



DÉPARTEMENT TSI

Classification et reconnaissance des formes

DEA IARFA

Isabelle BLOCH

ENST, département TSI, CNRS URA 820, 46 rue Barrault, 75634 Paris Cedex 13
Tél : 01 45 81 75 85, Fax : 01 45 81 37 94, E-mail : Isabelle.Bloch@enst.fr

28 octobre 2003

Table des matières

1	Présentation de la reconnaissance des formes en traitement d'images	5
1.1	Introduction	5
1.2	Définitions	5
1.3	Primitives et espace de représentation	6
2	Approches statistiques	9
2.1	Théorie bayésienne de la décision	9
2.2	Séparation linéaire	10
2.3	Méthode de k plus proches voisins (k-ppv)	10
2.4	Classification automatique : méthodes non hiérarchiques	11
2.4.1	Hypothèses des structures probabilistes connues	11
2.4.2	Méthode simple d'approximation	14
2.4.3	Critères de classification	14
2.4.4	K-moyennes	17
2.4.5	Algorithme ISODATA	18
2.4.6	Boules optimisées	20
2.4.7	Nuées dynamiques	20
2.4.8	Limites	21
2.5	Méthodes hiérarchiques	22
2.5.1	Distance ultra-métrique	22
2.5.2	Distance de chaîne	23
2.5.3	Relation d'équivalence et partition à partir d'une ultra-métrique	24
2.5.4	Hiérarchie et hiérarchie indicée	25
2.5.5	Équivalence entre hiérarchie indicée et ultra-métrique	25
2.5.6	Arbre de longueur minimum	27
2.5.7	Limites	27

3	Approches structurelles	29
3.1	Comparaison de chaînes	29
3.1.1	Définitions	29
3.1.2	Distance de Hamming	30
3.1.3	Inclusion d'une chaîne dans une autre	30
3.1.4	Distance d'édition de chaîne	30
3.2	Grammaires et automates	31
3.2.1	Définitions	31
3.2.2	Automates finis	32
3.2.3	Grammaires et reconnaissance des formes	32
3.2.4	Utilisation des grammaires en traitement et interprétation des images	33
3.3	Arbres et graphes	33
3.3.1	Définitions	33
3.3.2	Distance entre arbres	34
3.3.3	Isomorphismes de graphes	34
3.3.4	Extensions	34
3.3.5	Graphes d'association	34
3.3.6	Utilisations en traitement et interprétation d'images	35
4	Ouverture vers d'autres approches	37
4.1	Ensembles flous et possibilités	37
4.1.1	Définitions	37
4.1.2	Classification floue	38
4.2	Fonctions de croyance	40
4.3	Réseaux de neurones	40
4.4	Méthodes symboliques	41
4.4.1	Systèmes à base de connaissances	42
4.4.2	Modes de raisonnement et inférence	43

Chapitre 1

Présentation de la reconnaissance des formes en traitement d'images

1.1 Introduction

Les besoins en reconnaissances des formes :

- production d'une information résumée permettant de choisir
- représentation des connaissances et des observations
- expliquer les observations
- interpréter les résultats fournis par un classifieur ou discriminateur afin de guider le choix
- gérer les incohérences dans les observations ou entre certaines observations et les connaissances
- maintien de la cohérence entre les différentes étapes ou niveaux de la reconnaissance
- fusion multi-sources
- limites des probabilités, nécessité de rechercher d'autres méthodes (problèmes de la subjectivité, de l'ignorance, des événements rares, etc., problèmes de la quantification et de l'estimation)
- développement de techniques plus récentes, souvent issues de l'IA (flou, logique, etc.)
- réduire l'espace de recherche (cf maintien de la cohérence)

1.2 Définitions

Reconnaissance d'une forme parmi plusieurs possibilités à partir d'observations bruitées de celles-ci

□ Distinctions usuelles :

- classification supervisée / non supervisée
- paramétrique / non paramétrique
- structurel, syntaxique (modélisation des relations entre composantes des formes) / numérique (modèles probabilistes, flous, etc. des formes)

- Forme = un point d'un espace de description (de caractéristiques)
 - existence d'une métrique → raisonnement sur les vecteurs
 - pas de métrique → approche de type « fusion d'informations »

- Ensemble des classes = espace de décision
- Classe = zone de l'espace de caractéristiques
- Classification = attribution d'une observation à une classe

- Difficultés :
 - choix du nombre de classes ou zones
 - modélisation des classes
 - définition des frontières (décision)
 - recouvrement entre les zones

- Rejet :
 - en distance
 - en ambiguïté
 ⇒ reclassification, adaptation du système

- Trois étapes :
 - définition de l'espace de caractéristiques
 - définition des classes à partir des informations dont on dispose (ou que l'on peut apprendre)
 - modèles probabilistes
 - exemples, prototypes
 - etc.
 - construction d'une règle de décision / rejet

- Critères de performance :
 - matrice de confusion
 - probabilité d'erreur
 - etc.
 - difficulté : existence d'une « vérité », accès à cette vérité

1.3 Primitives et espace de représentation

- Niveaux de gris seulement :
 - raisonnement et modélisation à partir de l'histogramme
 - très peu d'information

□ Prise en compte de l'information spatiale

– localement : voisinages

– plus largement : relations spatiales entre entités

segments, lignes, polygones, contours (codage de Freeman, approximations polygonales, descripteurs de Fourier), mesures géométriques (moments, surface, compacité, allongement, diamètre, périmètre, courbure...), mesures topologiques (nombre de composantes connexes, nombre de trous...), squelette

Chapitre 2

Approches statistiques

Cette partie vise à donner un aperçu des méthodes numériques de classification. Des notions plus approfondies sur ces méthodes peuvent être trouvées dans les références [9, 12, 8, 10]. Des exemples de méthodes probabilistes et de méthodes automatiques seront présentés. Les secondes sont souvent utilisées quand on manque de connaissance (par exemple sur les lois probabilistes), ce qui conduit à faire des hypothèses fortes sur la structure de l'espace de décision (métrique sur l'espace des paramètres ou espace de représentation, forme des nuages des classes). Les méthodes donneront donc de bons résultats si les hypothèses sont vérifiées, et se dégraderont au fur et à mesure qu'on s'en éloigne. Ces méthodes nécessitent ainsi un bon savoir-faire de l'utilisateur, qui vient en quelque sorte remplacer l'apprentissage supervisé.

2.1 Théorie bayésienne de la décision

Nous ne faisons ici qu'un rappel succinct de cette méthode, les approches bayésiennes étant plus largement détaillées dans le cours *Statistiques pour l'apprentissage*.

On suppose que la forme à reconnaître est représentée par un vecteur x de l'espace \mathbb{R}^n (l'espace de caractéristiques) que l'on veut attribuer à une des classes ω_i de l'espace de décision Ω .

Soient $P(\omega_i)$ les probabilités a priori des classes, et $p(x|\omega_i)$ les densités de probabilités des mesures x conditionnellement aux classes. On suppose ces probabilités connues, ou résultant d'une étape d'apprentissage. Il peut s'agir soit d'un apprentissage paramétrique, dans lequel on suppose que $p(x|\omega_i)$ suit une loi connue (gaussienne par exemple) et on n'en apprend que les paramètres, soit d'un apprentissage non paramétrique, dans lequel on ne fait pas d'hypothèse sur la loi (on utilise par exemple l'histogramme ou l'histogramme lissé de x dans une région connue comme faisant partie de ω_i , suffisamment grande pour que les statistiques soient significatives).

La règle de Bayes permet de calculer la probabilité a posteriori :

$$P(\omega_i|x) = \frac{p(x|\omega_i)P(\omega_i)}{p(x)} = \frac{p(x|\omega_i)P(\omega_i)}{\sum_i p(x|\omega_i)P(\omega_i)}.$$

La décision selon la règle du maximum a posteriori revient à choisir la classe ω_i qui maximise

cette probabilité, c'est-à-dire telle que :

$$\forall j, P(\omega_i|x) \geq P(\omega_j|x).$$

Dans le cas de deux classes, la probabilité d'erreur vaut :

$$P(E) = \int_{R_1} p(x|\omega_2)P(\omega_2)dx + \int_{R_2} p(x|\omega_1)P(\omega_1)dx,$$

où R_1 (respectivement R_2) est la région de l'espace de représentation dans laquelle x est classé dans ω_1 (respectivement ω_2) alors qu'il appartient à ω_2 (respectivement ω_1). La règle de décision bayésienne, du maximum a posteriori, minimise cette erreur (partie hachurée sur la figure 2.1).

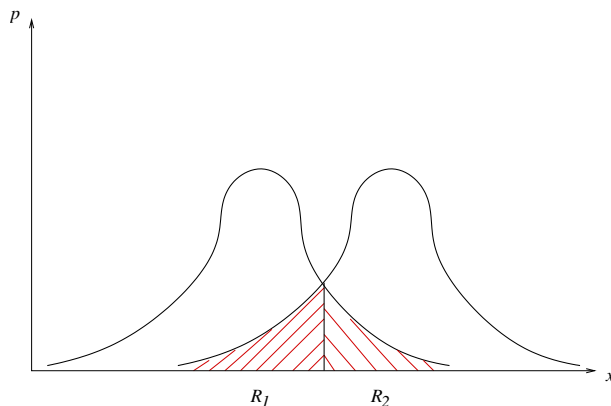


FIG. 2.1 – Décision bayésienne : minimisation de la probabilité d'erreur (zone hachurée). Courbe de gauche : $p(x|\omega_1)P(\omega_1)$. Courbe de droite : $p(x|\omega_2)P(\omega_2)$.

Des coûts de décision peuvent également être introduits.

2.2 Séparation linéaire

En supposant toujours les formes représentées par des vecteurs de \mathbb{R}^n , on dit que deux classes sont séparables linéairement s'il existe un hyperplan tel que tous les points d'une classe se trouvent dans un même demi-espace¹. Il s'agit en général des points d'apprentissage. Pour un nombre de classes supérieur à deux, on cherchera les hyperplans séparateurs en considérant les classes deux par deux.

Une fois les hyperplans séparateurs déterminés, par exemple par l'algorithme du perceptron (optimisation itérative par une méthode de gradient), la reconnaissance de formes ne faisant pas partie de l'ensemble d'apprentissage est effectuée simplement en déterminant dans quelle zone de \mathbb{R}^n , délimitée par ces hyperplans, se trouve la forme.

2.3 Méthode de k plus proches voisins (k-ppv)

La méthode des k-ppv est une méthode de classification non paramétrique. On suppose qu'on dispose d'un ensemble d'apprentissage, c'est-à-dire d'un ensemble de formes (toujours

¹Il arrive bien sûr très souvent que les classes ne soient pas linéairement séparables.

représentées par des vecteurs de \mathbb{R}^n) dont on connaît l'appartenance aux classes de l'espace de décision Ω . Soit x une forme à reconnaître. Etant donnée une métrique d sur \mathbb{R}^n , on détermine les k points x_1, \dots, x_k de l'ensemble d'apprentissage les plus proches de x au sens de d . La règle de décision consiste à affecter x à la classe majoritairement représentée dans les appartenances des x_i .

Les règles de rejet usuelles sont les suivantes :

- rejet en ambiguïté si la classe choisie n'est pas représentée par au moins k' voisins parmi les k ;
- rejet en distance si les plus proches voisins sont à une distance de x supérieure à un certain seuil.

On peut montrer que la probabilité d'erreur dans la méthode du 1-ppv ($k = 1$), notée P_1 est reliée à la probabilité d'erreur bayésienne $P(E)$ par l'inégalité suivante :

$$n \rightarrow +\infty \Rightarrow P(E) \leq P_1 \leq 2P(E)$$

où n est le cardinal de l'ensemble d'apprentissage.

2.4 Classification automatique : méthodes non hiérarchiques

Partant de l'hypothèse où les formes des densités de probabilités sous-jacentes sont connues, les méthodes non hiérarchiques se ramènent à un problème d'estimation de paramètres, très proche de ceux traités dans les chapitres précédents, en particulier de celui de l'apprentissage supervisé. La limite imposée par ce type d'hypothèse conduit à reformuler le problème de manière complètement différente, sous la forme d'un problème de partitionnement ou regroupement des objets à classifier selon certains critères. C'est cette formulation qui conduit à la plupart des algorithmes utilisés.

2.4.1 Hypothèses des structures probabilistes connues

On suppose ici que les paramètres suivants sont connus :

- nombre de classes c (les classes sont notées ω_i , $i = 1, \dots, c$),
- probabilités a priori $P(\omega_i)$ ($i = 1, \dots, c$),
- probabilités conditionnelles $p(x|\omega_i, \theta_i)$, où les θ_i sont les paramètres des lois de probabilités relatives à chaque classe, et x désigne un échantillon quelconque à classer.

Les inconnues sont donc les paramètres θ_i . La loi des probabilités totales s'exprime alors par :

$$p(x|\theta) = \sum_{i=1}^c p(x|\omega_i, \theta_i)P(\omega_i), \quad (2.1)$$

où θ désigne le vecteur de paramètres $\theta_1, \dots, \theta_c$. Cette équation exprime que la loi $p(x|\theta)$ est une loi de mélange.

La reconnaissance à partir d'un tel modèle suppose que le phénomène soit **identifiable**, c'est-à-dire que pour deux vecteurs de paramètres différents, les probabilités conditionnelles doivent être différentes pour au moins un échantillon x :

$$\theta \neq \theta' \Rightarrow \exists x \text{ tel que } p(x|\theta) \neq p(x|\theta'). \quad (2.2)$$

L'hypothèse d'identifiabilité est généralement admise.

La solution la plus fréquemment adoptée pour estimer les paramètres consiste à utiliser l'estimateur du maximum de vraisemblance, à partir de n échantillons x_1, \dots, x_n , et donc à chercher θ qui maximise l'expression :

$$p(x_1, \dots, x_n | \theta). \quad (2.3)$$

On choisit de plus souvent des échantillons indépendants, ce qui simplifie les calculs puisqu'on a alors :

$$p(x_1, \dots, x_n | \theta) = \prod_{k=1}^n p(x_k | \theta). \quad (2.4)$$

La résolution s'effectue en passant au logarithme (pour supprimer le produit) et à dériver l'expression, en supposant que $p(x_1, \dots, x_n | \theta)$ est différentiable :

$$\begin{aligned} \frac{\partial \log p(x_1, \dots, x_n | \theta)}{\partial \theta_i} &= \sum_{k=1}^n \frac{\partial \log p(x_k | \theta)}{\partial \theta_i} \\ &= \sum_{k=1}^n \frac{1}{p(x_k | \theta)} \frac{\partial}{\partial \theta_i} \left[\sum_{j=1}^c p(x_k | \omega_j, \theta_j) P(\omega_j) \right] \\ &= \sum_{k=1}^n \frac{1}{p(x_k | \theta)} \frac{\partial}{\partial \theta_i} [p(x_k | \omega_i, \theta_i) P(\omega_i)] \\ &= \sum_{k=1}^n \frac{P(\omega_i | x_k, \theta)}{p(x_k | \omega_i, \theta_i)} \frac{\partial p(x_k | \omega_i, \theta_i)}{\partial \theta_i} \\ &= \sum_{k=1}^n P(\omega_i | x_k, \theta) \frac{\partial \log p(x_k | \omega_i, \theta_i)}{\partial \theta_i}. \end{aligned}$$

Ce calcul suppose que θ_i et θ_j sont fonctionnellement indépendants pour $i \neq j$. L'estimation du maximum de vraisemblance revient donc à résoudre, pour chaque θ_i , l'équation :

$$\sum_{k=1}^n P(\omega_i | x_k, \hat{\theta}) \frac{\partial \log p(x_k | \omega_i, \hat{\theta}_i)}{\partial \theta_i} = 0. \quad (2.5)$$

Cette méthode est généralisable au cas où les $P(\omega_i)$ sont inconnues, en effectuant cette fois la maximisation sous les contraintes suivantes :

$$\begin{cases} \forall i, P(\omega_i) \geq 0, \\ \sum_{i=1}^c P(\omega_i) = 1. \end{cases} \quad (2.6)$$

On aboutit alors au système suivant (pour $i = 1, \dots, c$) :

$$\begin{cases} \hat{P}(\omega_i) = \frac{1}{n} \sum_{k=1}^n \hat{P}(\omega_i | x_k, \hat{\theta}), \\ \sum_{k=1}^n \hat{P}(\omega_i | x_k, \hat{\theta}) \frac{\partial \log p(x_k | \omega_i, \hat{\theta}_i)}{\partial \theta_i} = 0, \end{cases}$$

avec :

$$\hat{p}(\omega_i | x_k, \hat{\theta}) = \frac{p(x_k | \omega_i, \hat{\theta}_i) \hat{P}(\omega_i)}{\sum_{j=1}^c p(x_k | \omega_j, \hat{\theta}_j) \hat{P}(\omega_j)}.$$

Prenons comme exemple d'application le cas d'un mélange de lois normales, où les inconnues sont les moyennes des lois. Les probabilités conditionnelles ont la forme :

$$p(x|\omega_i, \theta_i) \stackrel{\mathcal{L}}{\equiv} \mathcal{N}(\mu_i, \Sigma_i), \quad (2.7)$$

où $\mathcal{N}(\mu_i, \Sigma_i)$ désigne la loi normale de moyenne μ_i (de même dimension que les échantillons) et de matrice de variance-covariance Σ_i (supposée connue). On a (d désignant la dimension de l'espace des échantillons) :

$$p(x|\omega_i, \mu_i) = \frac{1}{(2\pi)^{d/2} |\Sigma_i|^{1/2}} e^{-\frac{1}{2}(x-\mu_i)^t \Sigma_i^{-1} (x-\mu_i)},$$

d'où :

$$\log p(x|\omega_i, \mu_i) = -\log[(2\pi)^{d/2} |\Sigma_i|^{1/2}] - \frac{1}{2}(x - \mu_i)^t \Sigma_i^{-1} (x - \mu_i),$$

et donc :

$$\frac{\partial}{\partial \mu_i} [\log p(x|\omega_i, \mu_i)] = \Sigma_i^{-1} (x - \mu_i).$$

En dérivant le logarithme de l'équation 2.4, on obtient alors l'équation suivante, où $\hat{\mu} = (\hat{\mu}_1, \dots, \hat{\mu}_c)$ est le vecteur des moyennes estimées pour chaque classe :

$$\sum_{k=1}^n P(\omega_i | x_k, \hat{\mu}) \Sigma_i^{-1} (x_k - \hat{\mu}_i) = 0, \quad (2.8)$$

d'où on tire l'expression de $\hat{\mu}_i$:

$$\hat{\mu}_i = \frac{\sum_{k=1}^n P(\omega_i | x_k, \hat{\mu}) x_k}{\sum_{k=1}^n P(\omega_i | x_k, \hat{\mu})}. \quad (2.9)$$

Cette forme de résultat est très satisfaisante intuitivement, puisqu'on obtient $\hat{\mu}_i$ comme moyenne des échantillons pondérée par les probabilités a posteriori (caractérisant l'appartenance des échantillons à la classe ω_i).

Cependant, cette expression est difficile à calculer (elle fait intervenir les $\hat{\mu}_i$ des deux côtés de l'équation). Si l'on remplace dans cette expression les $P(\omega_i | x_k, \hat{\mu})$ par leur développement selon le théorème de Bayes :

$$P(\omega_i | x_k, \hat{\mu}) = \frac{p(x_k | \omega_i, \hat{\mu}_i) P(\omega_i)}{\sum_{j=1}^c p(x_k | \omega_j, \hat{\mu}_j) P(\omega_j)}, \quad (2.10)$$

avec $p(x|\omega_i, \hat{\mu}_i) \stackrel{\mathcal{L}}{\equiv} \mathcal{N}(\hat{\mu}_i, \Sigma_i)$, on obtient un système d'équations couplées non linéaires, dont la résolution est compliquée.

On lui préfère alors des schémas itératifs où, partant d'une initialisation $\hat{\mu}_i(0)$, on estime les μ_i à l'itération ($j + 1$) en fonction des estimations à l'itération j par :

$$\hat{\mu}_i(j + 1) = \frac{\sum_{k=1}^n P(\omega_i | x_k, \hat{\mu}(j)) x_k}{\sum_{k=1}^n P(\omega_i | x_k, \hat{\mu}(j))}. \quad (2.11)$$

Ce type de schéma d'optimisation ne garantit que l'obtention d'un optimum local, et nécessite donc qu'on parte d'une bonne initialisation $\hat{\mu}_i(0)$ (proche de la solution) pour obtenir l'optimum global.

Malgré les difficultés présentées, le cas où seules les moyennes sont inconnues est de loin le plus simple. Si l'on suppose maintenant que les moyennes, les matrices de variance-covariance Σ_i et les probabilités a priori $P(\omega_i)$ sont inconnues, la méthode du maximum de vraisemblance donne des solutions singulières en général (on peut trouver des exemples pour lesquels la vraisemblance est arbitrairement grande). Des exemples peuvent être trouvés dans [9]. Les solutions adoptées empiriquement consistent à rechercher de bonnes solutions en se restreignant aux maxima finis les plus grands de la fonction de vraisemblance. Il n'en reste pas moins que les calculs sont très lourds. Ils peuvent être toutefois simplifiés si la matrice de covariance est diagonale.

2.4.2 Méthode simple d'approximation

Devant les difficultés présentées dans la partie 2.4.1, des méthodes d'approximation ont été développées afin de simplifier les calculs. Remarquons d'abord que la probabilité a posteriori $P(\omega_i|x_k, \hat{\theta})$ est d'autant plus grande que la distance de Mahalanobis

$$(x_k - \hat{\mu}_i)^t \hat{\Sigma}_i^{-1} (x_k - \hat{\mu}_i)$$

est petite. Cela exprime en effet que l'échantillon x_k a une probabilité grande d'appartenir à la classe ω_i si la distance de x_k au « centre » de la classe (représenté par la moyenne μ_i), pondérée par la dispersion de la classe (représentée par Σ_i), est petite.

Une première approximation consiste à remplacer la distance de Mahalanobis par la distance euclidienne carrée $\|x_k - \hat{\mu}_i\|^2$, permettant de lever ainsi la difficulté de l'estimation des Σ_i . On cherche alors la moyenne $\hat{\mu}_m$ qui minimise $\|x_k - \hat{\mu}_i\|^2$.

Une deuxième approximation porte sur les probabilités a posteriori :

$$\hat{P}(\omega_i|x_k, \hat{\theta}) = \begin{cases} 1 & \text{si } i = m, \\ 0 & \text{sinon,} \end{cases} \quad (2.12)$$

revenant à « binariser » le processus : x_k est considéré comme appartenant de manière certaine à la classe ω_m , et n'appartenant (de façon certaine aussi) à aucune des autres classes.

Ces deux approximations conduisent alors à un schéma itératif très simple, où, à partir d'une estimation initiale des moyennes, on classe les échantillons en fonction de la moyenne la plus proche, puis on recalcule les moyennes selon la formule 2.11, où les probabilités sont simplifiées selon la deuxième approximation. On itère alors ce processus jusqu'à convergence. Cet algorithme se trouve parfois sous le nom de ISODATA de base.

On voit ici apparaître les méthodes non hiérarchiques du deuxième type, où on optimise un critère (ici, la distance au centre des classes) par des schémas itératifs. Les deux approximations effectuées font complètement disparaître les hypothèses sur les probabilités introduites au début de la partie 2.4.1, et permettent donc de faire le lien entre les deux types de méthodes.

2.4.3 Critères de classification

Cette fois, le problème de la répartition de n échantillons x_k (d'un espace E) dans c classes ω_i s'exprime comme l'optimisation d'un certain critère. Les critères les plus souvent employés sont décrits dans cette partie.

On note m_i la moyenne des échantillons appartenant à la classe i :

$$m_i = \frac{1}{n_i} \sum_{x \in \omega_i} x,$$

avec $n_i = \text{Card}\{x | x \in \omega_i\}$, et s_i la variance (scalaire) de la classe i :

$$s_i = \frac{1}{2n_i} \sum_{x \in \omega_i} \sum_{y \in \omega_i} \|x - y\|^2.$$

Le critère de l'erreur quadratique consiste à minimiser l'écart entre les échantillons et les centres des classes auxquelles ils sont affectés :

$$J_e = \sum_{i=1}^c J_i = \sum_{i=1}^c \sum_{x \in \omega_i} \|x - m_i\|^2. \quad (2.13)$$

Le critère de la variance minimum consiste à minimiser la somme des variances des classes :

$$J_v = \sum_{i=1}^c s_i. \quad (2.14)$$

Plus généralement, on peut remplacer la distance euclidienne carrée entre deux échantillons d'une même classe x et y par une fonction de « similarité » $s(x, y)$, dont la forme peut être adaptée au problème spécifique à traiter.

Les critères les plus populaires sont peut-être ceux portant sur la dispersion. En appelant toujours m_i la moyenne de la classe i et n_i sa population, la moyenne m des échantillons s'exprime sous la forme :

$$m = \frac{1}{n} \sum_{x \in E} x = \frac{1}{n} \sum_{i=1}^c n_i m_i.$$

La matrice de variance-covariance de la classe i (dont la trace est s_i), mesurant la dispersion de la classe, s'écrit :

$$S_i = \sum_{x \in \omega_i} (x - m_i)(x - m_i)^t.$$

La dispersion totale intra-classe est obtenue comme somme de ces variances :

$$S_W = \sum_{i=1}^c S_i = J_v. \quad (2.15)$$

La dispersion inter-classes est définie par l'expression suivante :

$$S_B = \sum_{i=1}^c n_i (m_i - m)(m_i - m)^t, \quad (2.16)$$

qui exprime une « distance » entre classes.

La dispersion totale peut s'exprimer de deux manières équivalentes :

$$S_T = \sum_{x \in E} (x - m)(x - m)^t = S_W + S_B. \quad (2.17)$$

Cette identité (dont la démonstration est très simple) est très importante. En effet, la première expression ne dépend que des données, et pas de la classification qui en est faite, montrant ainsi que la dispersion totale est constante. La deuxième forme dépend de la classification. Il est donc équivalent de minimiser la dispersion intra-classe ou de maximiser la dispersion inter-classes (puisque la somme des deux est constante pour un nombre de classes fixé). Ces deux optimisations expriment de manière mathématique le fait qu'on cherche à regrouper les échantillons en nuages qui soient espacés le plus possible les uns des autres, les points à l'intérieur de chaque nuage étant regroupés le plus possible.

Une fois choisi le critère de classification à optimiser, le problème consiste à trouver la répartition des échantillons en c classes qui satisfasse le mieux le critère. Une recherche exhaustive sur toutes les classifications possibles étant prohibitive, l'optimisation se fait de manière itérative, en vérifiant à chaque itération que le critère est amélioré. La démarche typique est celle qui consiste à déplacer un échantillon d'une classe vers une autre à condition que cette opération améliore le critère.

Prenons par exemple le critère de l'erreur quadratique. Si l'on change un échantillon x de la classe ω_i à la classe ω_j , les erreurs quadratiques J_i et J_j de ces deux classes deviennent :

$$J_j^* = J_j + \frac{n_j}{n_j + 1} \|x - m_j\|^2,$$

$$J_i^* = J_i - \frac{n_i}{n_i - 1} \|x - m_i\|^2.$$

Ainsi, le changement sera intéressant si le total des erreurs J_e a diminué (si l'on a plus gagné sur J_i que perdu sur J_j), c'est-à-dire si :

$$J_j^* - J_j \leq J_i - J_i^*.$$

Un algorithme possible de classification découlant de cette règle s'écrirait :

1. Choix d'une partition initiale des n échantillons et calcul de J_e , et des centres de classes m_1, \dots, m_c .
2. Soit un échantillon x de la classe ω_i .
Si $n_i = 1$ aller en 5 (le nombre de classes est fixé donc on interdit à une classe de se vider complètement).
Sinon, calcul des variations ρ_j des J_j correspondant au transfert de x vers la classe ω_j :

$$\rho_j = \begin{cases} \frac{n_j}{n_j + 1} \|x - m_j\|^2 & \text{si } j \neq i, \\ \frac{n_i}{n_i - 1} \|x - m_i\|^2 & \text{si } j = i. \end{cases}$$

3. Affecter x à ω_k si $\rho_k = \min_{j=1}^c \rho_j$.
4. Mise à jour de J_e (seulement deux termes ont changé, de quantités calculées à l'étape 2, donc le calcul est très rapide), de m_i et m_k .
5. Si J_e n'a pas changé en n essais (examen de tous les échantillons), fin.
Sinon, retour à l'étape 2.

Cet algorithme a l'avantage d'être optimal à chaque étape. En revanche, l'optimalité globale n'est pas garantie. Le fait de changer un seul point à la fois (et pas forcément celui qui améliore le mieux le critère) rend l'algorithme sensible aux minima locaux. De plus, l'ordre d'examen des points a alors beaucoup d'importance (la convergence peut se faire vers un minimum local différent si on change cet ordre).

2.4.4 K-moyennes

Un des algorithmes les plus utilisés, découlant directement de l'approche précédente est celui des k-moyennes (« k-means »). Il optimise le critère de l'erreur quadratique de manière itérative par les étapes suivantes (le nombre de classes est ici traditionnellement noté k , d'où le nom de k-moyennes).

1. **Initialisation** : Choix de centres initiaux $m_j(1)$ arbitraires (équidistribués, tirés au hasard, ou encore k échantillons choisis au hasard parmi les n).
2. **Affectation** : À l'itération i , x est affecté à ω_j si :

$$\|x - m_j(i)\| = \min_{l=1}^c \|x - m_l(i)\|. \quad (2.18)$$

Tous les échantillons sont classés selon cette règle (du centre le plus proche).

3. **Mise à jour des centres** : Calcul des nouveaux centres $m_j(i+1)$ pour minimiser l'erreur quadratique :

$$J_j = \sum_{x \in \omega_j} \|x - m_j(i+1)\|^2$$

En annulant la dérivée de cette expression par rapport à m_j , on obtient :

$$\frac{\partial J_j}{\partial m_j} = -2 \sum_{x \in \omega_j} (x - m_j) = 0,$$

d'où la valeur optimale de m_j pour l'itération $(i+1)$:

$$m_j(i+1) = \frac{1}{n_j} \sum_{x \in \omega_j} x. \quad (2.19)$$

4. **Test de convergence** : Si $\forall j, m_j(i+1) = m_j(i)$, fin.
Sinon, retour à l'étape 2.

Il n'existe pas pour cet algorithme de preuve générale de convergence vers l'optimum global. Les expériences montrent que le choix des centres initiaux a une grande influence sur l'optimum trouvé. Cette fois, contrairement à l'algorithme présenté dans la partie 2.4.3, la mise à jour n'est effectuée qu'une fois tous les points examinés.

Le comportement de l'algorithme est influencé, outre par le choix des centres initiaux, par le nombre de classes et par les propriétés géométriques des données. La figure 2.2 illustre ces influences : une classification en deux classes de l'ensemble de points (représentés dans un espace à deux dimensions) conduit à des résultats différents suivant l'initialisation. En revanche, une classification en trois classes (intuitivement plus naturelle d'après la géométrie des données) ne pose pas de problèmes.

La version présentée ici peut être rendue plus générale en remplaçant la norme euclidienne par une norme de type Mahalanobis par exemple, le critère à minimiser devenant alors :

$$\min \sum_{j=1}^k \sum_{x \in \omega_j} (x - m_j)^t S^{-1} (x - m_j),$$

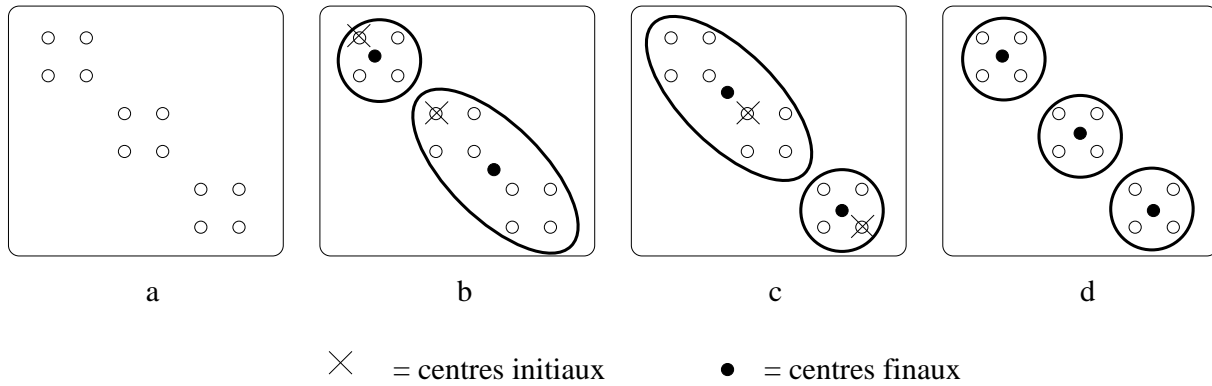


FIG. 2.2 – Influence des centres initiaux, du nombre de classes et de la géométrie des données dans l’algorithme des k-moyennes. a : ensemble de points à classifier, b et c : deux classifications en 2 classes, avec des centres initiaux différents, d : classification en trois classes.

où S est la matrice de covariance du nuage de points.

Notons que dans le cas des images, si la classification s’effectue sur le niveau de gris (les m_j sont alors les niveaux de gris moyens des classes), l’algorithme peut être exécuté de manière très rapide en travaillant uniquement sur l’histogramme de l’image. Cela permet de n’examiner qu’un nombre réduit d’échantillons : le nombre de niveaux de gris, plutôt que le nombre de points dans l’image. La méthode est souvent utilisée sous cette forme, comme initialisation de méthodes plus sophistiquées, prenant en compte plus de caractéristiques que le seul niveau de gris.

2.4.5 Algorithme ISODATA

Optimisant toujours le même critère, l’algorithme ISODATA est une version plus sophistiquée que celui des k-moyennes : il permet de faire varier le nombre de classes en cours d’algorithme par regroupement ou division, ces opérations étant effectuées alternativement, une itération sur deux. Les critères de regroupement et division dépendent eux des dispersions intra-classe et inter-classes. La contre-partie est l’introduction d’un nombre important de variables de contrôle de l’algorithme.

Les itérations suivent les étapes suivantes.

1. Définition des paramètres suivants :
 - nombre de classes voulues c ,
 - nombre minimum d’échantillons par classe θ_N ,
 - seuil de division des classes θ_S ,
 - seuil de regroupement des classes θ_C ,
 - nombre maximal de classes pouvant être regroupées L ,
 - nombre maximal d’itérations I .

Initialisation de n_c centres de classes m_1, \dots, m_{n_c} (éventuellement, n_c peut être différent du nombre de classes voulues).

2. Classification des échantillons selon la même règle que pour les k-moyennes (équation 2.18).

- Suppression des classes de moins de θ_N éléments. Les échantillons correspondants sont reclassés selon la règle 2.18 parmi les classes restantes.

Mise à jour de n_c .

- Calcul des nouveaux centres, comme moyenne de chaque classe (selon l'équation 2.19).
- Calcul de la distance moyenne des échantillons ω_j au centre :

$$\bar{D}_j = \frac{1}{n_j} \sum_{x \in \omega_j} \|x - m_j\|.$$

- Calcul de

$$\bar{D} = \frac{1}{n} \sum_{j=1}^{n_c} n_j \bar{D}_j.$$

- Si $n_c \leq c/2$, aller à l'étape 8.

Si l'itération courante est paire ou si $n_c \geq c/2$, aller à l'étape 11 (regroupement).

Sinon, aller à l'étape 8 (division).

- En appelant d la dimension de l'espace E des échantillons, calcul de

$$\sigma_j = (\sigma_{1j}, \dots, \sigma_{dj}),$$

avec :

$$\sigma_{ij} = \sqrt{\frac{1}{n_j} \sum_{x_k \in \omega_j} (x_{ik} - m_{ij})^2},$$

pour $i = 1, \dots, d$ et $j = 1, \dots, n_c$.

- Calcul de la direction dans laquelle la variance est maximale :

$$\forall j = 1, \dots, n_c, \quad \sigma_{mj} = \max_{i=1}^d \sigma_{ij}.$$

- Si $\sigma_{mj} > \theta_S$ et :

$$\begin{cases} \bar{D}_j > \bar{D} \text{ et } n_j > 2(\theta_N + 1) \\ \text{ou } n_c \leq c/2 \end{cases}$$

alors, division de ω_j en deux classes de centres m_j^+ et m_j^- tels que :

$$\begin{aligned} m_{mj}^+ &= m_{mj} + \lambda \sigma_{mj} \\ m_{mj}^- &= m_{mj} - \lambda \sigma_{mj} \\ m_{ij}^+ &= m_{ij}^- = m_{ij} \text{ pour } i \neq m, \end{aligned}$$

avec $0 < \lambda \leq 1$.

Si la séparation a lieu, aller à l'étape 2.

Sinon, aller à l'étape 11.

- Calcul, pour tous i et j tels que $i < j$, de l'écart entre les centres des classes ω_i et ω_j :

$$D_{ij} = \|m_i - m_j\|.$$

- Comparaison des D_{ij} à θ_C pour sélectionner les classes à regrouper.
- Regroupement, par paires, des classes telles que $D_{ij} \leq \theta_C$ en commençant par les classes les plus proches l'une de l'autre. Limiter le nombre de regroupements à L .
- Si la dernière itération est atteinte, fin.
Sinon, aller à l'étape 1 si l'utilisateur veut modifier des paramètres, à l'étape 2 sinon.
Incrémenter le nombre d'itérations.

2.4.6 Boules optimisées

L'algorithme des boules optimisées est proche de celui des k-moyennes. Le critère de classification est identique (centre de classe le plus proche). La différence réside dans l'introduction d'une contrainte supplémentaire sur la forme des classes : chaque classe doit être incluse dans une boule de rayon R fixé. Le nombre de classes n'est en revanche pas toujours imposé.

Voici une version de l'algorithme où seul le rayon maximal R est imposé.

1. Initialisation du nombre de classes $n_c = 1$ et du centre m_1 (on prend en général un des échantillons).
2. L'affectation d'un échantillon x à la classe ω_i de centre m_i suit la règle 2.18, augmentée d'une contrainte sur la distance au centre :

$$\begin{cases} d(x, m_i) &= \min_{j=1}^{n_c} d(x, m_j), \\ d(x, m_i) &\leq R, \end{cases}$$

où d est une distance sur l'espace E des échantillons (le plus souvent, la distance euclidienne).

Si la condition sur le rayon n'est pas satisfaite, on crée une nouvelle classe ω_{n_c+1} de centre $m_{n_c+1} = x$.

3. Mise à jour des centres des classes, selon l'équation 2.19.
4. Test de convergence : si la partition ne change plus, fin.
Sinon, aller à l'étape 2.

Cet algorithme est donc équivalent à celui des k-moyennes « avec rejet », c'est-à-dire où la décision d'appartenance à une classe est rejetée si le point est trop loin du centre de la classe, ce point donnant alors lieu à la création d'une nouvelle classe.

Des variantes de l'algorithme imposent un nombre maximum d'itérations, ou encore un nombre maximum de classes, les points ne répondant pas aux contraintes étant alors rejetés (sans reclassification).

2.4.7 Nuées dynamiques

La méthode des nuées dynamiques généralise les algorithmes précédents en remplaçant la mesure de distance entre un échantillon x et le centre m_i d'une classe ω_i par une mesure plus générale de « dissemblance » entre l'échantillon et la classe, notée dans la suite $f(x, \omega_i)$.

Le nombre de classes c étant fixé, une partition $P = \{\omega_1, \dots, \omega_c\}$ est la meilleure classification de l'ensemble des échantillons en c classes si le critère suivant est minimal :

$$\sum_{i=1}^c \sum_{x \in \omega_i} f(x, \omega_i), \quad (2.20)$$

qui est bien une généralisation des critères vus dans la partie 2.4.3, en particulier de celui utilisé dans les k-moyennes.

Ici encore, le minimum global est souvent difficile à trouver et les algorithmes itératifs utilisés recherchent des minima locaux. Ces algorithmes suivent exactement le même schéma que celui des k-moyennes où seule la règle d'affectation 2.18 est remplacée par :

$$x \in \omega_i(k+1) \Leftrightarrow f(x, \omega_i(k)) = \min_{j=1}^c f(x, \omega_j(k)), \quad (2.21)$$

où $\omega_i(k)$ désigne la classe i à l'itération k . Nous ne détaillerons donc pas plus ces algorithmes. Notons que leur convergence (vers un minimum local) n'est démontrable que pour certaines fonctions f .

Toute la difficulté de la méthode réside dans le choix d'une fonction de dissemblance f adaptée au problème. Nous décrivons ici la méthode de Diday. Elle consiste à représenter une classe par un « noyau », généralisant la notion de centre utilisée dans les parties précédentes. La fonction f est alors une mesure de dissemblance entre l'échantillon et le noyau de la classe. Donnons quelques exemples.

1. Le cas le plus simple est celui où le noyau est simplement le centre de gravité de la classe, donc exactement le centre utilisé précédemment. La fonction de dissemblance est alors la distance entre l'échantillon et le centre. Ce cas particulier des nuées dynamiques coïncide donc exactement avec les k-moyennes, et on retrouve les mêmes propriétés de convergence vers un minimum local.
2. Le noyau d'une classe ω peut être constitué de plusieurs points (y_1, \dots, y_p) au lieu d'un seul. Ils sont par exemple choisis de telle sorte que la fonction suivante soit minimale :

$$\sum_{x \in \omega} \inf_{k=1}^p d(x, y_k). \quad (2.22)$$

La fonction de dissemblance est alors la moyenne des distances de x à chacun des points du noyau de la classe :

$$f(x, \omega) = \frac{1}{p} \sum_{k=1}^p d(x, y_k). \quad (2.23)$$

Avec un seul centre, on obtient toujours des classes compactes où les points sont regroupés autour du centre. En revanche, ici une classe est représentée par plusieurs points, et la méthode est donc mieux adaptée aux formes complexes. On peut montrer la convergence de l'algorithme vers un minimum local.

3. Au lieu de prendre des points pour constituer le noyau, on peut prendre des objets géométriques plus compliqués. Par exemple, pour la reconnaissance de caractères majuscules, une bonne représentation peut être obtenue en prenant pour noyau d'une classe son axe principal d'inertie Δ_ω . La fonction de dissemblance s'exprime alors sous la forme :

$$f(x, \omega) = d(x, \Delta_\omega) = \inf_{y \in \Delta_\omega} d(x, y). \quad (2.24)$$

2.4.8 Limites

Si les méthodes présentées dans cette partie ont l'avantage d'être non supervisées, simples, rapides, elles souffrent d'un certain nombre d'inconvénients qui sont rappelés succinctement ici.

Elles nécessitent d'abord une connaissance sur les données et la classification que l'on souhaite en obtenir. Cette connaissance est, pour la plupart des méthodes, le nombre de classes. S'il est raisonnable dans beaucoup d'applications de supposer que ce nombre est connu, cette hypothèse ne permet pas en revanche de traiter de manière satisfaisante et automatique des problèmes pour lesquels des échantillons pathologiques sont susceptibles d'apparaître, et qui formeraient des classes supplémentaires.

Les méthodes n'utilisant pas le nombre de classes connu a priori remplacent cette connaissance par une autre, par exemple sur la géométrie des données. C'est le cas du rayon dans la méthode des boules optimisées. C'est aussi celui des paramètres de contrôle dans l'algorithme ISODATA, nécessitant d'avoir une idée des dispersions inter-classes et intra-classe.

Tous les algorithmes présentés sont itératifs et nécessitent une initialisation. Le problème, classique en optimisation, des minima locaux se pose ici de manière cruciale. Plusieurs solutions sont envisageables :

- on peut trouver facilement une initialisation proche de l'optimum (c'est le cas par exemple de la classification d'une image scanner à partir des niveaux de gris, où l'on connaît a priori le niveau de gris des os, celui des tissus mous, etc.) ;
- on exécute l'algorithme à partir de plusieurs positions de départ et on choisit la meilleure solution ;
- on exécute l'algorithme un grand nombre de fois, pour des initialisations tirées au hasard, et on recherche les solutions stables.

L'espace E dans lequel sont définis les échantillons est un espace de caractéristiques qui peuvent être de natures et de dimensions très différentes. Si l'on cherche par exemple à classer un certain nombre d'individus et que les caractéristiques retenues sont la taille de l'appartement qu'ils occupent, leur consommation mensuelle de lait et leur nombre d'enfants, l'espace E , tridimensionnel ici, a des axes qui représentent des variables très hétérogènes. Il est alors délicat de définir une métrique sur un tel espace. La solution classiquement adoptée consiste à utiliser une distance de Mahalanobis, permettant de tenir compte des échelles très différentes sur chacun des axes. Cependant, l'idée même de construire une métrique dans de telles conditions est critiquable. La solution qui consisterait à traiter chaque dimension séparément puis à les combiner fait appel à des notions de fusion de données qui sortent du cadre de ce cours.

2.5 Méthodes hiérarchiques

Les méthodes hiérarchiques consistent à effectuer plusieurs classifications emboîtées les unes dans les autres, permettant de choisir ensuite celle qui correspond au niveau de finesse désiré. L'application la plus évidente en est la taxinomie arborescente.

Ces méthodes reposent sur l'équivalence entre une hiérarchie indicée sur E (espace des échantillons) et une distance ultra-métrique sur E . Les parties suivantes sont consacrées à l'introduction de ces concepts, conduisant à la démonstration de l'équivalence.

2.5.1 Distance ultra-métrique

Une distance ultra-métrique sur E est une application δ de $E \times E$ dans \mathbb{R}^+ telle que :

$$\forall (x_1, x_2) \in E^2, \quad \delta(x_1, x_2) = 0 \Leftrightarrow x_1 = x_2, \quad (2.25)$$

$$\forall (x_1, x_2) \in E^2, \quad \delta(x_1, x_2) = \delta(x_2, x_1), \quad (2.26)$$

$$\forall (x_1, x_2, x_3) \in E^3, \quad \delta(x_1, x_2) \leq \sup[\delta(x_1, x_3), \delta(x_3, x_2)]. \quad (2.27)$$

Les conditions 2.25 et 2.26 sont équivalentes aux conditions de séparation et de symétrie respectivement d'une distance classique. En revanche, la troisième condition (équation 2.27) est plus forte que l'inégalité triangulaire classique qu'elle entraîne. Ainsi, toute ultra-métrie est une distance.

Une des propriétés des ultra-métries est que tout triangle est isocèle (la démonstration est immédiate à partir de l'équation 2.27 exprimée pour toutes les arêtes du triangle).

2.5.2 Distance de chaîne

La distance de chaîne permet de définir une distance entre échantillons en termes de chemin. Elle correspond à l'idée intuitive de distance entre deux cailloux dans une rivière, où l'on cherche à aller d'un cailloux à l'autre (sans mettre le pied dans l'eau) en minimisant le pas le plus grand que l'on doit faire entre deux cailloux successifs. De manière mathématique, quelques définitions sont nécessaires.

Soit une distance d sur E (toujours l'ensemble des échantillons). On appelle chemin de x_1 à x_2 la suite d'échantillons $c = (c_1, c_2, \dots, c_n)$ avec :

$$\forall i, \quad c_i \in E,$$

$$c_1 = x_1,$$

$$c_n = x_2.$$

Le pas du chemin est défini comme la quantité :

$$P(c) = \sup_{i=1}^{n-1} d(c_i, c_{i+1}). \quad (2.28)$$

La distance de chaîne est alors définie sur $E \times E$ par :

$$\delta(x_1, x_2) = \inf_{c \in CH(x_1, x_2)} P(c), \quad (2.29)$$

où $CH(x_1, x_2)$ représente l'ensemble de tous les chemins de x_1 à x_2 dans E .

On a alors le résultat important suivant : la distance de chaîne δ est une ultra-métrie.

La démonstration des conditions 2.25 et 2.26 est immédiate. Pour la condition 2.27, on a :

$$\begin{aligned} \delta(x_1, x_2) &= \inf_{c \in CH(x_1, x_2)} P(c) \\ &\leq \inf_{c \text{ passant par } x_3} P(c) \\ &\leq \inf_{c' \in CH(x_1, x_3)} \inf_{c'' \in CH(x_3, x_2)} \sup[P(c'), P(c'')] \\ &\leq \inf_{c' \in CH(x_1, x_3)} [\sup[P(c'), \inf_{c'' \in CH(x_3, x_2)} P(c'')]] \\ &\leq \sup[\inf_{c' \in CH(x_1, x_3)} P(c'), \inf_{c'' \in CH(x_3, x_2)} P(c'')] \\ &\leq \sup[\delta(x_1, x_3), \delta(x_3, x_2)]. \end{aligned}$$

Définissons un ordre sur les distances de la manière suivante : pour deux distances d_1 et d_2 sur E ,

$$d_1 \leq d_2 \Leftrightarrow \forall (x, y) \in E^2, \quad d_1(x, y) \leq d_2(x, y). \quad (2.30)$$

On peut montrer, pour cet ordre, que pour toute ultra-métrie Δ , on a la relation suivante avec la distance initiale sur E et la distance de chaîne δ (on a $\delta \leq d$) :

$$\Delta \leq d \Rightarrow \Delta \leq \delta. \quad (2.31)$$

Cette propriété exprime que la distance de chaîne est l'ultra-métrie « sous-dominante », c'est-à-dire qu'il n'existe pas d'ultra-métrie inférieure à d plus proche de d que la distance de chaîne. Notons qu'il n'existe pas de notion équivalente d'ultra-métrie sur-dominante. C'est en grande partie pour cette propriété, et bien sûr pour son interprétation concrète (voir l'exemple des cailloux), que la distance de chaîne est l'ultra-métrie la plus utilisée.

Pour la démonstration de cette propriété, considérons le chemin c sur lequel est atteint la distance de chaîne (minimum dans l'équation 2.29) :

$$\delta(x, y) = P(c) = \sup_{i=1}^{n-1} d(c_i, c_{i+1}),$$

avec $c = (c_1, \dots, c_n)$, $c_1 = x$, $c_n = y$.

On a alors pour une ultra-métrie Δ quelconque :

$$\begin{aligned} \Delta \leq d &\Rightarrow \forall i, \Delta(c_i, c_{i+1}) \leq d(c_i, c_{i+1}) \\ &\Rightarrow \sup_{i=1}^{n-1} \Delta(c_i, c_{i+1}) \leq \sup_{i=1}^{n-1} d(c_i, c_{i+1}) = \delta(x, y). \end{aligned}$$

Comme Δ est une ultra-métrie, on a :

$$\Delta(x, y) = \Delta(c_1, c_n) \leq \sup_{i=1}^{n-1} \Delta(c_i, c_{i+1}),$$

et donc $\Delta(x, y) \leq \delta(x, y)$, ce qui complète la démonstration.

2.5.3 Relation d'équivalence et partition à partir d'une ultra-métrie

Soit δ une ultra-métrie. La relation

$$\forall \delta_0 \in \mathbb{R}^+, x R_{\delta_0} y \Leftrightarrow \delta(x, y) \leq \delta_0 \quad (2.32)$$

définit une relation d'équivalence (la démonstration est immédiate à partir de la définition d'une ultra-métrie). La partition formée des classes d'équivalence de R_{δ_0} est notée $P(\delta_0)$.

Il découle de cette propriété l'application à la classification d'un ensemble d'échantillons à partir de la distance de chaîne et d'un seuil δ_0 ; $P(\delta_0)$ est alors simplement l'ensemble des classes obtenues. Une telle classification a les propriétés suivantes :

1. on peut passer d'un point à un autre d'une même classe, en restant dans la classe et en ne faisant jamais de pas de longueur supérieure à δ_0 ; cette propriété est proche de la notion de dispersion intra-classe présentée dans la partie 2.4.3 (la différence essentielle est que l'on peut avoir ici des classes très allongées à condition qu'il y ait des échantillons tout le long de la classe) ;
2. pour passer d'un point d'une classe à un point d'une autre classe, il faut faire au moins un saut de longueur supérieure à δ_0 ; cette propriété est donc proche de la notion de dispersion inter-classes.

2.5.4 Hiérarchie et hiérarchie indicée

On se place ici dans le cas où E est fini, et on note $\mathcal{P}(E)$ l'ensemble des parties de E . On appelle hiérarchie sur E un sous-ensemble \mathcal{H} de $\mathcal{P}(E)$ qui vérifie les propriétés suivantes :

$$E \in \mathcal{H}, \quad (2.33)$$

$$\forall x \in E, \quad \{x\} \in \mathcal{H}, \quad (2.34)$$

$$\forall (h, h') \in \mathcal{H}^2, \quad \begin{cases} h \cap h' = \emptyset, \\ \text{ou } h \subset h', \\ \text{ou } h' \subset h. \end{cases} \quad (2.35)$$

La partition introduite dans la partie 2.5.3 à partir d'une ultra-métrie induit une hiérarchie sur E : $\mathcal{H} = \bigcup_{\delta_0 \in \mathbb{R}^+} P(\delta_0)$ est une hiérarchie.

La démonstration de la propriété 2.33 découle du fait que E est fini et donc la plus grande distance entre deux éléments existe. La partition associée à cette distance est réduite à la classe E . La propriété 2.34 est vérifiée : $\{x\} = P(0)$. Pour 2.35, considérons h et h' deux éléments de \mathcal{H} . Alors :

$$\exists (\delta_0, \delta'_0), \text{ tel que } h \in P(\delta_0), h' \in P(\delta'_0).$$

Si $h \cap h' = \emptyset$, la propriété est vérifiée. Sinon, soit $x \in h \cap h'$. h est la classe de x dans la partition $P(\delta_0)$ et h' celle de x dans la partition $P(\delta'_0)$. h et h' peuvent donc s'écrire :

$$h = \{y / \delta(x, y) \leq \delta_0\},$$

$$h' = \{y / \delta(x, y) \leq \delta'_0\}.$$

Par conséquent, si $\delta_0 \leq \delta'_0$, alors $h \subset h'$. Sinon, $h' \subset h$, ce qui achève la démonstration.

Introduisons maintenant la notion de hiérarchie indicée : c'est un couple (\mathcal{H}, f) tel que \mathcal{H} est une hiérarchie et f est une application de \mathcal{H} dans \mathbb{R}^+ telle que :

$$\forall x \in E, \quad f(\{x\}) = 0, \quad (2.36)$$

$$\forall (h, h') \in \mathcal{H}^2, h \subset h', h \neq h' \Rightarrow f(h) < f(h'). \quad (2.37)$$

2.5.5 Équivalence entre hiérarchie indicée et ultra-métrie

Reprenons l'exemple de la hiérarchie $\mathcal{H} = \bigcup_{\delta_0 \in \mathbb{R}^+} P(\delta_0)$. Le couple (\mathcal{H}, f) est une hiérarchie indicée pour f définie par :

$$\forall h \in \mathcal{H}, \quad f(h) = \min\{\delta_0 / h \in P(\delta_0)\}. \quad (2.38)$$

Comme, pour tout échantillon x , le singleton $\{x\}$ appartient à $P(0)$, on a $f(\{x\}) = 0$, et la propriété 2.36 est vérifiée. Soit h et h' dans \mathcal{H} , tels que $h \subset h'$ et $h \neq h'$, et soit x dans h . On a alors :

$$\{\delta'_0 / h' \in P(\delta'_0)\} \subset \{\delta'_0 / h \in P(\delta'_0)\},$$

et donc :

$$\min\{\delta'_0 / h' \in P(\delta'_0)\} \geq \min\{\delta'_0 / h \in P(\delta'_0)\},$$

donc $f(h) \leq f(h')$. Comme de plus, $h \neq h'$, il existe un y dans h' qui ne soit pas dans h . Soit x dans $h \cap h'$. On a :

$$f(h) < \delta(x, y) \leq f(h')$$

et donc $f(h) < f(h')$ (avec une inégalité stricte), ce qui démontre la propriété 2.37.

Réciproquement, soit (\mathcal{H}, f) une hiérarchie indicée, et soit δ l'application de $E \times E$ dans \mathbb{R}^+ telle que :

$$\delta(x, y) = \min_{h \in \mathcal{H}} \{f(h) \mid (x, y) \in h^2\}.$$

Alors δ est une ultra-métrie.

Tout d'abord, remarquons que δ est toujours définie puisque E est fini.

Démontrons la propriété 2.25. $\delta(x, x) = 0$ car $f(\{x\}) = 0$ par définition (propriété 2.34). Si $\delta(x, y) = 0$, alors :

$$\exists h \in \mathcal{H} \text{ tel que } x \in h, y \in h \text{ et } f(h) = 0.$$

Si $h \neq (\{x\})$, $f(h) > f(\{x\}) = 0$, ce qui est impossible. Donc $x = y$.

La démonstration de la propriété 2.26 est triviale.

Soient h tel que $\delta(x, y) = f(h)$, h' tel que $\delta(x, z) = f(h')$, h'' tel que $\delta(y, z) = f(h'')$. Puisque $z \in h' \cap h''$, $h' \cap h'' \neq \emptyset$. Donc $h' \subset h''$ puisque \mathcal{H} est une hiérarchie (propriété 2.35), ou le contraire mais le raisonnement serait analogue. Donc x , qui est dans h' , appartient à h'' . y appartient également à h'' . Or :

$$f(h) = \min\{f(h) \mid (x, y) \in h^2\}.$$

Donc $h \subset h''$. D'où :

$$f(h) \leq f(h'') \leq \sup[f(h'), f(h'')],$$

ce qui montre la propriété 2.27.

La figure 2.3 illustre l'équivalence entre hiérarchie indicée et ultra-métrie : l'emboîtement des classes d'équivalence quand δ_0 varie correspond aux différents étages de la hiérarchie. Pour aller d'un échantillon x à un autre y dans la hiérarchie, il faut « remonter » jusqu'à un niveau minimum qui est exactement $\delta(x, y)$.

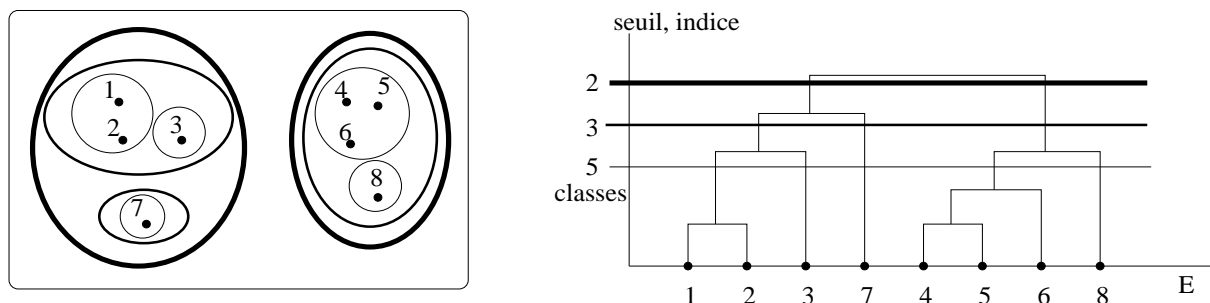


FIG. 2.3 – Équivalence entre ultra-métrie (à gauche, les partitions induites pour différentes valeurs de δ_0 , en 2, 3 et 5 classes respectivement) et hiérarchie indicée (à droite).

2.5.6 Arbre de longueur minimum

Rappelons qu'un arbre est un graphe connexe sans circuit. À partir d'un ensemble de n échantillons, on peut donc construire des arbres dont les nœuds sont les échantillons et ces arbres ont $n - 1$ arcs. La longueur d'un arc étant définie comme la distance entre les deux nœuds extrémités, on appelle arbre de longueur minimum un arbre dont la longueur des arcs est minimum.

L'analogie entre arbre de longueur minimum et hiérarchie indicée (et ultra-métrique) est la suivante : partant d'un arbre de longueur minimum construit à partir des n échantillons à classer, on coupe les arcs dont la longueur est supérieure à un certain seuil (δ_0) ; les composantes connexes obtenues sont exactement les classes de $P(\delta_0)$, ou encore celles obtenues à l'indice δ_0 de la hiérarchie.

2.5.7 Limites

Parmi les limites des méthodes hiérarchiques, on retrouve le problème de la définition de la métrique de base qu'il faut définir sur E (voir partie 2.4.8). Les méthodes sont très sensibles au choix de cette distance.

Ces méthodes sont également très sensibles aux positions relatives des échantillons. En effet, la distance de chaîne conduit à des résultats très différents si un seul point est mal placé : il peut regrouper deux classes ou au contraire en diviser. Cette grande sensibilité se trouve également sous le nom d'instabilité structurelle.

Chapitre 3

Approches structurelles

Les approches structurelles reposent sur des représentation de formes quelconques sous forme d'agencement, d'assemblage de formes plus simples, les primitives, considérées, comme motifs élémentaires. Les structures les plus simples sont les représentations par chaînes, et nous verrons dans un premier temps comment les comparer pour effectuer la reconnaissance. Les chaînes sont par exemple utilisées pour comparer et reconnaître des formes à partir de leur squelette (voir cours de morphologie mathématique) ou de leurs contours. Les grammaires et les automates permettent une analyse structurée, imitant celle du langage naturel. Des modèles plus généraux reposent sur des graphes (avec le cas particulier des arbres), et sont souvent utilisés pour la mise en correspondance et l'analyse de scènes complexes, dans lesquelles l'agencement spatial des structures est de grande importance.

Des descriptions plus complètes de ces approches peuvent être trouvées dans [11, 18, 14].

3.1 Comparaison de chaînes

3.1.1 Définitions

Une chaîne est une suite ordonnée (une phrase) d'éléments (des lettres) d'un alphabet fini.

Soit X l'alphabet, et a_1, a_2, \dots, a_n des éléments de X (quelconques, pas forcément tous différents). Une chaîne s'écrira alors

$$a = a_1 a_2 \dots a_n$$

et on dira que la chaîne est de longueur $|a| = n$.

L'opération la plus simple entre deux chaînes $a = a_1 a_2 \dots a_m$ et $b = b_1 b_2 \dots b_m$, définies sur le même alphabet, est la concaténation, définie par :

$$a * b = a_1 a_2 \dots a_n b_1 b_2 \dots b_m$$

de longueur $|a * b| = |a| + |b| = n + m$.

Cette opération est associative, non commutative, et elle admet comme élément neutre la chaîne vide, notée λ (de longueur nulle).

On appelle sous-chaîne d'une chaîne a une chaîne b telle qu'il existe deux chaînes c et d (éventuellement vides) telle que a puisse s'écrire sous la forme $a = c * b * d$. Les chaînes c et d s'appellent respectivement préfixe et suffixe de a . Toutes ces chaînes doivent bien sûr être définies sur le même alphabet.

Le principe de la reconnaissance des formes à partir de structures de chaînes consiste à comparer une forme à reconnaître à un prototype, les deux étant représentés dans le même alphabet.

3.1.2 Distance de Hamming

Lorsque les deux chaînes sont de même longueur, une manière simple de les comparer est d'évaluer leur distance de Hamming, consistant à compter les éléments différents dans les deux chaînes.

Ainsi, la distance de Hamming entre $a = a_1 a_2 \dots a_n$ et $b = b_1 b_2 \dots b_n$ est définie par :

$$d_H(a, b) = \sum_{i=1}^n \delta(a_i, b_i)$$

où

$$\delta(a_i, b_i) = \begin{cases} 1 & \text{si } a_i \neq b_i \\ 0 & \text{si } a_i = b_i \end{cases}$$

3.1.3 Inclusion d'une chaîne dans une autre

Etablir si une chaîne est incluse dans une autre permet d'extraire une partie d'une forme correspondant à un modèle ou à un prototype. Par exemple la chaîne $b = b_1 b_2 b_3$ est incluse dans la chaîne $a = a_1 a_2 b_1 b_2 b_3 a_3$.

L'algorithme le plus classique pour établir cette inclusion est celui de Morris et Pratt [1], de complexité linéaire, et dont le principe consiste à déplacer un pointeur dont la valeur indique la longueur du plus long préfixe de b reconnu comme une sous-chaîne de a .

3.1.4 Distance d'édition de chaîne

La comparaison de chaînes de longueurs différentes se fait en construisant une distance appelée distance d'édition de chaîne, exprimée comme le coût minimum d'une suite d'opérations élémentaires. On distingue trois types d'opérations :

- insertion d'un élément a_i , de coût noté $\gamma(\lambda, a_i)$,
- suppression d'un élément a_i , de coût $\gamma(a_i, \lambda)$,
- substitution de a_i par b_j , de coût $\gamma(a_i, b_j)$.

Cet distance se calcule par l'algorithme de Wagner et Fisher [26], de programmation dynamique. Il consiste à calculer, pour deux chaînes $a = a_1 a_2 \dots a_n$ et $b = b_1 b_2 \dots b_m$, la matrice $\delta(i, j)$ des coûts de transformations partielles, l'optimum (donc la distance) étant finalement donné par la valeur de $\delta(n, m)$:

- $\delta(0, 0) = 0$
- de $i = 1 \dots n$: $\delta(i, 0) = \delta(i - 1, 0) + \gamma(a_i, \lambda)$
- de $j = 1 \dots m$: $\delta(0, j) = \delta(0, j - 1) + \gamma(\lambda, b_j)$

- de $i = 1 \dots n$ et de $j = 1 \dots m$:
 - $m_1 = \delta(i - 1, j - 1) + \gamma(a_i, b_j)$
 - $m_2 = \delta(i - 1, j) + \gamma(a_i, \lambda)$
 - $m_3 = \delta(i, j - 1) + \gamma(\lambda, b_j)$
 - $\delta(i, j) = \min(m_1, m_2, m_3)$
- $d(a, b) = \delta(n, m)$

Calculons à titre d'exemple la distance entre les deux chaînes LOUVAIN et LEUVEN (deux villes de Belgique), définies sur l'alphabet usuel. Choisissons les coûts élémentaires suivants :

$$\gamma(\lambda, a_i) = \gamma(a_i, \lambda) = \gamma(a_i, b_j) = 1.$$

Le calcul des $\delta(i, j)$ par l'algorithme de Wagner et Fisher donne alors le résultat suivant :

	λ	L	O	U	V	A	I	N
λ	0	1	2	3	4	5	6	7
L	1	0	1	2	3	4	5	6
E	2	1	1	2	3	4	5	6
U	3	2	2	1	2	3	4	5
V	4	3	3	2	1	2	3	4
E	5	4	4	3	2	2	3	4
N	6	5	5	4	3	3	3	3

Supposons maintenant que l'on ait une forme à reconnaître parmi un ensemble de modèles possibles. La méthode comporte alors les étapes suivantes :

- extraire les primitives de la forme à reconnaître et des modèles (les contours par exemple) ;
- représenter ces primitives sous forme de chaînes dans un même alphabet ;
- calculer la distance d'édition de chaîne entre la forme à reconnaître et chacun de modèles ;
- choisir le modèle ayant donné la plus petite distance (éventuellement rejeter cette décision avec les règles de rejet habituelles).

3.2 Grammaires et automates

Les grammaires formelles (introduites par Chomsky dans les années 1950) permettent de construire des phrases sur un alphabet. Les phrases construites par une grammaire ont une structure commune, qui définit donc une classe à laquelle on peut attribuer des formes représentées par des phrases ayant cette structure. Ce point de vue sur les grammaires explique leur utilisation en reconnaissance des formes.

3.2.1 Définitions

Une grammaire \mathcal{G} est définie par le quadruplet $\mathcal{G} = (X, V, S, P)$ où

- X est l'alphabet avec lequel sont écrites les phrases, l'ensemble des phrases étant noté X^* ;
- V est un alphabet auxiliaire ou non terminal ;
- S est un élément de V appelé axiome ;

- P est un ensemble de règles de production ou règles d'écriture.

Différents types de grammaires peuvent être distingués selon leur complexité. Les plus simples sont les grammaires régulières, pour lesquelles les règles de production ont la forme $A \rightarrow aB$, avec A et B dans V et $a \in X$, ou $A \rightarrow a$.

3.2.2 Automates finis

Les grammaires régulières peuvent être décrites de manière équivalente par des automates finis.

Un automate fini est un quintuplet $\mathcal{A} = (X, A, \delta, q_0, Q')$ où X est un alphabet, Q un ensemble fini d'états, $q_0 \in Q$ l'état initial, $Q' \subset Q$ l'ensemble des états finaux, $\delta : Q \times C^X \rightarrow Q$ une fonction de transition. Ces éléments sont souvent représentés graphiquement comme indiqué sur la figure 3.1.



FIG. 3.1 – Représentation graphique de l'état initial q_0 , d'un état final $q_F \in Q'$ et d'une transition $\delta(q, a) = q'$.

Le lien entre automate fini et grammaire régulière est immédiat : l'état initial correspond à l'axiome, les alphabets sont les mêmes et les règles de production correspondent à la fonction de transition, comme illustré sur la figure 3.2.

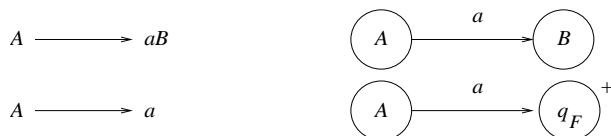


FIG. 3.2 – Correspondance entre règles de production et fonction de transition.

On peut alors montrer que la famille des langages sur un alphabet X reconnus par un automate fini est la même que celle des langages sur X engendrés par une grammaire régulière.

3.2.3 Grammaires et reconnaissance des formes

Les trois classes de problèmes typiques sont :

1. la génération, c'est-à-dire la construction du langage, donc des phrases qui peuvent être obtenues par application des toutes les règles de production possibles à l'axiome ;
2. l'analyse syntaxique, ou la reconnaissance d'une phrase comme faisant partie du langage engendré par une grammaire ;
3. l'inférence de grammaire, c'est-à-dire la détermination des règles de la grammaire à partir d'exemples de phrases du langage (ce qui est évidemment un problème particulièrement difficile).

L'utilisation des grammaires en reconnaissance des formes relève essentiellement du deuxième type de problèmes.

Lorsque la grammaire est régulière et que le langage peut donc être représenté par un automate fini, répondre à la question « la phrase $x \in X$ fait-elle partie du langage représenté par $\mathcal{A} = (X, A, \delta, q_0, Q')$? » peut se résoudre simplement en partant de l'état initial, en en traitement successivement les symboles de la phrase par l'automate jusqu'à l'arrivée à un état final. Si cet état est bien un état final au sens défini par Q' , la phrase est reconnue et la réponse est oui, sinon elle n'est pas reconnue comme faisant partie du langage. L'algorithme correspondant est le suivant, pour $x = a_1 a_2 \dots a_n$:

- $q_1 = \delta(q_0, a_1)$
- $q_2 = \delta(q_1, a_2)$
- ...
- $q_n = \delta(q_{n-1}, a_n)$
- si $q_n \in Q'$, la phrase est reconnue, sinon la réponse à la question est non.

3.2.4 Utilisation des grammaires en traitement et interprétation des images

La difficulté principale pour pouvoir utiliser les grammaires pour les images est de ce ramener à des structures de chaînes, dont essentiellement mono-dimensionnelles, alors que les images sont composées de structures bi-dimensionnelles, voire tri-dimensionnelles. Cela impose également de définir un ordre sur les structures. Ainsi les contours d'objets 2D se prêtent bien à cette modélisation, alors que des représentations par squelette ou d'objets à 3D sont plus délicates à utiliser. Dès que les primitives utilisées sont de niveau un peu plus élevé, il n'existe en général pas d'ordre naturel et il faut donc en définir un.

Il est souvent nécessaire en image de travailler sur des extensions des grammaires présentées ici, afin de tenir compte des imperfections inhérentes aux images. Ainsi on utilisera des grammaires stochastiques, dans lesquelles plusieurs règles peuvent être mises en concurrence et apparaissent avec certaines probabilités, ou des grammaires tolérantes, qui acceptent de petites erreurs par rapport aux phrases strictement engendrées par la grammaire. On a alors besoin de définir des distances entre phrases et les outils présentés plus haut pour les chaînes peuvent être utilisés (en particulier l'algorithme de Wagner et Fisher).

Des extensions de ce qui précède à la définition de grammaires d'arbres ou de graphes ont également été développées, et permettent de définir des langages avec des structures plus complexes qui ne sont plus nécessairement linéaires.

3.3 Arbres et graphes

3.3.1 Définitions

Un graphe est un couple (N, A) où N est un ensemble de nœuds et A un ensemble d'arcs, définis comme des relations binaires entre les nœuds (A peut être considéré comme un sous-ensemble de $N \times N$). Les arcs sont donc orientés.

Un graphe non orienté est un graphe pour lequel $(n_1, n_2) \in A \Rightarrow (n_2, n_1) \in A$.

Un graphe complet est un graphe tel que $A = N \times N$.

Un graphe est attribué si les nœuds et/ou les arcs sont munis d'attributs, de caractéristiques, ou d'étiquettes, pouvant être numériques ou symboliques.

Un arbre est un graphe connexe sans cycle. Le parcours d'un arbre s'effectue en fixant un ordre sur les nœuds. On distingue classiquement les parcours en profondeur d'abord et en largeur d'abord.

3.3.2 Distance entre arbres

Comme pour les chaînes, la reconnaissance d'une forme représentée sous la forme d'un arbre comme faisant partie d'une classe dont les exemples ou les prototypes sont également représentés sous forme d'arbre nécessite la définition d'une distance entre arbres.

La distance de chaîne se généralise en remplaçant les symboles des chaînes par des sous-arbres. On définit ainsi un coût de suppression, ajout d'un sous-arbre, ou changement de l'étiquette d'un nœud.

L'algorithme de Selkow [22] généralise directement l'algorithme de Wagner et Fisher vu plus haut. D'autres algorithmes sont possibles, par exemple celui de Lu [17] qui utilise des transformations élémentaires plus générales : insertion et suppression d'un nœud n'importe où dans l'arbre et changement d'étiquette d'un nœud.

3.3.3 Isomorphismes de graphes

Lorsque les formes à reconnaître et les prototypes sont représentés par des graphes, leur identification s'exprime comme la recherche d'un isomorphisme entre le graphe représentant la forme à reconnaître et un des graphes représentant les prototypes. L'isomorphisme garantit une correspondance parfaite entre les structures.

Deux graphes (N, A) et (N', A') sont dits isomorphes s'il existe une bijection $f : N \Rightarrow N'$ telle que :

$$\forall (n_1, n_2) \in N^2, (n_1, n_2) \in A \Leftrightarrow (f(n_1), f(n_2)) \in A'.$$

Les algorithmes permettant de déterminer si deux graphes sont isomorphes ou non sont le plus souvent récursifs, et de complexité exponentielle dans le cas le pire.

3.3.4 Extensions

Imposer un isomorphisme est une condition très contraignante, qui ne laisse aucune place à de petites modifications ou erreurs de la forme par rapport au prototype de la classe à laquelle elle appartient. Ces contraintes peuvent être relâchées en recherchant des isomorphismes de sous-graphes seulement, ou de simples morphismes en imposant alors une contrainte supplémentaire exprimée le plus souvent comme une fonction de similarité entre les nœuds et entre les arcs (en particulier dans le cas de graphes attribués). Ces problèmes relèvent du domaine de la mise en correspondance inexacte de graphes.

3.3.5 Graphes d'association

Une forme particulière de graphe est souvent utilisée en reconnaissance structurelle des formes : le graphe d'association qui est une représentation directe de la correspondance entre deux graphes. Les nœuds du graphe d'association représentent un appariement entre

un nœud d'un graphe (N, A) et un nœud d'un graphe (N', A') . Les arcs correspondent aux compatibilités d'association entre deux paires de nœuds. Par exemple si l'association entre $n_1 \in N$ et $n'_1 \in N'$ (représentée par le nœud (n_1, n'_1) dans le graphe d'association) est compatible avec l'association entre $n_2 \in N$ et $n'_2 \in N'$ (représentée par le nœud (n_2, n'_2) dans le graphe d'association), il y aura alors un arc $((n_1, n'_1), (n_2, n'_2))$ dans le graphe d'association. Cette compatibilité est définie en fonctions des arcs de A et de A' et des similarités entre nœuds et entre arcs.

La recherche de la meilleure correspondance entre deux graphes s'appuyant sur leur graphe d'association s'effectue en général par l'une des méthodes suivantes :

- relaxation : on cherche à maximiser la cohérence globale des appariements en calculant les probabilités des appariements en fonction de leur cohérence avec des appariements voisins dans le graphe d'association ;
- recherche de cliques maximales : une clique maximale est un sous-graphe complet de taille maximale dans le graphe d'association ; une telle clique représente donc le sous-ensemble maximal d'appariements compatibles ;
- génération et vérification d'hypothèses : une hypothèse est un sous-ensemble d'appariements dont on vérifie la cohérence et qu'on fait grandir progressivement.

Dans tous ces problèmes, la difficulté principale est bien sûr de réduire la complexité de la recherche, les méthodes exhaustives étant de complexité prohibitive.

3.3.6 Utilisations en traitement et interprétation d'images

Les graphes sont souvent utilisés en image pour représenter une scène de manière structurée. Le niveau de description dépend des primitives utilisées pour définir les nœuds du graphe.

A bas niveau, les graphes les plus simples sont ceux où les pixels sont les nœuds et les relations de voisinage élémentaires définissent les arcs (voir cours sur les représentations discrètes). Ce sont les graphes utilisés également dans la plupart des applications de restauration ou de segmentation utilisant des champs de Markov.

A un niveau plus élevé, on commence par extraire des primitives de l'image (segments, régions, etc.) qui représentent les nœuds du graphe, les arcs étant alors définis comme des relations entre ces primitives (par exemple des relations d'adjacence entre les régions).

A haut niveau, les nœuds sont des objets ou des structures de la scène (ce qui suppose un traitement avancé de l'image) et les arcs sont définis comme des relations spatiales entre ces entités.

En reconnaissance des formes, ce sont le plus souvent les deux derniers types de graphes qui sont utilisés, ainsi que les graphes d'association correspondant. Par exemple, si l'on dispose de modèles d'objets, des représentations par graphe peuvent être construites en prenant pour nœuds les différentes composantes des objets et pour arcs les relations de connexion, d'adjacence ou de position entre ces composantes (avec très souvent des attributs). Un objet à reconnaître sera représenté de manière similaire et l'identification de l'objet sera exprimée comme la recherche de la meilleure correspondance entre les graphes. Autre exemple, imaginons que l'on veuille reconnaître non plus un objet parmi un ensemble d'objets, mais tous les objets constituant une scène dont on connaît un modèle. Il s'agira par exemple de reconnaître des structures anatomiques dans une image médicale à partir d'un atlas anatomique, ou des

structures urbaines à partir d'une carte de la région imagée. Le modèle sera représenté par un graphe dans lequel chaque nœud correspond à une structure et les arcs à des relations spatiales entre structures. Dans les images, il est très difficile de trouver une segmentation parfaite et on devra souvent se contenter d'une sur-segmentation. Dans le graphe correspondant, plusieurs nœuds correspondront à une structure. C'est un exemple typique dans laquelle il faut abandonner la condition de bijectivité dans la mise en correspondance. On se tournera donc vers des méthodes de mise en correspondance inexacte, dans lesquelles les similarités entre attributs joueront un rôle important pour prendre en compte le côté souvent schématique du modèle ainsi que les imperfections des images.

Chapitre 4

Ouverture vers d'autres approches

Dans ce chapitre, nous donnons quelque coups de projecteur sur d'autres approches, sortant du cadre de ce cours, mais qui connaissent de larges développements. Des théories numériques non probabilistes ont vu le jour dans les années 1960 et 1970, et sont depuis reprises pour modéliser et résoudre des problèmes de reconnaissance des formes : théorie des ensembles flous et des possibilités, et théorie des fonctions de croyance. Nous en donnons les principes, sans entrer dans les détails, dans les sections 4.1 et 4.2. Il faut ensuite mentionner les méthodes s'appuyant sur les réseaux neuro-mimétiques (section 4.3) qui figurent ici pour mémoire, un cours complet leur étant dédié. Enfin une ouverture sur des approches symboliques telles que celles développées en intelligence artificielle, est brièvement proposée dans la section 4.4. Ces approches s'appuient à la fois sur le domaine de la représentation de connaissances et sur celui du raisonnement.

4.1 Ensembles flous et possibilités

4.1.1 Définitions

La théorie des ensembles flous a été introduite par L. Zadeh et le premier article sur le sujet date de 1965 [27].

Soit \mathcal{S} l'univers, ou espace de référence. C'est un ensemble classique (ou net). On notera par x, y , etc. ses éléments (ou points). En traitement d'images, \mathcal{S} sera typiquement l'espace dans lequel est définie l'image (\mathbb{Z}^n ou \mathbb{R}^n , avec $n = 2, 3, \dots$). Les éléments de \mathcal{S} sont alors les points de l'image (pixels, voxels). L'univers peut également être un ensemble de valeurs prises par des caractéristiques de l'image telles que l'échelle des niveaux de gris. Les éléments sont alors des valeurs (niveaux de gris). L'ensemble \mathcal{S} peut aussi être un ensemble de primitives ou objets extraits des images (segments, régions, objets...) dans une représentation de plus haut niveau du contenu de l'image.

Un sous-ensemble X de \mathcal{S} est défini par sa fonction caractéristique μ_X , telle que :

$$\mu_X(x) = \begin{cases} 1 & \text{si } x \in X \\ 0 & \text{si } x \notin X \end{cases} \quad (4.1)$$

La fonction caractéristique μ_X est une fonction binaire, spécifiant l'appartenance (binaire) de chaque point de \mathcal{S} à X .

La théorie des ensembles flous traite de l'appartenance graduelle. Un sous-ensemble flou de \mathcal{S} est défini par sa fonction d'appartenance μ de \mathcal{S} dans $[0, 1]^1$. Pour tout x de \mathcal{S} , $\mu(x)$ est la valeur dans $[0, 1]$ représentant le degré d'appartenance de x au sous-ensemble flou (on dit souvent plus simplement « ensemble flou »).

Pour des problèmes de classifications dans lesquelles les classes sont mal définies ou ont des frontières imprécises, une représentation d'une classe par un ensemble flou défini sur l'espace de représentation permet de bien tenir compte de ces imprécisions.

La théorie des possibilités, dérivée de celle des ensembles flous, a été introduite par L. Zadeh dans [28], et ensuite été développée par plusieurs chercheurs, en particulier D. Dubois et H. Prade en France [6, 7]. Elle a également donné lieu à des développements en classification : les classes sont en générale considérées comme des ensembles classiques, et on définit des distributions de possibilités, chaque valeur représentant le degré de possibilité pour qu'une forme appartienne à une classe.

4.1.2 Classification floue

Nous donnons dans cette partie des exemples de méthodes de classification floue. Outre leur utilisation pour des problèmes de classification, elles sont souvent utilisées pour estimer des fonctions d'appartenance dans des espaces de caractéristiques.

C-moyennes floues. Un des algorithmes les plus classiques est celui des « fuzzy C-means » (C-moyennes floues) [3], qui généralise l'algorithme des k-moyennes vues au chapitre 2, toujours en optimisant de manière itérative une fonction objectif, s'écrivant :

$$J_m = \sum_{j=1}^C \sum_i^N \mu_{ij}^m \|x_i - m_j\|^2, \quad (4.2)$$

sous la contrainte $\sum_{j=1}^C \mu_{ij} = 1$, où m est un paramètre dans $]1, +\infty[$ appelé facteur de flou. La notation μ_{ij} représente le degré d'appartenance de la forme i (de caractéristiques x_i) à la classe j et C est le nombre de classes. Chaque classe j est représentée, dans le même espace que les x_i , par un « centre », noté m_j .

La minimisation de cette fonctionnelle donne :

$$\mu_{ij} = \frac{1}{\sum_{j=1}^C \left[\frac{\|x_i - m_i\|}{\|x_i - m_j\|} \right]^{\frac{2}{m-1}}}, \quad (4.3)$$

et les centres des classes sont donnés par :

$$m_j = \frac{\sum_i \mu_{ij}^m x_i}{\sum_i \mu_{ij}^m}. \quad (4.4)$$

Le principe de l'algorithme est exactement le même que dans les k-moyennes, sauf qu'aucune affectation n'est réalisée au cours de l'algorithme : à chaque étape sont calculés des

¹L'intervalle $[0, 1]$ est le plus utilisé, mais n'importe quel intervalle ou autre ensemble (un treillis typique) pourrait être utilisé.

degrés d'appartenance de chaque forme à toutes les classes. L'algorithme itère, à partir d'une initialisation des centres de classes, le calcul des μ_{ij} et le calcul des centres, jusqu'à ce qu'un critère de convergence soit atteint (en général un seuil sur la variation des m_j entre deux itérations). Si une affectation nette est nécessaire, une étape finale de décision est ajoutée. On choisit en général d'affecter x_i à la classe j qui maximise μ_{ij} . Cette décision peut être assortie de règles de rejet. Par exemple une règle classique de rejet en distance consiste à vérifier que la valeur maximale retenue est supérieure à un certain seuil. Une règle classique de rejet en ambiguïté portera par exemple sur la différence entre les deux valeurs maximales de μ_{ij} (à i fixé).

Un des inconvénients majeurs de cette méthode est la forme des fonctions d'appartenance, qui ne sont pas décroissantes en fonction de la distance au centre de la classe. Cette propriété est induite essentiellement par la contrainte de normalisation des fonctions d'appartenance.

C-moyennes possibilistes. Pour éviter ce comportement contre-intuitif, a été proposé l'algorithme des C-moyennes possibilistes [16], dans lequel la fonction objectif est modifiée comme suit :

$$J = \sum_{j=1}^C \sum_{i=1}^N \mu_{ij}^m \cdot \|x_i - m_j\|^2 + \sum_{j=1}^k \eta_j \sum_{i=1}^N (1 - \mu_{ij})^m \cdot \|x_i - m_j\|^2, \quad (4.5)$$

où η_j est un paramètre associé à la classe j qui contrôle la décroissance des fonctions d'appartenance et détermine la distance au centre de la classe à laquelle on aura une appartenance égale à 1/2.

Cette fois l'optimisation donne :

$$\mu_{ij} = \frac{1}{1 + \frac{\|x_i - m_j\|^2 \frac{1}{m-1}}{\eta_j}} \quad (4.6)$$

qui est une fonction décroissante en fonction de $\|x_i - m_j\|$ ce qui est intuitivement bien plus satisfaisant.

k-ppv flous. Enfin, signalons une extension de la méthode des k plus proches voisins au cas flou.

Dans le cas binaire, l'appartenance à une classe peut s'écrire sous une forme fonctionnellem en faisant intervenir la fonction d'Heaviside U :

$$\mu_i(x) = U\left[\sum_{j=1}^k \mu_i(x_j)\right] \quad (4.7)$$

avec

$$U(t) = \begin{cases} 1 & \text{si } t > \frac{k}{2} \\ 0 & \text{sinon} \end{cases}$$

Cette forme permet une extension simple au cas flou, en remplaçant la fonction en « marche d'escalier » par une fonction plus souple, par exemple :

$$\mu_i(x) = \frac{1}{1 + \exp\left[\frac{1}{b_i}\left(\frac{k}{2} - t_i(x)\right)\right]} \quad (4.8)$$

avec

$$t_i(x) = \sum_{j=1}^k \mu_i(x_j)$$

ou encore, en tenant compte des distances aux voisins :

$$t_i(x) = \sum_{j=1}^k \mu_i(x_j) \exp[-\lambda (\frac{d(x, x_j)}{d_m^i})^2].$$

Le paramètre b_i permet de contrôler le flou, et peut être défini par exemple en fonction de l'entropie floue :

$$b_i = H(i) = \frac{1}{K} \sum_x [\mu_i(x) \log(\mu_i(x)) + (1 - \mu_i(x)) \log(1 - \mu_i(x))].$$

4.2 Fonctions de croyance

Dans la théorie des fonctions de croyance [23], on caractérise un élément d'un espace D appelé « ensemble de discernement » par un couple de valeurs, appelées fonctions de croyance Bel et de plausibilité Pls , définies à partir d'une fonction appelée fonction de masse m . La fonction m est une fonction de 2^D dans $[0, 1]$ et on impose souvent $\sum_{A \subseteq D} m(A) = 1$. Cette fonction porte donc potentiellement sur tous les sous-ensembles de D , et pas seulement sur les singletons comme on le ferait dans une approche probabiliste. La fonction Bel est définie par :

$$\forall A \in 2^D, Bel(A) = \sum_{B \subseteq A, B \neq \emptyset} m(B). \quad (4.9)$$

et la plausibilité Pls par :

$$\forall A \in 2^D, Pls(A) = \sum_{B \cap A \neq \emptyset} m(B) = 1 - Bel(A^C). \quad (4.10)$$

Si l'ensemble de discernement D est l'espace de décision, donc l'ensemble des classes, on voit que ce modèle permet de prendre naturellement en compte l'ambiguïté entre classes (en affectant une masse à la réunion de ces classes plutôt qu'à chacune séparément), et la confiance et l'ignorance que l'on peut avoir sur une décision A (sous-ensemble de D), mesurée par l'intervalle $[Bel(A), Pls(A)]$.

Différents modèles ont été proposés pour l'estimation des fonctions de masse (par exemple en fonction de distances à des prototypes des classes) et l'utilisation de ce formalisme en fusion d'informations [25] et en reconnaissance des formes [5, 10].

4.3 Réseaux de neurones

Les réseaux de neurones sont présentés plus en détails dans le cours *Apprentissage numérique et réseaux de neurones*. Nous n'en donnons ici que les principes pour la reconnaissance des formes [19].

Un réseau connexionniste est défini par un ensemble d'éléments dont les liens sont définis par une certaine architecture. Les entités du réseau sont appelées neurones ou cellules. Les connexions reliant les entités du réseau sont munies de valeurs numériques, appelées poids. La dynamique définit comment les informations se propagent dans le réseau, comment les entités sont couplées.

L'architecture organise les neurones en couches par exemple, les connexions ne se faisant que d'une couche à l'autre. Par exemple une couche d'entrée va correspondre aux caractéristiques des formes à reconnaître alors qu'une couche de sortie correspondra aux différentes classes dans lesquelles les formes peuvent être affectées. Les couches intermédiaires, ou couches cachées ont une interprétation souvent moins directe et sont chargées de faire l'affectation, le calcul étant défini à la fois par l'architecture et par les poids.

Un aspect important est celui de l'apprentissage, consistant à déterminer les poids optimaux pour que, sur un ensemble connu appelé ensemble d'apprentissage, les sorties soient bien celles qui sont attendues. L'ensemble d'apprentissage doit être représentatif des différentes classes possibles. Un réseau est également caractérisé par ses capacités de généralisation, qui dépendent fortement de l'ensemble d'apprentissage.

Les réseaux les plus simples (perceptrons) réalisent une séparation linéaire des classes. Des architectures plus complexes, avec plus de couches par exemples, permettent d'effectuer des classifications plus sophistiquées.

4.4 Méthodes symboliques

De l'intelligence artificielle sont nées des représentations symboliques des connaissances. Le domaine de la représentation de connaissances (*knowledge representation*) se caractérise par :

- la définition d'une représentation comme un ensemble de conventions syntaxiques et sémantiques pour décrire un élément de connaissance ;
- des représentations logiques (dont l'expressivité dépend de la logique utilisée) ;
- des représentations compactes (seules les propriétés et caractéristiques pertinentes sont explicitées) ;
- une facilité de manipulation ;
- ce qui est important est effectivement explicite.

La plupart des données manipulées dans les domaines de l'image sont analogiques ou numériques. Les représentations analogiques nécessitent une description complète du monde. Le passage à des représentations logiques, moins coûteuses, plus compactes, nécessite une conversion de l'analogique vers le symbolique.

Les exigences des représentations symboliques se situent à plusieurs niveaux :

- au niveau ontologique : tous les concepts importants doivent être pris en compte ;
- au niveau épistémique : on ne doit pas être obligé d'exprimer ce qui n'est pas connu ;
- au niveau computationnel : la représentation doit permettre un calcul efficace des propriétés exprimées.

Les deux premiers niveaux induisent des contraintes sur le langage et le troisième sur les mécanismes d'inférence.

La communauté de la représentation de connaissances (symbolique) s'intéresse au raisonnement non monotone, au raisonnement automatique, aux logiques de description, aux

représentations subjectives (préférences, souhaits...), aux ontologies, etc. [20]. Plus près de notre propos, elle s'intéresse également à l'apprentissage, à l'intégration et la fusion de bases de connaissances, à la décision et au diagnostic, au raisonnement temporel et spatial, à la représentation des actions et à la planification. Il y a sans aucun doute des pistes à explorer dans ce sens.

4.4.1 Systèmes à base de connaissances

De tels systèmes peuvent être vus comme des extensions des systèmes experts classiques, en permettant des modes différents de représentation des connaissances et de raisonnement.

Les systèmes à base de règles sont détaillés dans le cours *Systèmes à base de règles, prolog*.

La construction d'un système à base de connaissance implique des acteurs ayant trois rôles distincts :

- l'utilisateur remplit la base de faits avec les données à traiter ;
- l'expert construit la base de connaissances ;
- le développeur construit le moteur d'inférence et la stratégie de raisonnement.

Parmi les différents types de connaissances représentées dans ces systèmes, on distingue les connaissances :

- déclaratives (comment sont les choses) ;
- procédurales (comment on fait) ;
- épisodiques (relatives à l'expérience précédente) ;
- et les métaconnaissances (connaissances sur la connaissance).

Il est en général extrêmement difficile de passer d'un type de connaissance à l'autre (et même souvent impossible) et donc ces connaissances sont par essence bien différentes et peuvent toutes être nécessaires.

Le contrôle consiste à rechercher des chemins entre les connaissances initiales et les buts, par des techniques de chaînage avant (application de règles d'inférence lorsque de nouvelles données sont déclarées, les conséquences pouvant alors déclencher de nouvelles règles d'inférence), ou chaînage arrière (application de règles d'inférence lorsque de nouvelles requêtes sont formulées, les prémisses non encore vérifiées de ces règles conduisant alors au déclenchement de nouvelles règles).

Les exemples les plus classiques de systèmes à base de connaissances comportent :

- les règles de production ;
- les *frames* ;
- les réseaux sémantiques ;
- les systèmes avec incertitude : Mycin [24], etc.

Les systèmes de règles de production (de la forme *si... et/ou... alors...*) sont des systèmes faciles à adapter ou à étendre et dont le fonctionnement et les résultats peuvent être facilement expliqués. Ils ont l'inconvénient d'une représentation fragmentée de la connaissance induisant un manque d'efficacité. Leur pouvoir d'expression dépend essentiellement du type de logique utilisée. Par exemple la logique du premier ordre ou des prédicats permet de manipuler des variables et des quantificateurs, par opposition à la logique propositionnelle où tout est constant.

Les *frames* constituent une forme déclarative de systèmes à base de connaissances, dans

lesquels une liste d'attributs ou de propriétés est assortie de caractéristiques et de valeurs de ces caractéristiques. Ils trouvent leur utilité dans la description de concepts généraux, de classes d'objets. Des liens hiérarchiques, d'héritage, de spécialisation et d'instanciation permettent de manipuler des classes de granularités différentes. Ces systèmes sont le plus souvent statiques, mais une certaine dynamique peut y être introduite en affectant des procédures aux attributs.

Les réseaux sémantiques s'appuient sur une représentation graphique de la base de connaissances, dans laquelle les nœuds représentent les concepts et les objets, et les arcs représentent des relations. Les règles d'inférence s'appuient sur des propriétés d'héritage lorsque l'on passe par des arcs d'une classe à une classe plus spécifique. Ces réseaux sont beaucoup utilisés pour le traitement du langage naturel par exemple.

En vision, des tâches spécifiques de focalisation et adaptation (avec leurs mécanismes attentionnel, de révision ou réparation et de maintien de la cohérence), de coopération et fusion (confrontative, augmentative, intégrative), de coordination (délibérative, réactive, optimale) sont ajoutées aux systèmes à base de connaissances. Ces approches sont détaillées dans [13] et ne sont pas reprises ici.

4.4.2 Modes de raisonnement et inférence

Les modes d'inférence utilisés dans les systèmes à base de connaissances sont plus variés que dans les systèmes experts classiques. On distingue les modes suivants :

- la déduction, qui fournit des conséquences à partir de faits (par exemple si la base de faits contient A et la proposition $A \rightarrow B$, alors on peut déduire B) ;
- la contraposition permet de raisonner sur les non-observations (par exemple si l'on a $A \rightarrow B$ et non B , on peut inférer non A) ;
- l'abduction cherche à remonter aux causes expliquant les observations (par exemple de $A \rightarrow B$ et de l'observation de B , on infère que A est une cause possible de B) ;
- l'induction permet d'inférer des règles à partir d'observations régulières ou habituelles (par exemple, si l'on a B à chaque fois que l'on a A , on peut inférer $A \rightarrow B$) ;
- la projection fournit des conséquences à partir d'actions (si la base de faits contient la proposition $A \rightarrow B$ et que l'on fait A , on s'attend à ce que B soit réalisé) ;
- la planification établit les actions à effectuer pour attendre des buts (si l'on veut B et que la base contient $A \rightarrow B$, alors on infère l'action A).

Ces deux derniers modes d'inférence sont particulièrement développés dans les systèmes à base de connaissances embarqués, tels que ceux qui sont utilisés par exemple en robotique mobile [21].

L'interprétation d'images et la fusion d'informations imposent souvent de faire appel à différents modes de raisonnement, afin de mieux saisir et représenter les finesses et les subtilités du raisonnement humain.

En raisonnement monotone, l'obtention de plus d'informations conduit naturellement à plus de conclusions : si d'une base KB on déduit A , on déduira également A de $KB \cup B$. Les logiques classiques, propositionnelle et du premier ordre, relèvent de ce mode de raisonnement.

En raisonnement non monotone, une nouvelle information peut invalider des conclusions précédentes. En présence d'informations et de connaissances imparfaites comme c'est le cas

en fusion d'informations, les sources de non-monotonie dans le raisonnement viennent essentiellement des hypothèses et des restrictions qui sont effectuées. Celles-ci sont nécessaires pour pouvoir raisonner, mais peuvent être remises en cause si de nouvelles informations ou éléments de connaissance sont disponibles. Ces hypothèses à la source de la non-monotonie comportent :

- l'utilisation de propriétés typiques ;
- les exceptions possibles ;
- l'hypothèse de monde fermé.

Les logiques non monotones reposent sur des notions de préférence (quel monde, quelle situation sont plus « normaux » que d'autres, quels sont les buts préférés si tous ne peuvent être atteints...), de changement ou révision de croyances (les fameux postulats AGM [2]), et bien sûr sur un certain nombre de postulats gérant la non-monotonie, appelés postulats de rationalité. Un exemple de tels postulats, de monotonie « prudente » exprime que si une base KB permet de déduire $A \rightarrow B$ et permet de déduire C , alors elle permet de déduire $A \wedge C \rightarrow B$.

Les notions de contingence, de vérité nécessaire ou possible, ne sont pas bien représentées en logique classique. Manipuler de tels concepts implique l'introduction de modalités dans la logique. Les logiques modales [4, 15] permettent de raisonner sur des propositions A (A est vrai), $\Box A$ (A est nécessaire), $\Diamond A$ (A est possible). Des formes numériques de ces notions se retrouvent dans la théorie des fonctions de croyance en termes de croyance et plausibilité et dans la théorie des possibilités en termes de nécessité et possibilité.

Enfin, les notions d'imprécision et d'incertitude, peuvent être représentées dans les logiques floues et possibilistes.

Bibliographie

- [1] A. V. Aho, J. E. Hopcroft, and J. P. Ullman. *The Design and Analysis of Computer Algorithms*. Addison Wesley, 1975.
- [2] C. E. Alchourrón, P. Gärdenfors, and D. Makinson. On the Logic of Theory Change : Partial Meet Contraction and Revision Functions. *Journal of Symbolic Logic*, 50 :510–530, 1985.
- [3] J. C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Plenum, New-York, 1981.
- [4] B. Chellas. *Modal Logic, an Introduction*. Cambridge University Press, Cambridge, 1980.
- [5] T. Dencœux. A k-nearest Neighbor Classification Rule based on Dempster-Shafer Theory. *IEEE Transactions on Systems, Man and Cybernetics*, 25(5) :804–813, 1995.
- [6] D. Dubois and H. Prade. *Fuzzy Sets and Systems : Theory and Applications*. Academic Press, New-York, 1980.
- [7] D. Dubois and H. Prade. *Possibility Theory*. Plenum Press, New-York, 1988.
- [8] B. Dubuisson. *Diagnostic et reconnaissance des formes*. Hermès, Paris, 1990.
- [9] R. Duda and P. Hart. *Pattern Classification and Scene Analysis*. Wiley, New-York, 1973.
- [10] B. Dubuisson (Ed.). *Diagnostic, intelligence artificielle et reconnaissance des formes*. Hermès, Paris, 2001.
- [11] K. S. Fu. *Syntactic Methods in Pattern Recognition*. Academic Press, 1974.
- [12] G. Gaillat. *Méthodes statistiques de reconnaissance de formes*. Coll. ENSTA, Paris, 1983.
- [13] C. Garbay. Architectures logicielles et contrôle dans les systèmes de vision. In J. M. Jolion, editor, *Les systèmes de Vision*, chapter 7, pages 197–252. Hermès, 2001.
- [14] M. Gondran and M. Minoux. *Graphes et algorithmes*. Eyrolles, Paris, 1980.
- [15] G. E. Hughes and M. J. Cresswell. *An Introduction to Modal Logic*. Methuen, London, UK, 1968.
- [16] R. Krishnapuram and J. M. Keller. A Possibilistic Approach to Clustering. *IEEE Transactions on Fuzzy Systems*, 1(2) :98–110, 1993.
- [17] S. Y. Lu. A Tree to Tree Distance and its Application to Cluster Analysis. *IEEE Transactions in Pattern Analysis and Machine Intelligence*, 1(2), 1979.
- [18] L. Miclet. *Méthodes structurelles pour la reconnaissance des formes*. Eyrolles, Collection CNET-ENST, Paris, 1984.

- [19] M. Milgram. *Reconnaissance des formes : méthodes numériques et connexionnistes*. Armad Colin, Paris, France, 1993.
- [20] H. Reichgelt. *Knowledge Representation : An AI Perspective*. Ablex Publishing, 1991.
- [21] A. Saffiotti. Artificial Intelligence fo Embedded Systems. Technical report, AASS, University of Örebro, Sweden, 2002.
- [22] S. M. Selkow. The Tree-to-Tree Editing Problem. *Information Processing Letters*, 6(6), 1977.
- [23] G. Shafer. *A Mathematical Theory of Evidence*. Princeton University Press, 1976.
- [24] E.H. Shortliffe. *Computer Based medical consultation : Mycin*. Elsevier, 1976.
- [25] I. Bloch (sous la direction de). *Fusion d'informations en traitement du signal et des images*. Hermès, Paris, France, 2003.
- [26] R. A. Wagner and M. J. Fisher. The String to String Correction Problem. *Journal of ACM*, 21(1) :168–173, 1974.
- [27] L. A. Zadeh. Fuzzy Sets. *Information and Control*, 8 :338–353, 1965.
- [28] L. A. Zadeh. Fuzzy Sets as a Basis for a Theory of Possibility. *Fuzzy Sets and Systems*, 1 :3–28, 1978.