec } k \geq 2 \end{cases}$$

Ainsi, si un objet o est 1-classifi  dans D alors $Q_o(o, D) = 1$, sinon $Q_o(o, D) \leq 0$.

Nous pouvons alors d finir la qualit  sur l'ensemble des objets (d finition 2.5). C'est ce crit re que nous utiliserons pour  valuer nos classifications.

D finition 2.5 (Degr  de partitionnement) *Le degr  de partitionnement d'une r partition D est d fini par :*

$$\begin{aligned} Q_P(D) &= \max \left(0, \frac{1}{N} \sum_{o \in S} Q_o(o, D) \right) \\ &= \max \left(0, \frac{1}{N} \left(|S_1(D)| - \sum_{k=2}^K ((k-2) \times |S_k(D)|) \right) \right) \end{aligned}$$

On voit facilement que $Q_P(D) = 1$ si et seulement si D est une partition de S . En effet, si D est une partition, $Q_o(o, D) = 1$ pour tous les objets o , donc $\sum_{o \in S} Q_o(o, D) = N$ et $Q_P(D) = 1$. Si D n'est pas une partition, il existe un objet o tel que $Q_o(o, D) < 1$, donc $\sum_{o \in S} Q_o(o, D) < N$ et $Q_P(D) < 1$. De plus, si aucun objet n'est 1-classifi , la somme des qualit s relatives   chacun des objets sera n gative ou nulle, et donc Q_P sera nul.

Plus pr cis ment nous allons montrer la validit  de ce crit re en  tudiant 3 cas. Nous allons comparer deux distributions D et D' d'un ensemble S de N  l ments. On consid re que le taux d'objet k -classifi s dans D (respectivement D') est de p_k (respectivement p'_k), avec $\sum_{i=0}^K p_i = 1$. On a alors $\overline{Q_o(D)} = \frac{1}{N} \sum_{o \in S} Q_o(o, D) = p_1 -$

$$\sum_{k=2}^K (k-2)p_i.$$

Dans chacun des 3 cas pr sent s ci-dessous, les distributions D et D' ne diff rent que pour un objet o . Dans chacun des cas, on consid re intuitivement que la classification D' est « meilleure » que D car D' est plus proche d' tre une partition de l'ensemble de donn es.

cas 1 D et D' sont indentiques,   la diff rence que o est 0-classifi  dans D , mais 1-classifi  dans D' . Donc :

- $p_0 = p'_0 + \frac{1}{N}$
- $p_1 + \frac{1}{N} = p'_1$
- $p_k = p'_k, k \in [2; K]$

Alors, $\overline{Q_o(D')} - \overline{Q_o(D)} = p'_1 - p_1 = \frac{1}{N}$, et donc $\overline{Q_o(D')} > \overline{Q_o(D)}$.

cas 2 D et D' sont indentiques,   la diff rence que o est i -classifi  (avec $i \geq 2$) dans D , mais 1-classifi  dans D' . Donc :

- $p_0 = p'_0$
- $p_1 + \frac{1}{N} = p'_1$
- $p_i = p'_i + \frac{1}{N}$
- $p_k = p'_k, k \in [2; K] \setminus \{i\}$

Alors, $\overline{Q_o(D')} - \overline{Q_o(D)} = p'_1 - p_1 + (i-2)(p_i - p'_i) = \frac{i-1}{N}$, et donc $\overline{Q_o(D')} > \overline{Q_o(D)}$.

cas 3 D et D' sont indentiques, à la différence que o est i -classifié dans D , mais j -classifié dans D' , avec $2 \leq j < i$. Donc :

- $p_0 = p'_0$
- $p_1 = p'_1$
- $p_i = p'_i + \frac{1}{N}$
- $p_j + \frac{1}{N} = p'_j$
- $p_k = p'_k, k \in [2; K] \setminus \{i, j\}$

Alors, $\overline{Q_o(D')} - \overline{Q_o(D)} = (i-2)(p_i - p'_i) + (j-2)(p_j - p'_j) = \frac{i-j}{N}$, et donc $\overline{Q_o(D')} > \overline{Q_o(D)}$.

Nous voyons dans l'étude de ces trois cas que la qualité de D' est meilleure que celle de D , ce qui correspond à l'attente que nous avons de ce critère.

Cependant si l'on n'utilise que ce critère de qualité, il est possible qu'une classe englobe la quasi-totalité des objets et que le résultat forme une partition parfaite, mais sémantiquement incohérente. Un second critère est donc utilisé afin de tenir compte de la qualité interne des classes extraites.

2.2.2 Qualité des classes

Définition 2.6 (Qualité des classes) *La qualité des classes d'une répartition est définie par :*

$$Q_C(D) = \prod_{D_i \in D} q(D_i)$$

où $q(D_i)$ est un critère de qualité d'une classe (par exemple la compacité) qui prend ses valeurs dans $[0; 1]$.

2.2.3 Qualité totale d'une répartition

Il nous est maintenant possible de définir un critère de qualité d'une répartition (définition 2.7) à partir des définitions précédentes.

Définition 2.7 (Qualité totale d'une répartition) *La qualité totale d'une répartition D est définie par :*

$$Q(D) = Q_P(D) \times Q_C(D)$$

2.3 Évolution des extracteurs

Pour découvrir un ensemble de poids optimal pour chaque extracteur nous utilisons un algorithme génétique.

Les algorithmes génétiques [Holland, 1975, Goldberg, 1989, Renders, 1995] sont des algorithmes d'optimisation stochastiques inspirés de la théorie de l'évolution de Darwin. Chaque individu est une solution candidate au problème à résoudre et est représenté par un gène qui est l'ensemble des paramètres à optimiser. Un ensemble d'individus est

cr    al  atoirement, puis    chaque g  n  ration, les individus les plus performants, selon le crit  re de qualit   qui d  pend du probl  me    r  soudre, sont s  lectionn  s pour cr  er la g  n  ration suivante par des op  rations de croisement (recombinaison des g  nes de plusieurs individus) ou de mutation (modification al  atoire d'un g  ne). La nouvelle g  n  ration ainsi cr   e a de grandes chances de comprendre des individus plus performants.

De tels algorithmes ont d  j     utilis  s dans le cadre de la classification non supervis  e mais toutes ces approches cherchent    classer les donn  es selon tous les attributs et ne cherchent pas de pond  rations sur les attributs. Dans [Murthy et Chowdhury, 1996], chaque individu repr  sente une proposition de classification. Un chromosome comporte autant de g  nes que d'objets    classer, chaque g  ne repr  sente la classe de l'objet correspondant. Le but est alors de minimiser la somme des carr  s des distances par rapport aux centres des classes. Dans [Tseng et Yang, 2001, Garai et Chaudhuri, 2004], deux m  thodes assez semblables sont pr  sent  es. La classification s'op  re en deux phases. Dans la premi  re phase, les donn  es sont regroup  es dans des classes tr  s compactes (et nombreuses). Dans la seconde phase, les classes sont fusionn  es par l'algorithme g  n  tique. Un g  ne de cet algorithme est un ensemble de classes, dites classes s  lectionn  es. Les classes s  lectionn  es seront les noyaux des classes finales. Chacune des autres classes (c'est-  -dire les classes qui n'ont pas   t   s  lectionn  es dans le g  ne repr  senteant l'individu) est fusionn  e avec la classe s  lectionn  e qui lui est la plus proche.

2.3.1 Chromosomes

Dans notre approche modulaire de classification, nous avons   tudi   deux possibilit  s pour utiliser un apprentissage   volutif.

Une premi  re, classique, consistait    utiliser un individu pour repr  senter un ensemble d'extracteurs. Un chromosome repr  senteant un individu est alors constitu   de l'ensemble des poids $w_i^j \in [0, 1]$ sur chacun des attributs pour chacun des extracteurs. L'  valuation d'un individu se fait en calculant la qualit   totale de r  partition obtenue en utilisant les extracteurs d  finis par le chromosome (cf. d  finition 2.7). Sur la figure 1 est repr  sent   un chromosome dans le cas d'une classification avec K classes cherch  es et pour des donn  es sur n attributs. Le chromosome repr  sente les K extracteurs.

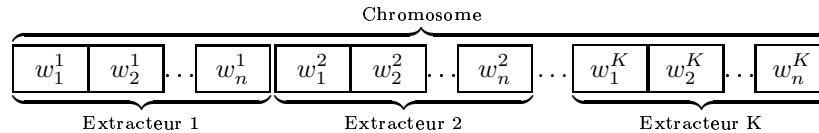


FIG. 1 – Un chromosome dans l'approche classique

Mais nous avons rapidement abandonn   cette m  thode car le nombre de g  nes sur chaque chromosome est alors tr  s important et il y a donc un plus grand risque de rester bloquer dans un maximum local. Nous pensons qu'il est plus judicieux de diviser le probl  me et de faire   voluer des objets plus simples.

Nous avons donc pr  f  r   une autre approche, que nous   tudierons ici plus en d  tail. Chaque individu repr  sente un extracteur. Un chromosome est constitu   des poids

$w_i \in [0, 1]$ sur chaque attribut pour l'extracteur qu'il représente. Le but évident est de faire évoluer plusieurs individus en collaboration afin que les classes obtenues par un ensemble d'individus forment une bonne classification de l'ensemble des données. Sur la figure 2 est représenté un chromosome dans le cas d'une classification sur des données sur n attributs. Le chromosome ne représente qu'un seul extracteur.

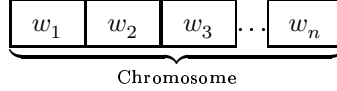


FIG. 2 – Un chromosome dans l'approche par coévolution

Un algorithme de coévolution coopérative est utilisé sur ces individus. La coévolution coopérative a été définie dans [Potter et De Jong, 1994, Potter *et al.*, 1995]. Il s'agit d'algorithmes évolutionnaires dans lesquels plusieurs populations sont utilisées. Ces différentes espèces cohabitent dans un écosystème dans lequel chaque espèce occupe une niche spécifique. Ainsi, chaque population évolue dans un environnement qui dépend des autres populations et qui évolue donc en même temps qu'elles.

Dans notre cas, une population par classe à mettre en évidence est utilisée, les répartitions étant construites en utilisant une classe extraite de chaque population (c'est-à-dire, une classe extraite par un individu de chacune des populations). Comme la méthode est non supervisée, la niche de chaque population (c'est-à-dire la classe représentative) n'est pas connue a priori, mais sera découverte au cours de l'apprentissage en fonction des données qu'il reste à « occuper ».

2.3.2 Évaluation d'un individu

La fonction d'évaluation de la définition 2.7 permet de définir la qualité d'une répartition, c'est-à-dire d'un ensemble de classes extraites. Il est maintenant nécessaire d'évaluer la qualité d'un individu, donc la qualité d'un extracteur. Comme cela est décrit dans [Mayer, 1998], cette évaluation peut se faire face à un seul individu (représentatif) de chacune des autres populations (environnement simple), face à un ensemble d'individus de chacune des populations (environnements multiples) ou face à tous les individus (environnement complet). L'évaluation dans un environnement complet pose un problème important, car il faut étudier toutes les combinaisons possibles comportant un extracteur de chaque population. Si l'on cherche K classes avec p individus dans chaque population, alors $K \times p$ classifications et p^K calculs de qualité sont effectués à chaque génération. Le temps de calcul devient rédhibitoire pour un nombre important de données. Afin d'accélérer l'évaluation de la qualité des extracteurs, nous n'évaluons les individus que dans un environnement simple, que nous appelons, dans le cadre de notre méthode de classification, répartition de référence (définition 2.8).

Définition 2.8 (Répartition de référence) *Pour une génération g donnée ($g \neq 1$), on définit la répartition de référence comme la meilleure répartition trouvée au cours des générations précédentes, elle est notée $\Delta(g) = \{\Delta_1(g), \dots, \Delta_i(g), \dots, \Delta_K(g)\}$ où $\Delta_i(g)$ correspond à classe retenue pour la i -ème population.*

La r partition de r f rence est recalcul e   chaque g n ration.   la g n ration $g + 1$, $\Delta(g + 1)$ est la meilleure r partition qu'il est possible de cr er en utilisant $\Delta_i(g)$ et le meilleur individu $B_i(g)$ de chaque population   la g n ration g , c'est- -dire :

$$Q(\Delta(g + 1)) = \max(Q\{P_1(g), \dots, P_K(g)\} \mid P_i(g) = \Delta_i(g) \text{ ou } P_i(g) = B_i(g))$$

On peut alors donner une d finition de la qualit  d'un extracteur (d finition 2.9).

D finition 2.9 (Qualit  d'un extracteur) *La qualit  d'un extracteur X_i^j (le j -i me individu de la i -i me population),   une g n ration g , est d finie par la qualit  de la r partition obtenue en rempla ant la i - me classe de $\Delta(g)$ par $X_i^j(S)$. Ainsi, $Q(X_i^j) = Q(D_i^j)$, o  $D_i^j = \{\Delta_1(g), \dots, X_i^j(S), \dots, \Delta_K(g)\}$*

Ainsi, seuls $K \times p + 2^K$ calculs de qualit  sont effectu s   chaque g n ration. Le gain obtenu est important.

L' valuation des individus   la premi re g n ration se fait en choisissant al atoirement un individu dans chacune des autres populations.

Il reste   savoir si l'utilisation d'une r partition de r f rence permet un apprentissage efficace. En effet, beaucoup moins de r partitions seront  valu es que dans un environnement complet.

Nous avons donc compar  les deux m thodes. Pour cela nous avons utilis  un jeu de donn es tr s petit, car la recherche dans l'ensemble de toutes les solutions possibles est trop longue pour les donn es sur lesquelles nous travaillons normalement. Les classifications obtenus ne sont pas significatives, cependant il est possible de comparer l' volution de l'apprentissage, m me s'il n'y a pas de convergence vers des r sultats s mantiquement corrects. La figure 3 montre l' volution des deux m thodes sur 100 g n rations. L'apprentissage se fait avec trois populations de 50 individus chacune. Ainsi, dans un environnement complet, 125 000 r partitions sont  valu es   chaque g n ration contre 128 lorsqu'on utilise une r partition de r f rence.

Il est   noter que les valeurs obtenues varient d'une ex cution   l'autre, mais restent relativement similaires.

  la premi re g n ration, un environnement complet permet d'obtenir la meilleure r partition possible avec les classes propos es par les individus. Sa qualit  de 0,67 contre 0,54 dans le cas d'une r partition de r f rence al atoire.

Mais par la suite, on voit tr s clairement, que l'utilisation d'une r partition de r f rence ne nuit absolument pas aux r sultats. Au contraire, l'apprentissage en est m me am lior .

Dans un environnement complet, la qualit  atteint la valeur 0,76 (  la 57  g n ration). On remarque que la qualit  oscille beaucoup au cours des g n rations et que les am liorations semblent plus dues au hasard qu'  un r el apprentissage. Ceci est d    l'effet « Red queen » : si une population s'est adapt e   son environnement   une g n ration g , celui-ci peut changer   la g n ration $g + 1$ et il sera peut- tre n cessaire de recommencer l'apprentissage [Cliff et Miller, 1995, Floreano et Nolfi, 1997, Paredis, 1997].

En utilisant une r partition de r f rence, la qualit  atteint la valeur 0,8   la 78  g n ration. Ici, il n'y a pas d'oscillation car la r partition de r f rence att nue l'effet

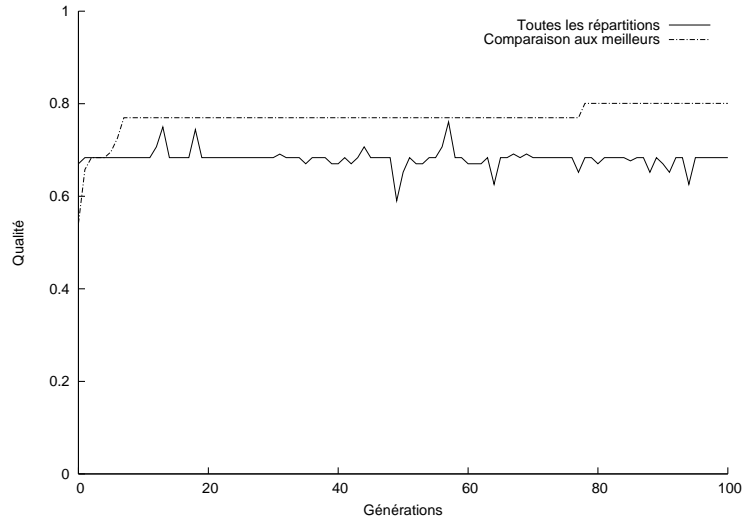


FIG. 3 – Comparaison des deux méthodes d'évaluation

« Red queen », puisque les changements d'une génération à l'autre sont moins importants. On conserve en effet toujours les meilleures classes extraites et l'évaluation se fait en fonction de celles-ci. Ainsi, tant qu'il n'y a pas d'amélioration des individus, l'environnement auquel ils doivent s'adapter ne change pas.

2.3.3 Évolution génétique

Un algorithme de coévolution peut être schématisé par la figure 4. On voit que chaque génération peut être divisée en quatre phases successives :

1. Évaluation des individus selon le critère de qualité.
2. Sélection du meilleur individu de chaque population.
3. Modification du critère de qualité en fonction des meilleurs individus de chaque population.
4. Sélection des individus, en fonction de leur qualité, et reproduction afin de créer de nouveaux individus pour la génération suivante, et ce au sein de chacune des populations.

Dans notre cas, ces étapes sont : (1) l'évaluation de la qualité des extracteurs de chaque population (définition 2.9), (2) la sélection du meilleur extracteur, (3) la mise à jour de la répartition de référence (définition 2.8), et (4) la création de nouveaux extracteurs.

Les opérations génétiques utilisées sont les opérations classiques. Chaque individu d'une nouvelle génération est créé soit à partir du croisement de deux individus (pour chaque gène de l'individu « enfant », on affecte aléatoirement la valeur de l'un des deux « parents »), soit par mutation d'un individu (des gènes sont modifiés aléatoirement),

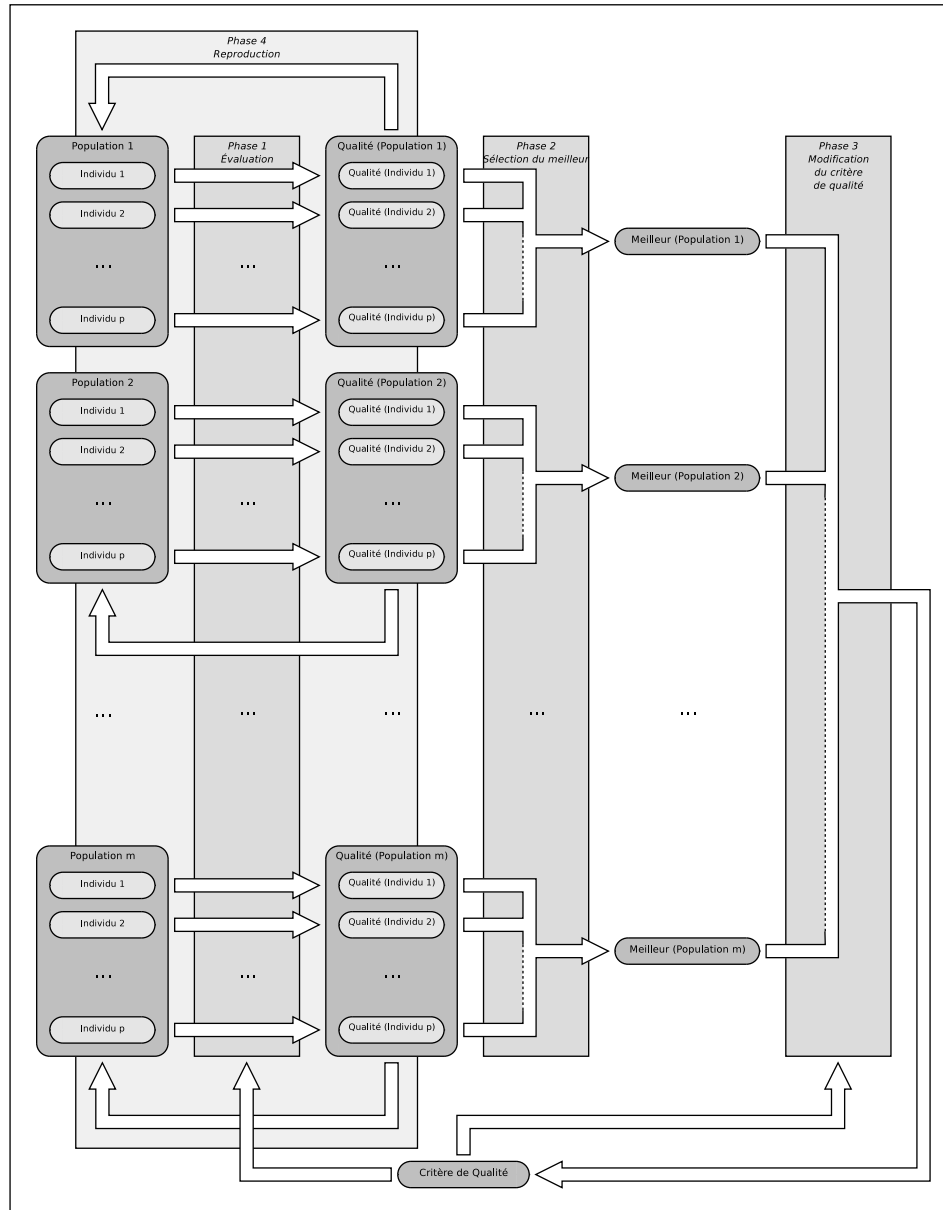


FIG. 4 – Algorithme g n tique utilis 

soit par copie d'un individu. La s lection des individus utilis s pour la construction de la g n ration suivante se fait al atoirement, la probabilit  de choisir un individu d pendant de sa qualit  (roulette-wheel selection). Le taux de chacune des op rations

génétiques est déterminé a priori par l'utilisateur.

2.4 Unification des résultats

La classification globale est une répartition constituée de K classes obtenues indépendamment, ne formant pas nécessairement une partition de l'ensemble de données à classer. Il peut y avoir des objets qui appartiennent à plusieurs classes (objets k -classifiés, avec $k > 1$) et d'autres qui n'appartiennent à aucune d'entre elles (objets 0-classifiés).

Nous avons défini deux méthodes, l'unification brute et l'unification relative afin de pallier ce problème et obtenir ainsi une partition en résultat :

- l'unification brute consiste simplement à placer les objets qui ne sont pas 1-classifiés dans une nouvelle classe d'objets considérés comme non classifiés ;
- l'unification relative consiste à affecter un objet à la classe qui lui correspond le mieux (parmi soit toutes les classes pour les objets 0-classifiés, soit uniquement les classes qui contiennent l'objet pour les objets k -classifiés, avec $k > 0$).

Dans le cas général, le calcul de l'unification relative se fait en utilisant un degré d'appartenance à une classe, définie en fonction de la méthode utilisée. Chaque objet est alors affecté à la classe à laquelle il « appartient le plus » (c'est-à-dire, à la classe pour laquelle le degré d'appartenance est le plus élevé).

Ainsi, dans le cas des méthodes basées sur une distance, on affecterait chaque objet à la classe dont le centre est le plus proche. Or, dans notre approche, il n'est pas possible de comparer directement les distances car chaque classe est obtenue à partir d'une métrique différente. Nous définissons donc une distance relative d'un objet au centre de classe la classe extraite (cf. définition 2.10), qui sera utilisée pour choisir la classe à laquelle appartient l'objet.

Définition 2.10 (Distance relative) Soient $X = (M, w, r)$ un extracteur et $C_1, \dots, C_{K_M^w}$ les classes obtenues en appliquant la méthode M avec les poids w sur l'ensemble S de données à classer. On définit la distance relative d'un objet o au centre de la classe extraite $X(S)$ par :

$$d_r(o, X(S)) = \frac{d_w(o, g_{X(S)})}{d_w(o, g)}$$

où $g_{X(S)}$ est le centre de la classe extraite par $X(S)$ et g le centre de la classe C_i , différente de $X(S)$, le plus proche de o .

Par exemple, soit un extracteur $X = (M, w, r)$ et $\{C_1, C_2, C_3\}$ les trois classes obtenues en appliquant M avec les poids w sur un ensemble de données à deux dimensions, C_1 étant la classe extraite par X selon le critère r . Sur la figure 5, sont représentées les trois classes, avec leurs centres de gravité, g_1 , g_2 et g_3 et deux objets A et B . On voit clairement que $d_r(A, C_1) = \frac{d_w(A, g_1)}{d_w(A, g_2)}$ est inférieur à 1 (A appartient à la classe C_1), alors que $d_r(B, C_1) = \frac{d_w(B, g_1)}{d_w(B, g_3)}$ est supérieur à 1 (B appartient à la classe C_3).

L'unification relative se fait alors en affectant o à la classe extraite la plus proche selon la distance relative. On voit facilement qu'un objet est affecté à l'une des classes

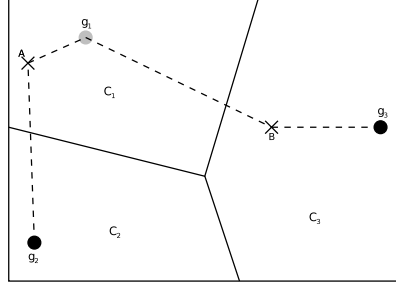


FIG. 5 – Calcul de la distance relative

extraites   laquelle il appartient, et donc, si o est 1-classifi , il sera affect    la classe de l'unique extracteur qui l'a extrait.

3 R sultats

Dans les exp rimentations pr sent es ici, la m thode de classification M utilis e pour d finir les extracteurs est une m thode bas e sur K -means. Le crit re de qualit  d'une classe r est une mesure de compacit  d'une classe (d finition 3.1) d finie dans [Wemmert *et al.*, 1999].

D finition 3.1 (Crit re de compacit ) *On d finit la qualit  d'une classe C_k par :*

$$r(C_k) = \begin{cases} 0, & \text{si } \frac{1}{n_k} \sum_{l=1}^{n_k} \frac{d(x_{k,l}, g_k)}{d(x_{k,l}, g)} > 1 \\ 1 - \frac{1}{n_k} \sum_{l=1}^{n_k} \frac{d(x_{k,l}, g_k)}{d(x_{k,l}, g)}, & \text{sinon} \end{cases}$$

avec n_k le nombre d'objets appartenant   la classe C_k , $x_{k,l}$ le l -i me  l ment de C_k , g_k le centre de gravit  de C_k et g le centre de gravit  diff rent de g_k le plus proche de $x_{k,l}$.

3.1 Donn es artificielles

Nous avons test  notre m thode sur des donn es artificielles. Ces donn es (10 000 objets) comportent 5 classes de 2 000 objets chacune. Chaque objet est repr sent  par 20 attributs r els (les valeurs allant de 0   255). Ces attributs sont plus ou moins bruit s selon une loi gaussienne, certains sont m me totalement al atoires. Ces attributs pr sentent de nombreuses redondances (certains attributs ont des valeurs similaires   d'autres et ont donc une plus grande importance dans la classification). D'autres ont des valeurs identiques d'une classe   l'autre et ne permettent pas de les discriminer. Chacun des attributs peut  tre repr sent  par une image en niveau de gris, comme sur la figure 6, les classes  tant regroup es en 5 bandes horizontales. Sur les 4 attributs repr sent s, on peut remarquer les probl mes suivants :

- les deux premiers attributs (FIG. 6(a) et FIG. 6(b)) sont peu bruités, mais ne permettent pas seuls de discriminer toutes les classes (par exemple, les 1^{re}, 3^e et 5^e classes sont identiques et ainsi que la 2^e et la 4^e) ;
- le 3^e attribut (FIG. 6(c)) n'est pas pertinent pour la 1^{re} et 3^e classe, car les valeurs sont uniformément distribuées. Il ne permet pas non plus de distinguer les autres classes entre elles, car les valeurs sont trop similaires ;
- le 3^e attribut (FIG. 6(c)) est corrélé avec le 1^{er} attribut (FIG. 6(a)), pour la 5^e classe (les valeurs sont identiques) ;
- le dernier attribut (FIG. 6(d)) représenté n'est, de façon évidente, pertinent pour aucune des classes.

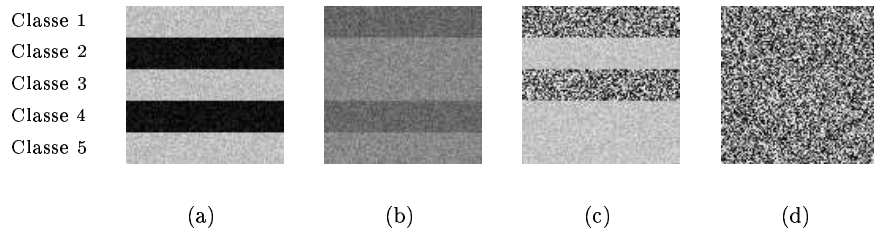


FIG. 6 – Données artificielles

Nous avons réalisé deux séries d'expériences sur ces données :

- une série avec *K*-means, initialisé pour 5 classes ;
- une série avec notre méthode, en utilisant 5 populations.

Afin de comparer les résultats, nous avons évalué la qualité réelle des classifications obtenues en mettant en correspondance les classes extraites avec les classes réelles. Pour cela, nous avons utilisé une matrice de confusion. Pour chaque classe réelle, on sélectionne la classe extraite qui la représente le mieux. Si une classe extraite correspond à plusieurs classes réelles, on choisit celle qui a le plus d'objets en commun. On calcule alors le rapport entre le nombre d'objets correctement classifiés et le nombre total d'objets. Ce calcul est illustré par les deux exemples sur la figure 7. Sur la figure 7(a), la 1^{re} classe extraite correspond aux classes réelles 1, 3 et 5, mais elle a plus d'objets communs avec la classe 1. De même pour la 4^e classe extraite qui correspondra à la classe réelle 4. La qualité finale sera ainsi de $\frac{1611+1499}{10000} = 0,311$. Sur la figure 7(b), une et une seule classe extraite correspond à chacune des classes réelles. La qualité totale est de 0,9006. La figure 8(a) présente un résultat représentatif avec la méthode *K*-means. La qualité est de 18,65%. Sur la figure 8(c) est représenté un résultat représentatif avec notre méthode de pondération d'attributs. La qualité est alors de 90,06%. La courbe sur la figure 8(d) représente l'évolution de la qualité de la répartition de référence au cours de l'apprentissage. La courbe en pointillée sur la figure 8(e) représente l'évolution de la qualité réelle de l'unification brute, c'est-à-dire en n'affectant aucune classe aux objets qui ne sont pas 1-classifiés. Enfin, la courbe en trait plein représente la qualité réelle de l'unification relative. On remarque que l'évolution de la qualité calculée par notre méthode correspond bien à l'évolution de la qualité de l'unification brute. En-

Classes réelles	Classes extraites					Classes correspondantes
	1	2	3	4	5	
1	1611	297	16	70	6	1
2	0	0	449	1209	342	4
3	1039	534	285	142	0	1
4	0	0	0	1499	501	4
5	1176	824	0	0	0	1

(a) Mauvaise classification

Classes réelles	Classes extraites					Classes correspondantes
	1	2	3	4	5	
1	2000	0	0	0	0	1
2	0	709	1291	0	0	3
3	272	0	0	1728	0	4
4	0	1991	9	0	0	2
5	4	0	0	0	1996	5

(b) Bonne classification

FIG. 7 – Correspondance entre classes extraites et classes réelles

fin, globalement la qualité de l'unification relative augmente, bien que présentant de nombreuses oscillations, mais reste toujours supérieure à celle de l'unification brute.

3.2 Données réelles

Nous avons également appliqué notre méthode sur des données réelles dans le cadre de la télédétection. Il s'agissait soit de classer des images dites hyperspectrales, c'est-à-dire des images qui comportent de nombreuses données radiométriques pour chaque pixel (cf. 3.2.1), soit de classer des régions obtenues après la segmentation d'une image (cf. 3.2.2), ces régions étant représentées par de nombreux attributs de types variés (réels, histogrammes, etc.).

3.2.1 Classification d'images hyperspectrales

Sur la figure 9, nous avons représenté 5 bandes d'un extrait d'une image hyperspectrale¹ d'un quartier de Strasbourg. Il s'agit d'une image DAIS de 152×156 pixels dont on a extrait 44 bandes sur 79. Ainsi, la bande 18 (FIG. 9(a)) semble être pertinente et non bruitée. Par contre, les bandes 24 et 26 (FIG. 9(b) et FIG. 9(c)) apparaissent fortement corrélées. Enfin, la bande 29 (FIG. 9(d)) semble ne porter aucune informa-

¹Image gracieuse fournie par l'équipe TRIO du LSIIT

Algorithme génétique de pondération non supervisée d'attributs

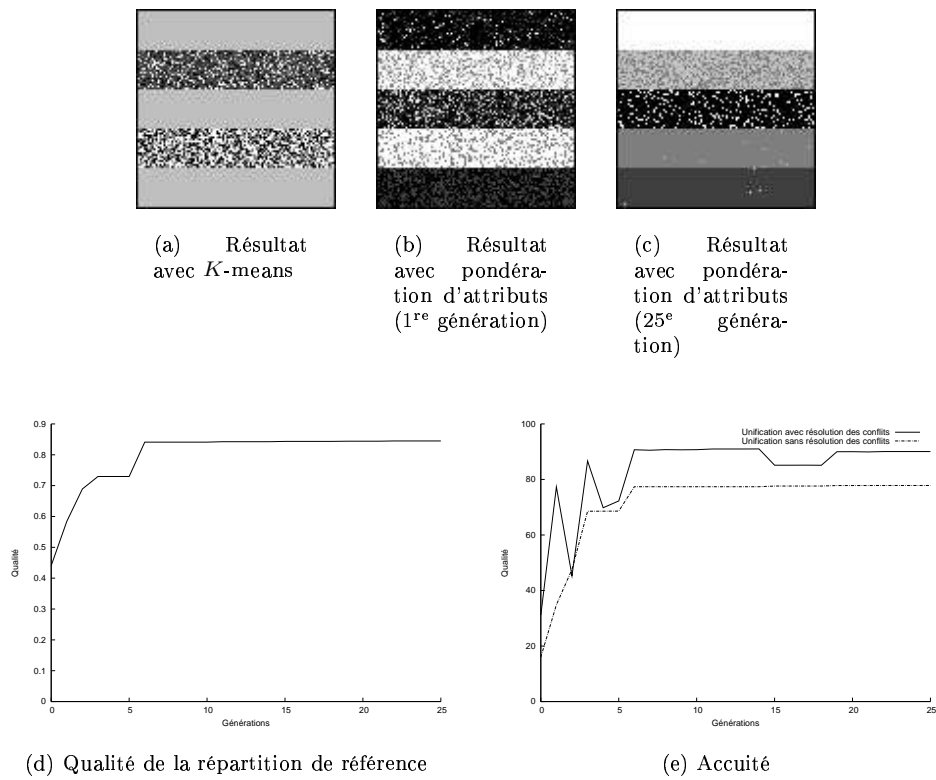


FIG. 8 – Apprentissage sur des données artificielles

tion, alors que la bande 31 (FIG. 9(e)) est visiblement très bruitée. On voit bien qu'une pondération de ces bandes devrait permettre d'obtenir une meilleure classification.

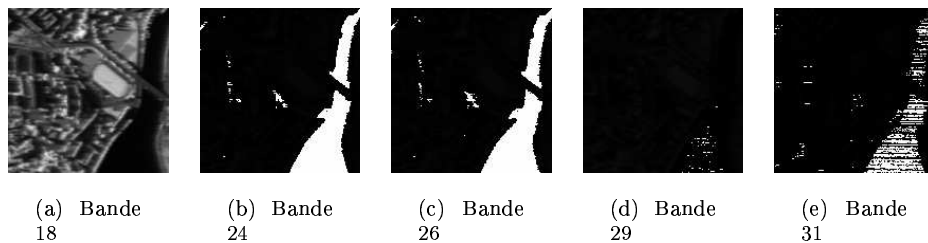


FIG. 9 – Quelques bandes spectrales de l'image

Nous avons donc appliqué notre méthode sur cet extrait. L'algorithme a été paramétré pour rechercher 5 classes. Chacune des populations était composée de 150 individus. L'apprentissage s'est déroulé sur 50 générations.

Sur la figure 10, nous pouvons observer l'évolution du critère de qualité totale d'une répartition (définition 2.7). Nous remarquons que l'évolution présente 4 phases :

- de la génération 0 à 12, forte amélioration de la qualité de la classification ;
- de la génération 13 à 28, amélioration très lente, voire stagnation de cette qualité ;
- de la génération 29 à 37, nouvelle augmentation très forte de la qualité ;
- de la génération 38 à 50, augmentation très lente.

Sur la figure 11 sont représentés les résultats correspondant aux différents points d'inflexion (les classes de la répartition de référence, ainsi que l'ensemble des objets 0-classifiés ou k -classifiés, avec $k > 1$). Sur la table 1 est indiqué le pourcentage de pixels 0-classifiés, c'est-à-dire, les pixels qui ne sont dans aucune classe.

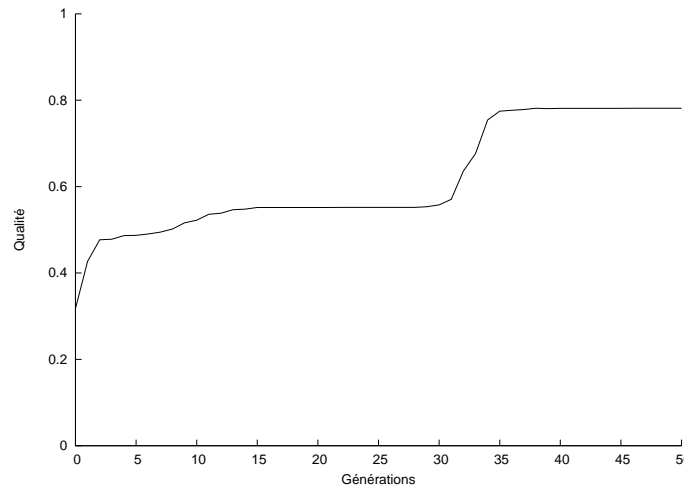


FIG. 10 – Évolution de la qualité

Génération 12	Génération 22	Génération 29	Génération 31	Génération 46
75,67%	72,24%	70,13%	58,78%	26,05%

TAB. 1 – Pourcentage de pixels 0-classifiés

Nous expliquons cette évolution de la manière suivante :

- dans la première période (de 0 à 12), l'amélioration provient principalement de la spécialisation de chacune des populations d'extracteurs ;
- dans la seconde période (13 à 28), les spécialisations trouvées dans l'étape précédente n'ont plus permis d'évolution significative de la qualité, ceci étant vraisemblablement dû à la présence d'un maximum local ;































	Classe 1	Classe 2	Classe 3	Classe 4	Classe 5	Non classifiés
Génération 12						
Génération 22						
Génération 29						
Génération 31						
Génération 46						

FIG. 11 – Résultats de la classification

- à la 29^e génération, la 3^e population vient de se spécialiser dans une classe radicalement nouvelle par rapport à la génération précédente, mais aussi, et surtout, par rapport aux autres populations de la génération 29. Ceci explique la première inflexion de la courbe après cette phase de stagnation. En effet cette nouvelle spécialisation a permis de classer des objets qui ne l'étaient pas encore (baisse du nombre d'objets 0-classifiés de 72,24% à 70,13% en une génération) ;
- cette amélioration est immédiatement suivie à la 31^e génération par une nouvelle amélioration encore plus forte due au fait que la 1^{re} population propose elle aussi une classe radicalement différente. Cette spécialisation a de plus supprimé un conflit entre la 1^{re} et la 4^e population ;
- les populations ont conservé ces nouvelles spécialisations jusqu'à la fin de l'apprentissage qui n'a plus consisté alors qu'à affiner les classes proposées. Par exemple la 4^e population a pu regrouper l'ensemble des pixels que nous savons être de l'eau.

D'une façon plus générale, nous observons à la 46^e génération que des classes intéressantes ont été découvertes. Ainsi, la 1^{re} population a bien « identifié » les routes, la 2^e les ombres, la 4^e l'eau et la 5^e la végétation (un stade et des espaces verts). Le 3^e a

mis en  vidence les pixels de bordures, difficiles   classifier.

Les objets non classifi s sont des objets de types tr s vari s, chacun des types  tant tr s peu repr sent s dans l'image, et donc difficile   classifier. En effet, ce sont souvent des pixels mixtes (pixel   la fronti re des zones diff rentes).

Nous avons  galement test  si les pond rations obtenues pour chaque population  taient identiques   chaque ex cution de l'algorithme. L'algorithme converge chaque fois vers la m me classification globale (m me si deux classes identiques d'une ex cution   l'autre ne sont pas obtenues par la m me population), mais, contrairement   nos attentes, tr s peu de similitudes ont  t  observ es quant aux pond rations des attributs. Cela est principalement d  aux d pendances entre les caract ristiques qui produisent des redondances d'informations. Il est alors possible d'avoir plusieurs distances tr s diff rentes qui produisent des classes identiques. Il est  galement possible qu'une caract ristique ne soit que tr s peu significative pour un type d'objets, et donc que la classification ne change pas quelque soit le poids donn    cette caract ristique.

3.2.2 Classification de r gions

Sur la figure 12, nous pouvons voir deux extraits des r sultats sur une image haute r solution de 400×400 pixels. Nous avons classifi  cette image par deux m thodes : une classification radiom trique et notre m thode appliqu e   la classification de r gions obtenues apr s une segmentation de l'image (ces r gions sont repr sent es par une dizaine d'attributs de types vari s). Dans les deux cas, 5 classes ont  t  demand es. On voit tr s clairement, que les b timents ont  t  mieux d tect s sur la figure 12(c) que sur la figure 12(b). En effet, sur le premier extrait, avec notre m thode, le toit du b timent pr sente beaucoup moins de trous qu'avec une classification classique. Sur le second extrait, il s'agit d'un ensemble de toits, d coup  en de nombreuses classes avec une classification radiom trique, qui appar it beaucoup plus clairement avec notre m thode.

Ces r sultats sont tr s prometteurs et montre le bien-fond  de notre m thode. En effet, m me s'il y a encore des erreurs de classification, notre m thode a pu d couvrir les classes importantes de mani re totalement non supervis e.

4 Conclusion

Nous avons d fini une m thode de pond ration d'attributs non supervis e par approche *wrapper*. Cette m thode calcule des pond rations continues et locales aux classes. Elle a  t  d finie pour traiter des donn es repr sent es par des attributs nombreux, qui peuvent  tre bruit s, corr l s ou non pertinents.

Nous avons d fini une approche modulaire de classification qui, au lieu de chercher   classifier toutes les donn es de fa on monolithique, utilise des experts locaux, sp cialis s chacun dans une classe.

Les classes  tant construites de mani re ind pendante les unes des autres, il a  t  n cessaire de d finir un nouveau crit re de qualit  de classification.

L'apprentissage est assur  par un algorithme de co volution coop rative, les populations collaborant pour optimiser le crit re global de qualit .

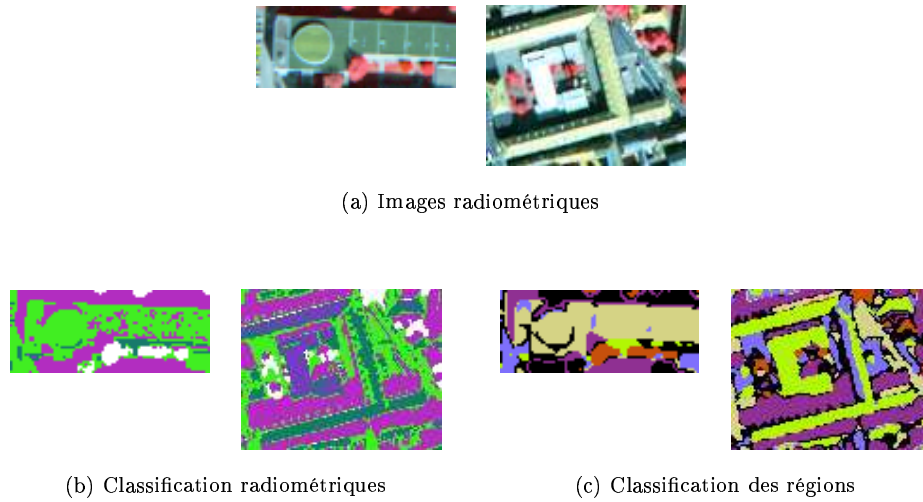


FIG. 12 – Résultats sur une image haute résolution de l'Esplanade (Strasbourg)

Nous avons défini une méthode générale qui peut être employée avec tous types d'algorithme de classification non supervisée et tous critères de qualité de classe, et donc peut faire face à tout types de données. En effet, il suffit de définir les extracteurs à partir de méthodes d'analyse spécifiques aux données à traiter (par exemple, les données symboliques) pour pouvoir appliquer notre méthode. Nous l'avons testée avec un algorithme de classification basé sur la distance, mais la méthode étant complètement indépendante de la notion de similarité ou de distance, elle peut être utilisée avec d'autres algorithmes de classification.

Bien que nos résultats soient encourageants, il reste de nombreux points à éclaircir. En particulier, la fonction de qualité que nous avons définie modélise bien, sur les données sur lesquelles nous avons testé notre méthode, la qualité de l'unification brute. Néanmoins, une définition floue de la qualité globale de classification permettrait de modéliser la qualité de l'unification relative.

Notre approche présente certaines limites. Tout d'abord, les algorithmes génétiques, en particulier les méthodes par coévolution ont des limites bien connues, en particulier des temps d'exécution relativement longs (plusieurs heures sur les données présentées). Comme cela dépend fortement de la méthode de classification utilisée pour définir les extracteurs, il est difficilement envisageable d'utiliser des méthodes complexes demandant un grand temps de calcul à eux seuls.

De plus, il est actuellement nécessaire de connaître a priori le nombre de classes dans les données. C'est un problème commun à de nombreuses méthodes de classification non supervisée, comme l'algorithme K -means qui est pourtant une méthode très utilisée. Certaines méthodes ont été proposées pour tester si le nombre de classe est optimal,

mais cela nécessite d'exécuter l'algorithme plusieurs fois, avec un nombre de classes différent à chaque fois. Nous travaillons actuellement sur une possibilité de faire varier le nombre de classes en cours d'apprentissage, en éliminant les populations provoquant de nombreux conflits avec les autres et en rajoutant des populations si une partie des données n'est extraite par aucun extracteur. Il est à noter que dans le domaine de la géographie, le nombre de classes K est souvent connu a priori.

Enfin, un problème est apparu lors d'un test sur des données réelles : dans le cadre du projet de recherche européen TIDE², nous avons été amené à classifier une image de la lagune de Venise. Or sur cette image, l'eau est une classe très compacte quelque soit la dimension observée (c'est-à-dire la bande spectrale). Ainsi quelque soit la pondération utilisée, la meilleure classe obtenue par un extracteur sera l'eau. Il a donc été nécessaire de pré-traiter l'image afin de supprimer les pixels correspondant à la classe d'eau.

D'une manière plus générale, nous comptons nous intéresser à l'utilisation de connaissances du domaine pour améliorer chaque étape de l'apprentissage.

Références

- [Bauer et Kohavi, 1999] E. Bauer et R. Kohavi. An empirical comparison of voting classification algorithms : Bagging, boosting, and variants. *Machine Learning*, 36 :105–142, 1999.
- [Bel Mufti et Bertrand, 1997] G. Bel Mufti et P. Bertrand. Validation d'une classe par rééchantillonnage. In *Cinquième rencontres de la Société Francophone de Classification, Lyon*, pages 251–254, 1997.
- [Bloch *et al.*, 2001] I. Bloch, A. Hunter, A. Appriou, A. Ayoun, S. Benferhat, P. Bernard, L. Cholvy, R. Cooke, F. Cuppens, D. Dubois, H. Fargier, M. Grabisch, R. Kruse, J.Lang, S. Moral, H. Prade, A. Saffotti, P. Smets, et C. Sossai. Fusion : General concepts and characteristics. *International Journal of Intelligent Systems*, 16 :1107–1134, 2001.
- [Blum et Langley, 1997] Avrim Blum et Pat Langley. Selection of relevant features and examples in machine learning. *Artificial Intelligence*, 97(1–2) :245–271, 1997.
- [Bock et Diday, 2000] H.-H. Bock et E. Diday. *Analysis of Symbolic Data. Exploratory methods for extracting statistical information from complex data*. Springer Verlag, 2000.
- [Bolshakova et Azuaje, 2003] N. Bolshakova et F. Azuaje. Cluster validation techniques for genome expression data. *Signal processing*, 83(4) :825–833, 2003.
- [Chan *et al.*, 2004] E.Y. Chan, W.K. Ching, M.K. Ng, et J.Z. Huang. An optimization algorithm for clustering using weighted dissimilarity measures. *Pattern Recognition*, 37 :943–952, 2004.
- [Chavent et Lechevallier, 2002] M. Chavent et Y. Lechevallier. Dynamical clustering of interval data. optimization of an adequacy criterion based on hausdorff distance. pages 53–60. 2002.

²Tidal Inlets Dynamics Environment

- [Cliff et Miller, 1995] Dave Cliff et Geoffrey F. Miller. Tracking the red queen : Measurements of adaptive progress in co-evolutionary simulations. In *European Conference on Artificial Life*, pages 200–218, 1995.
- [Dy et Brodley, 2000] Jennifer G. Dy et Carla E. Brodley. Feature subset selection and order identification for unsupervised learning. In *Proceedings 17th International Conf. on Machine Learning*, pages 247–254. Morgan Kaufmann, San Francisco, CA, 2000.
- [Fisher, 1987] D. H. Fisher. Knowledge acquisition via incremental conceptual clustering. *Machine learning*, 2 :139–172, 1987.
- [Floreano et Nolfi, 1997] D. Floreano et S. Nolfi. God save the red queen! competition in co-evolutionary robotics. In *Genetic Programming 1997 : Proceedings of the Second Annual Conference*, pages 398–406, 1997.
- [Frigui et Nasraoui, 2004] H. Frigui et O. Nasraoui. Unsupervised learning of prototypes and attribute weights. *Pattern Recognition*, 34 :567–581, 2004.
- [Garai et Chaudhuri, 2004] G. Garai et B.B. Chaudhuri. A novel genetic algorithm for automatic clustering. *Pattern Recognition Letters*, 25 :173–187, 2004.
- [Goldberg, 1989] D.E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., 1989.
- [Grabmeier et Rudolph, 2002] J. Grabmeier et A. Rudolph. Techniques of cluster algorithms in data mining. *Data Mining and Knowledge Discovery*, 6(4) :303–360, 2002.
- [Günter et Burke, 2001] S. Günter et H. Burke. Validation indices for graph clustering. In *Proc. 3rd IAPR- TC15 Workshop on Graph-based Representations in Pattern Recognition*, pages 229–238. J.-M. Jolion, W. Kropatsch, M. Vento, 2001.
- [Halkidi et al., 2001] M. Halkidi, Y. Batistakis, et M. Vazirgiannis. On clustering validation techniques. *Journal of Intelligent Information Systems*, 17(2-3) :107–145, 2001.
- [Holland, 1975] J.H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975.
- [Howe et Cardie, 1997] N. Howe et C. Cardie. Examining locally varying weights for nearest neighbor algorithms. In *ICCBR*, pages 455–466, 1997.
- [John et al., 1994] G.H. John, R. Kohavi, et K. Pfleger. Irrelevant features and the subset selection problem. In *Proceedings of the Eleventh International Conference on Machine Learning*, pages 121–129, 1994.
- [Jolliffe et al., 1966] I.T. Jolliffe, P. Adriaans, et D. Zantinge. *Principal Component Analysis*. Springer Verlag, 1966.
- [Kittler, 1998] J. Kittler. On combining classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(3) :226–239, 1998.
- [Lennon, 2002] M. Lennon. *Méthodes d'analyse d'images hyperspectrales*. PhD thesis, Université de Rennes I, 2002.
- [Levine et Domany, 2001] E. Levine et E. Domany. Resampling method for unsupervised estimation of cluster validity. *Neural Computation*, 13(11) :2573–2593, 2001.

- [MacQueen, 1965] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297, Berkeley, CA, 1965. University of California Press.
- [Mayer, 1998] H.A. Mayer. Symbiotic coevolution of artificial neural networks and training data sets. In *Proceedings of the Fifth International Conference on Parallel Problem Solving from Nature*, pages 511–520, 1998.
- [Murthy et Chowdhury, 1996] C.A. Murthy et N. Chowdhury. In search of optimal clusters using genetic algorithms. *Pattern recognition letters*, 17 :825–832, 1996.
- [Nicolayannis *et al.*, 1997] N. Nicolayannis, M. Terrenoireand, et D. Tounissoux. Pertinence d’une classification. In *Cinqui me rencontres de la Soci t  Francophone de Classification, Lyon*, pages 257–259, 1997.
- [Paredis, 1997] J. Paredis. Coevolving cellular automata : Be aware of the red queen ! In *7th Int. Conference on Genetic Algorithms (ICGA 97)*, pages 393–400, 1997.
- [Potter *et al.*, 1995] M.A. Potter, K.A. De Jong, et J.J. Grefenstette. A coevolutionary approach to learning sequential decision rules. In *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 366–372, 1995.
- [Potter et De Jong, 1994] M.A. Potter et K.A. De Jong. A cooperative coevolutionary approach to function optimization. In *Proceedings of the Third Conference on Parallel Problem Solving from Nature*, pages 249–257, 1994.
- [Renders, 1995] J.-M. Renders. *Algorithmes g n tiques et r seaux de neurones*. Hermes, 1995.
- [Tibshirani *et al.*, 2001] R. Tibshirani, G. Walther, D. Botstein, et P. Brown. Cluster validation by prediction strength. Technical report, Department of Biostatistics, Stanford University, 2001.
- [Tseng et Yang, 2001] L.Y. Tseng et S.B. Yang. A genetic approach to the automatic clustering problem. *Pattern recognition*, 34 :415–424, 2001.
- [Wemmert *et al.*, 1999] C. Wemmert, P. Gan arski, et J.J. Korczak. Un syst me de raffinement non-supervis  d’un ensemble de hi rarchies de classes. In M. Sebag, editor, *Actes de la Premi re Conf rence d’Apprentissage, CAP’99*, Palaiseau, France, 1999.
- [Wemmert *et al.*, 2000] C. Wemmert, P. Gan arski, et J.J. Korczak. A collaborative approach to combine multiple learning methods. *International Journal on Artificial Intelligence Tools*, 9(1) :59–78, 2000.
- [Wettschereck *et al.*, 1997] D. Wettschereck, D.W. Aha, et T. Mohri. A review and empirical evaluation of feature weighting methods for a class of lazy learning algorithms. *Artificial Intelligence Review*, 11(1-5) :273–314, 1997.
- [Wettschereck et Aha, 1995] D. Wettschereck et D.W. Aha. Weighting features. In Manuela Veloso et Agnar Aamodt, editors, *Case-Based Reasoning, Research and Development, First International Conference*, pages 347–358, Berlin, 1995. Springer Verlag.

Summary

Clustering complex objects, composed by a large number of features is often difficult. There often are correlated features, and noisy or irrelevant features. To solve this problem, an unsupervised feature weighting method is proposed. It is a wrapper approach which searches for local continuous weights.

In this method, a set of extractors (of individual cluster) is defined. Each extractor extract one single class, using a specific strategy. Each extractor uses a different set of feature weights. The global result is obtained by the union of all extracted clusters.

In order to find an optimal set of weights, according to a clustering quality criterion, a cooperative coevolution learning is used.

In this paper, we explain our method, particularly, how extractors are defined, how clustering quality is evaluated, how the weights are discovered and how is the final clustering built. We also present first results on artificial data and remote sensing images.