Chapitre 2

Les cartes auto-organisatrices

Sommaire

1 Présentation 38
1.1 L'inspiration biologique 38
1.2 Cartes de Kohonen 40
1.3 Cas d'utilisation 43
2 Algorithme 44
2.1 Complexité 44
2.2 Version batch 46
2.3 Hyper-paramètres 47
3 Adaptation à l'apprentissage supervisé 51
3.1 Apprentissage par quantification vectorielle 51
3.2 Modèle LASSO 53
3.3 Cartes étiquetées 54
3.4 Récapitulatif 56
4 Validation statistique 57
4.1 Apprentissage non-supervisé 57
4.2 Apprentissage supervisé 61
5 Visualisation de l'apprentissage 64
5.1 En contexte non-supervisé 65
5.2 En contexte supervisé 68
5.3 Contextualisation et navigation 70
6 Expérimentations 71
6.1 Carte étiquetée : Kohonen-Opt 71
6.2 Validation statistique 72
6.3 Passage à l'échelle 74
7 Bilan et perspectives 76

Les cartes auto-organisatrices

Publications

Prudhomme, E. et S. Lallich (2005). Validation statistique des cartes de Kohonen en apprentissage supervisé. In *Extraction des connaissances et apprentissage (Actes de EGC'2005)*, Volume 1 of *RNTI-E-3*, pp. 79–90. Cépaduès Editions.

Prudhomme, E. et S. Lallich (2005). Quality measure based on Kohonen maps for supervised learning of large high dimensional data. In *Proc. of International Symposium on Applied Stochastic Models and Data Analysis (ASMDA'2005)*, Brest, France, pp. 246–255.

Dans ce chapitre, nous présentons les cartes auto-organisatrices et leur utilisation comme outil de représentation des données. Les trois premières sections de ce chapitre sont consacrées à l'obtention d'une telle représentation. Nous commençons par introduire les cartes auto-organisatrices en apprentissage non-supervisé en donnant quelques exemples d'application en fouille de données (section 1) ainsi que le détail de différents algorithmes qui permettent de générer de telles cartes (section 2). Nous présentons alors plusieurs adaptations de ces algorithmes à l'apprentissage supervisé (section 3) dont Kohonen-Opt, une méthode que nous avons proposée au cours de nos recherches.

Dans les trois sections suivantes, nous nous intéressons à l'exploitation de cette représentation. En section 4, nous décrivons différentes manières de valider statistiquement la représentation, que ce soit d'un point de vue non-supervisé (validation de la modélisation des distances) ou supervisé. Sur ce dernier point, nous proposons une mesure de validation de la qualité de cette représentation. Dans la section 5, nous nous intéressons aux méthodes de visualisation des cartes auto-organisatrices puis, en section 5.3, aux aspects de contextualisation et de navigation.

Enfin, nous menons plusieurs expérimentations en section 6. En premier lieu, nous validons nos propositions, aussi bien sur la fonction de prédiction Kohonen-Opt, que sur notre mesure statistique de la qualité de la représentation. Ensuite, nous vérifions les capacités de passage à l'échelle des cartes lorsque le nombre d'individus ou d'attributs augmente. Les difficultés des cartes face à l'augmentation du nombre d'attributs nous conduisent à proposer en section 7 de nouvelles solutions qui ouvrent sur le chapitre 3.

1 Présentation

1.1 L'inspiration biologique

À l'origine, les premiers travaux proposant des cartes auto-organisatrices étaient orientés vers la modélisation de zones corticales organisant spatialement les informations des connexions afférentes. Ce type d'organisation des informations traitées par le cerveau est bien illustré par le mécanisme de la vision – historiquement la voie sensorielle la plus étudiée en neurosciences. Au cours de ce mécanisme, la rétine envoie les stimuli visuels captés au cortex visuel primaire (aire V1) qui conserve l'organisation spatiale du champ visuel. La figure 2.1 illustre ce mécanisme en Imagerie par Résonnance Magnétique fonctionnelle (IRMf) lors de l'activation de l'aire V1 par des stimuli visuels de type mire en 3D (figure de gauche) puis en 2D (figure de droite, reconstituée à partir des données 3D). Sur la représentation 2D, on constate que l'organisation spatiale des couleurs les unes par rapport aux autres est conservée entre la mire et l'aire visuelle, phénomène désigné sous le terme de rétinotopie. Elle n'est pas exacte puisque les zones centrales qui correspondent à la vision de la fovéa sont sur-représentées par le cortex visuel pour assurer des traitements plus fins de ces informations centrales.

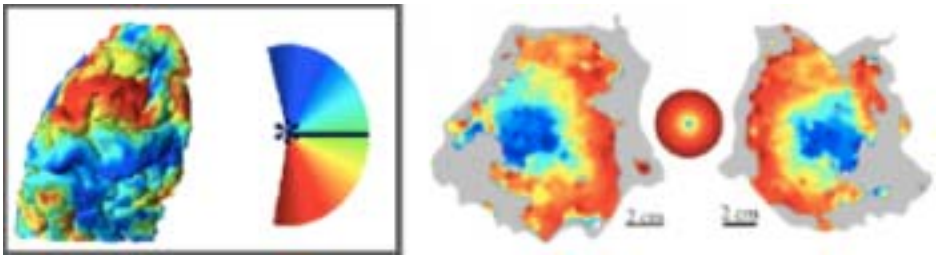


Fig. 2.1: Illustration de la rétinotopie.

La rétinotopie n'est pas la seule organisation somatotopique du cortex visuel. D'autres organisations de ce type se mettent en place au fur et à mesure que l'information visuelle est traitée. Un exemple connu concerne la détermination de l'orientation de la lumière perçue en chaque point de l'espace visuel, décrite par Hubel et Wiesel[(1)] dès 1959. Ils ont montré que chaque point de l'espace était analysé au sein d'un module cortical spécifique qu'ils ont appelé hypercolonne (à gauche sur la figure 2.2). Parmi les neurones présents dans les hypercolonnes, les cellules dites simples s'activent ou non suivant un mécanisme de type *on/off* en fonction de l'orientation du stimulus qu'elles reçoivent. En mesurant l'activité de ces cellules à l'aide d'une électrode (plantée dans le cerveau d'un chat), Hubel et Wiesel ont remarqué que si l'électrode avance perpendiculairement par rapport au cortex, alors elle

[1] Hubel et Wiesel ont obtenu le prix Nobel de médecine pour leurs travaux sur les colonnes de dominance visuelle et la neurophysiologie visuelle en général en 1981.

rencontre des cellules répondant toujours à la même orientation. En revanche, en déplaçant l'électrode parallèlement par rapport au cortex, elle rencontre des cellules dont l'orientation d'activation change progressivement pour réaliser une rotation de 180 degrés tous les 1mm (voir Hubel et Wiesel (1962) pour une description de l'architecture fonctionnelle du cortex visuel du chat). Sur une ligne, le cortex a ainsi une organisation topique de l'orientation des stimuli.

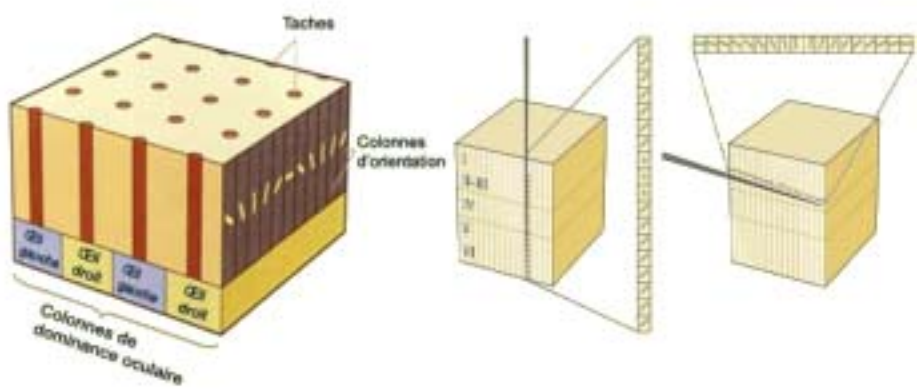


Fig. 2.2: À gauche, illustration d'une hypercolonne du cortex visuel, module cortical nécessaire et suffisant à l'analyse d'un point de l'espace visuel. À droite, disposition somatotopique des colonnes d'orientation au sein d'une hypercolonne.

Hubel et Wiesel ont également travaillé sur la formation de ces colonnes au cours du développement du chat. Ils ont ainsi pu montrer que ces colonnes ne sont pas totalement prédéterminées à la naissance. Au contraire, elles évoluent en entrant en compétition les unes avec les autres pour représenter les différentes zones du champ visuel. Par exemple, si un œil est obstrué au cours du développement, les colonnes de dominance qui auraient dû représenter la vision de cet œil sont recrutées pour représenter l'œil restant. Pour comprendre comment cette organisation corticale émerge au cours du développement, von Der Malsburg (1973) a proposé un modèle computationnel des colonnes d'orientation puis de la rétinotopie (Willshaw et von der Malsburg, 1976). Ces travaux ont montré qu'il était possible d'auto-organiser des neurones computationnels en cartes sur lesquelles deux neurones voisins répondent à des stimuli proches, en appliquant des règles d'évolution relativement simples.

1.2 Cartes de Kohonen

Les cartes auto-organisatrices de Kohonen (1982) sont une version plus aboutie des propositions de von der Malsburg. Ces cartes sont composées d'un vecteur de neurones (ou modèles) M de taille m, répartis uniformément sur une grille en deux dimensions. En plus de sa position sur la carte, chaque modèle est associé à une position dans l'espace d'apprentissage, appelée vecteur du modèle (ici m_i représente le i^{e} vecteur de M). Ces vecteurs sont mis à jour au cours de l'apprentissage pour produire l'auto-organisation des modèles en fonction de leur similarité, de telle sorte qu'à la fin de l'apprentissage deux modèles proches sur la carte représentent deux régions proches dans l'espace d'apprentissage.

L'apprentissage en lui-même se déroule de manière itérative. À chaque itération t, un exemple x est présenté à la carte. La première étape consiste à trouver le modèle qui en assure la meilleure représentation. Ce modèle, dit vainqueur, est celui qui minimise la distance entre l'exemple et lui-même, dont l'indice dans M pour l'exemple x est noté :

$$c = \text{argmin}_i(d(x, m_i))$$

Une fois ce modèle déterminé, celui-ci est mis à jour pour prendre en compte l'exemple présenté. Cette mise à jour rapproche le vecteur du modèle de l'exemple dans l'espace d'apprentissage à une vitesse $\alpha(t)$ qui décroît avec le nombre d'itérations. Ici, à l'inverse des méthodes de type *winner-take-all* où seul le vainqueur est modifié, les modèles voisins du modèle vainqueur sont également mis à jour. L'étendue du voisinage concernée par cette mise à jour est modélisée par la fonction $\sigma(t)$ qui décroît à chaque itération. Cette étape assure l'auto-organisation de la position des modèles sur la carte avec la position de leurs vecteurs dans l'espace d'apprentissage. De plus, bien que plusieurs modèles soient mis à jour simultanément, l'ampleur de la modification n'est pas la même pour tous. L'ampleur de la modification est déterminée en fonction de la position du modèle par rapport au vainqueur de telle sorte que plus il est proche du vainqueur sur la carte, plus la modification le rapproche de l'exemple (en notant r_i la position du modèle i sur la carte et r_c la position du modèle vainqueur, la distance entre les modèles i et c sur la carte est noté $\|ri - r_c\|$). Il s'agit donc d'une méthode de type *winner-take-most* qui assure au vainqueur d'être le plus modifié par l'apprentissage de l'exemple. Ce dernier aspect de l'algorithme est modélisé par une fonction de voisinage $N_v(t)$. La règle de modification des poids est donc construite de sorte que, en début d'apprentissage, tout à la fois les modèles à modifier se rapprochent fortement des exemples ($\alpha(t)$ est grand) et le nombre de modèles à modifier autour du modèle vainqueur (le voisinage) soit large ($N_v(t)$ grand). Par la suite, l'ampleur de la modification et le nombre de modèles à modifier décroissent. Finalement, la règle de modification d'un vecteur modèle m_i à l'instant t de présentation de l'exemple x est donnée par :

$$m_i(t+1) = m_i(t) + \alpha(t)N_v(t)\left(x(t) - m_i(t)\right) \tag{2.1}$$

La fonction de voisinage $N_v(t)$ peut soit être une fonction rectangulaire (*bubble*) définissant un périmètre au sein duquel l'ampleur de la modification est la même pour tous les modèles, soit prendre la forme d'une gaussienne de manière à graduer la modification des modèles en fonction de leur éloignement du vainqueur. Le tableau ci-dessous donne le détail de ces deux fonctions et le graphique 2.3 montre les valeurs prises par une fonction $N_v(t)$ gaussienne pour un modèle i donné, en fonction de sa distance au vainqueur $\|r_i - r_c\|$ et de la taille du voisinage à modifier à l'itération t, $\sigma(t)$. Enfin, les fonctions $\alpha(t)$ et $\sigma(t)$ sont généralement choisies de manière à décroître linéairement en $\frac{1}{t}$.

Rectangulaire	Gaussien
$N_v(t) = \begin{cases} 1 & \text{if } \|r_i - r_c\| \leq \sigma(t); \\ 0 & \text{sinon.} \end{cases}$	$N_v(t) = e^{-\frac{\|r_i - r_c\|^2}{\sigma^2(t)}}$

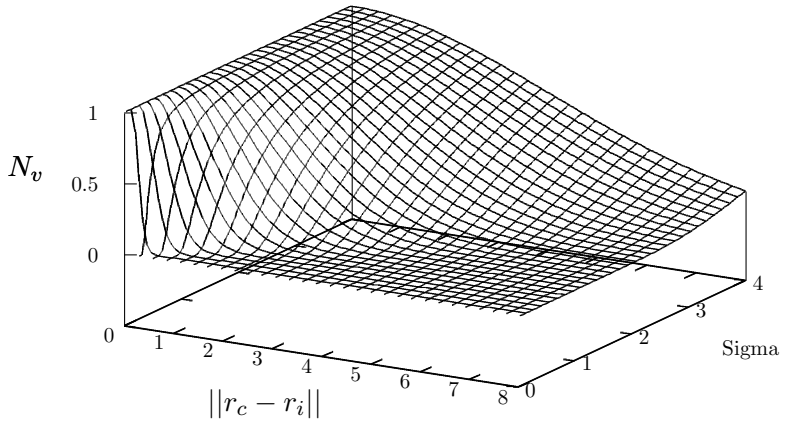


Fig. 2.3: Fonction de voisinage gaussienne des cartes auto-organisatrices

Le résultat de cet algorithme est un apprentissage non-supervisé des exemples (réalisé uniquement à partir de leurs positions dans l'espace d'apprentissage, sans se soucier qu'ils aient ou non une classe), apprentissage qui assure leur projection vers l'espace de la carte. L'un des points forts des cartes auto-organisatrices est que cette projection de l'espace est non-linéaire (contrairement à ce qui se passe pour l'ACP, voir section 2.1). Cette non-linéarité permet aux cartes de coller au plus près de la distribution des données, même dans les cas où cette distribution est complexe. Ainsi, pour chaque exemple dont on connaît les coordonnées dans l'espace d'apprentissage, il est possible de lui associer des coordonnées sur la carte (l'espace de projection) qui seront celles du modèle qui représente l'exemple. La projection conserve les propriétés de la topologie locale de l'espace d'apprentissage, c'est à dire que si deux exemples sont proches dans l'espace d'apprentissage alors

ils auront des modèles proches (voire identiques) sur la carte. Cela signifie également que cette projection est discrète puisque deux exemples ayant des coordonnées suffisamment proches dans l'espace d'apprentissage seront représentés par le même modèle et auront des coordonnées identiques sur la carte.

Si les cartes auto-organisatrices font partie de la famille des méthodes de projection, ce sont également des méthodes de quantification vectorielle. Ces méthodes recherchent un ensemble de points (appelés prototypes ou *codebook vector* en anglais) qui résument au mieux les données. Le résultat d'une méthode de quantification vectorielle est donc un dictionnaire (*codebook*) qui contient l'ensemble des prototypes. Ces méthodes trouvent ainsi une utilité naturelle en apprentissage non-supervisé, où elles servent à trouver une partition des données capable d'associer à chaque classe un exemple type.

En pratique, les méthodes de quantification vectorielle cherchent à minimiser la distorsion Δ entre les exemples et les prototypes qui les représentent, au sens d'une mesure de distorsion (le plus souvent la distance euclidienne). Si p_i est le prototype qui représente l'exemple x_i au sens de la mesure de distorsion δ, alors la distorsion Δ minimisée par une méthode de quantification vectorielle s'écrit :

$$\Delta = E\left(\delta(x_i, p_i)\right)$$

Il existe de nombreuses méthodes capables de réaliser une quantification vectorielle dont on pourra trouver une bonne revue en se reportant à la thèse de Demartines (1994). Une fois que le dictionnaire qui minimise cette mesure est obtenu, l'espace représenté par chaque prototype peut se délimiter par le diagramme de Voronoï[2] (voir fig. 2.4).

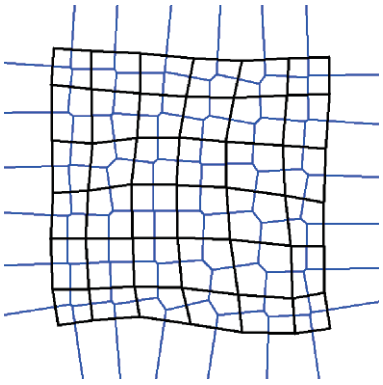


Fig. 2.4: Pavage de Voronoï (en bleu) associé aux modèles d'une carte apprise sur des données réparties uniformément dans un espace à 2 dimensions.

[2]Le diagramme de Voronoï est le dual géométrique de la triangulation de Delaunay dont la solution optimale dans le plan est obtenue avec une complexité en $O(n \log n)$ (Fortune, 1987)

Si l'on considère les modèles des cartes auto-organisatrices comme des prototypes, alors les cartes répondent aux critères de la quantification vectorielle. La distorsion est minimisée à travers la règle de modification des poids (eq. 2.1) qui rapproche les modèles des exemples qu'ils représentent, de sorte que les modèles deviennent les centres de gravité de ces exemples. Quant à la projection sur l'espace de la carte, elle est assurée par la fonction de voisinage de cette même règle de modification des poids qui assure que deux modèles voisins sur la carte soient sensibles aux régions qui leurs sont proches dans l'espace d'apprentissage. Autrement dit, le principe *winner-take-most* des cartes auto-organisatrices – qui engendre la modification de plusieurs modèles sur un même exemple – leur permet de réaliser une quantification vectorielle dans l'espace d'apprentissage en même temps que la projection dans un espace de dimension inférieure.

1.3 Cas d'utilisation

Grâce à leur capacité de synthèse et de visualisation, les cartes auto-organisatrices sont devenues un outil privilégié pour l'analyse exploratoire de données. Pour preuve, 7718 travaux basés sur les cartes auto-organisatrices ont été recensés dans la littérature depuis leur création[(3)]. En pratique, une raison récurrente du recours aux cartes auto-organisatrices est leur capacité de visualisation. Ainsi, Rousset et al. (2005) détaillent par des exemples comment les cartes peuvent servir à l'exploration de données sociologiques en s'appuyant sur les différentes manières de présenter visuellement une information par les cartes (nous revenons plus longuement sur ces stratégies en section 5, à la fois pour l'apprentissage supervisé et non-supervisé). Dans cet article, les auteurs illustrent également avec des données réelles (relatives à l'aspect de la peau) l'intérêt d'obtenir une projection non-linéaire en montrant comment, sur ces données, une projection linéaire comme celle de l'ACP ne réussit pas à conserver les proximités.

Les cartes ont également servi à la représentation de grands volumes de données. À ce titre, la réalisation la plus remarquable est certainement *WebSOM* (Lagus et al., 2004) qui permet la classification et la visualisation de grands volumes de documents à l'aide d'une carte. En pratique, *WebSOM* a été appliqué sur différents corpus, le plus gros comprenant près de 7 000 000 de résumés (appris par une carte de 1 002 240 modèles). Cette application des cartes auto-organisatrices est certainement l'une des meilleures illustrations de leur capacité de passage à l'échelle.

Enfin, notons que les cartes sont encore utilisées comme modèle du fonctionnement cognitif. Comme nous l'avons vu en section 1.1, les cartes auto-organisatrices se sont inspirées des cartes corticales qui représentent l'information sensorielle dans les cortex primaires. Newman et Polk (2007) se sont alors intéressés à l'émergence de la sémantique à partir

[3]La liste complète de ces travaux est disponible à l'adresse :
http ://www.cis.hut.fi/research/som-bibl/

de ces cartes sensorielles. Pour représenter les cartes sensorielles, Newmann et Polk ont décrit des stimuli visuels suivant 4 caractéristiques (couleur, forme, taille, surface, chacune étant représentée par un espace multi-dimensionnel) et ont ensuite entraîné une carte auto-organisatrice par caractéristique sur ces stimuli. Ils ont alors cherché à construire une carte qui révèle la sémantique de ces stimuli à partir de l'organisation des cartes sensorielles. Pour apprendre cette carte sémantique, ils ont ré-exprimé chaque stimuli dans l'espace de leurs coordonnées sur les cartes sensorielles. Par exemple, un stimuli représenté par les modèles de position (1,2), (2,7), (3,1), (5,9) sur les cartes sensorielles 1, 2, 3 et 4 sera représenté par le vecteur (1,2,2,7,3,1,5,9) lors de l'apprentissage de la carte sémantique. En appliquant cette stratégie, Newmann et Polk ont remarqué que la carte sémantique réussissait à retrouver la classification naturelle des stimuli qui n'était plus présente au niveau de chaque carte sensorielle (nous nous servons de cette stratégie dans la section 4.2 du chapitre 3).

Finalement, si les domaines d'application des cartes sont si vastes, c'est qu'elles sont aussi simples à mettre en œuvre qu'à interpréter, notamment au travers de la visualisation des données qu'elles offrent à l'utilisateur. En outre, la robustesse de leur algorithme facilite toutes sortes d'adaptations et leur rapidité les rend applicables à de grands volumes de données. Dans la section suivante, nous détaillons cet algorithme en rappelant quelques conseils sur son paramétrage et en donnant quelques-unes des adaptations proposées.

2 Algorithme

Dans cette section, nous nous intéressons à la mise en œuvre pratique des cartes auto-organisatrices. En partant de l'algorithme des cartes donné ci-dessous (algorithme 5), nous étudions leur complexité et détaillons quelques techniques d'optimisations proposées dans le livre de Kohonen (2001). Nous donnons également quelques indications sur le choix des hyper-paramètres et présentons trois adaptations, *Neural Gas*, *Growing Neural Gas* et *Parameter Less-SOM* (PL-SOM), dont l'objectif est justement de réduire le nombre de ces paramètres.

2.1 Complexité

Comme l'illustre l'algorithme 5, la complexité générale des cartes est linéaire en fonction des données. En effet, à chaque itération, un exemple est présenté à la carte et la première étape de l'algorithme détermine le modèle vainqueur qui représente le mieux cet exemple. Cette étape nécessite de calculer la distance entre l'exemple et chacun des modèles soit $m \times d$ itérations. Au cours d'une deuxième étape, le programme met à jour chaque composante des modèles dans le voisinage du vainqueur. Là encore, le programme doit parcourir chaque modèle pour déterminer s'il fait partie du voisinage avant d'appliquer la modification sur les composantes des modèles concernés. Dans le pire des cas, tous les modèles de la carte

Algorithme 5 : Cartes auto-organisatrices, version standard

entrées : $\mathbf{M}$, la matrice de taille $m \times d$ des modèles de la carte et $m_{n,i}$ la valeur du n^e modèle sur la i^e dimension. Les coordonnées sur la carte du i^e modèle de M sont $r_i = (i \mod l,\ i/l)$. $\mathbf{X}$, la matrice de taille $n \times d$ des exemples et $x_{n,i}$ la valeur du x^e exemple sur la i^e dimension. $\mathbf{n_{epochs}}$, $\alpha_\mathbf{0}$ et $\sigma_\mathbf{0}$ respectivement le nombre d'itérations de l'algorithme, le taux d'apprentissage initial et l'étendue initiale du voisinage modifié.

sorties : $\mathbf{m}$, la matrice des modèles auto-organisés sur X.

$e \leftarrow 0$;
pour $epoch \leftarrow 1$ **à** n_{epochs} **faire**
  pour $n \leftarrow 1$ **à** m **faire**
    si $d(m_n, x_e) < d_{\min}$ **alors**
      $c \leftarrow n$;
      $d_{\min} \leftarrow d(m_n, x_e)$
    fin-si
  fin-pour
  pour $n \leftarrow 1$ **à** m **faire**
    $N_v \leftarrow e^{-\frac{\|r_c - r_n\|}{\sigma^2}}$;
    si $N_v \neq 0$ **alors**
      pour $d \leftarrow 1$ **à** d **faire**
        $m_{n,i} \leftarrow m_{n,i} + \alpha \times N_v \times (m_{n,i} - x_{e,i})$;
      fin-pour
    fin-si
  fin-pour
  $\alpha \leftarrow \alpha_0 \times (1 - \frac{t}{n_{epochs}})$;
  $\sigma \leftarrow 1.0 + (\sigma_0 - 1.0) \times (1 - \frac{t}{n_{epochs}})$;
  $e \leftarrow (e < N - 1)?(e + 1) : 0$;
fin-pour
retourner m

sont modifiés, soit $m \times d$ opérations et dans le meilleur des cas, seul le modèle vainqueur est modifié, soit m opérations. Enfin, ces deux étapes sont répétées pour plusieurs parcours de la base d'exemples ($n_{epochs} = k \times n$, k le nombre de parcours). Au final, la complexité de l'algorithme est en $O(k \times n \times m \times d)$ donc bien linéaire en fonction du nombre d'exemples, de modèles et d'attributs.

2.2 Version batch

La première optimisation possible de l'algorithme 5 consiste à appliquer la mise à jour des modèles par lots d'exemples (*batch method*) plutôt qu'exemple par exemple (Kohonen, 2001). Cette version utilise la notion de région d'influence d'un modèle pour déterminer sa position. Comme l'illustre la figure 2.5, la région d'influence d'un modèle se définit comme les régions de Voronoï de ce modèle et de celles de ses voisins (ici au sens de la fonction $\sigma(t)$). À partir de cette définition, il est possible de faire deux constatations. La première est qu'à chaque parcours des exemples, un modèle i n'est modifié que par les exemples situés dans sa région d'influence. La deuxième est qu'à l'état d'équilibre (où les modèles ne sont plus modifiés), la position d'un modèle correspond à celle du barycentre de sa région d'influence.

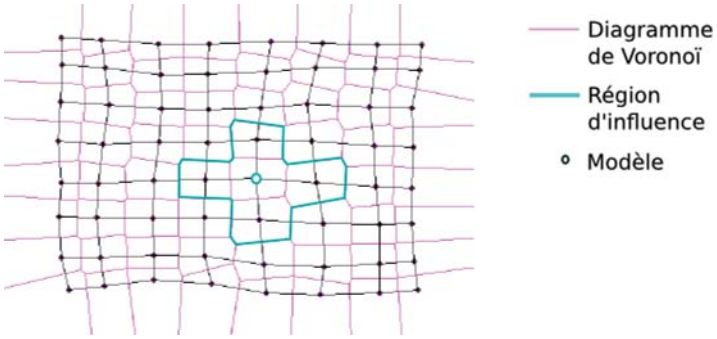


Fig. 2.5: Région d'influence (en bleu) d'un modèle avec $\sigma(t) = 1$ dans la version par lot des cartes auto-organisatrices. Les modèles sont représentés dans l'espace des attributs (2 dimensions).

D'après ces constatations, la position d'un modèle i est déterminée en deux étapes dans la version par lots. D'abord, les positions des modèles de la région d'influence de i sont approximées comme le barycentre de leur région de Voronoï. Ensuite, la position du modèle i est déterminée comme le barycentre des modèles de sa région d'influence (pondérés par le nombre d'exemples que chacun de ces modèles représente). La règle de modification des poids d'un modèle est ainsi donnée par :

$$m_i = \frac{\sum_j^M h_{ij} \times n_j \bar{x}_j}{\sum_j^M h_{ij} \times n_j} \tag{2.2}$$

où h_{ij} est une fonction de voisinage qui détermine si le modèle j fait partie de la région d'influence de i, n_j est le nombre d'exemples dans la région de Voronoï du modèle j et $\bar{x}_j$ est le barycentre des exemples de la région de Voronoï du modèle j.

Notons que l'inconvénient de cette version est qu'elle nécessite, pour fonctionner correctement, une organisation préalable des modèles (dont l'objectif est d'éviter des torsions de la carte, par exemple en réalisant préalablement une ACP, voir section 2.3). En revanche, cette version présente l'avantage de supprimer l'hyper-paramètre α du taux d'apprentissage ce qui permet, outre une mise en œuvre plus facile, d'assurer une meilleure convergence de la carte.

2.3 Hyper-paramètres

Le nombre de paramètres à fixer a priori est souvent reproché aux cartes autoorganisatrices. Ces paramètres sont le taux d'apprentissage initial α_0, l'étendue σ_0 du voisinage autour d'un modèle, la fonction de voisinage N_v, la topologie et la taille de la carte et enfin le choix de la position initiale des modèles. Ces paramètres ont une influence d'importance variable sur le résultat final de la carte et, à l'extrême, un mauvais choix de paramètres engendre une mauvaise projection de l'espace. Lorsque l'on cherche la meilleure organisation possible des données, il est donc conseillé de tester plusieurs valeurs possibles pour les paramètres de manière à trouver un paramétrage qui soit adapté. Éventuellement, cette recherche peut être automatisée grâce aux mesures de qualité présentées en section 4 (voir par exemple Polani (1997)).

En dehors de la recherche du meilleur ensemble de paramètres, il est possible de choisir raisonnablement certains paramètres suivant les conseils de différents auteurs (Demartines, 1994; Kohonen, 2001).

Initialisation des modèles. Le choix de la position initiale des modèles peut être déterminant sur la durée de l'apprentissage et ses chances de réussir. C'est pourquoi le choix au hasard de la position initiale des modèles dans l'espace d'apprentissage n'est pas conseillé. En effet, un problème possible de cette stratégie est d'attribuer des valeurs à certaines dimensions du modèle qui ne sont jamais observées sur les exemples. À la place, il est préférable d'initialiser les modèles directement à partir des exemples, ce qui assure que les valeurs qui leur sont attribuées soient réalistes. Cependant, cette stratégie n'organise pas les modèles dans l'espace en fonction de leur position sur la carte. En pratique, il est donc déconseillé de l'utiliser avec la version batch. Une autre solution consiste alors à fonder le choix de la position des modèles sur les deux premiers vecteurs propres de la matrice des corrélations entre attributs (c'est-à-dire les deux premiers axes factoriels d'une ACP). Ces deux vecteurs propres forment un plan de projection sur lequel il est possible d'ordonner

les modèles en fonction de leur position sur la carte et d'obtenir leur position dans l'espace d'apprentissage.

Taux d'apprentissage et étendue du voisinage. Suivant la méthode d'initialisation choisie, l'apprentissage ne se déroule pas de la même manière. En effet, si une organisation des modèles a été assurée lors de l'initialisation, l'apprentissage se concentre sur la convergence des modèles vers leur position finale. À l'inverse, sans organisation préalable des modèles, l'apprentissage commence par ré-organiser les modèles suivant leur position sur la carte. On distingue donc deux phases d'apprentissage, une phase (facultative) d'organisation des modèles et une phase de convergence. Au début de la phase d'organisation, les modèles sont souvent loin de leur position finale et de leur voisinage. Pour rétablir l'ordre, α_0 et σ_0 sont choisis suffisamment grands pour déplacer les modèles rapidement en direction de leur position finale. En revanche, lors de la phase de convergence, les modèles sont presque organisés et il faut les déplacer lentement de manière à trouver la position d'équilibre. Au cours de cette phase, α_0 et σ_0 sont choisis suffisamment petits pour ne pas briser l'organisation de la carte. Par exemple, si l est le nombre de modèles du plus petit côté de la carte, on pourra choisir un couple $(\alpha_0; \sigma_0) = (0,9; \frac{l}{2})$ pour la phase d'organisation et, au contraire, un couple $(\alpha_0; \sigma_0) = (0,1; 3)$ pour la phase de convergence.

Fonction de voisinage. En section 2, nous avons mentionné deux fonctions de voisinage N_v, l'une rectangulaire et l'autre gaussienne. Si la fonction rectangulaire est plus rapide à calculer (la fonction de voisinage est calculée $k \times N \times M$ fois dans la version standard), la fonction gaussienne assure, quant à elle, un résultat plus doux de l'organisation (Demartines, 1994).

Topologie et dimensions de la carte. Le choix des dimensions de la carte et de sa topologie est important (il est évident qu'une carte carrée aura du mal à rendre compte d'une distribution rectangulaire des données). Aussi, comme il est difficile de connaître *a priori* la distribution des données, il est conseillé de préférer des cartes rectangulaires (moins spécifiques que des cartes carrées) avec des topologies hexagonales qui sont capables de s'adapter à un plus grand nombre de distributions des données (Ultsch et Herrmann, 2005).

Par ailleurs, contrairement à l'intuition immédiate, il est également conseillé de choisir les dimensions de la carte, non pas en fonction du nombre d'exemples, mais plutôt de la dimensionnalité de l'espace d'apprentissage. En effet, plus l'espace d'apprentissage est grand, plus la projection de la carte est complexe. Dans ce cas, des modèles supplémentaires rajoutent une certaine liberté à la carte pour satisfaire la projection.

Le problème du paramétrage des cartes a conduit à proposer des adaptations de l'algorithme standard. La version par lots, présentée en section 2.2, supprime le paramètre α. Cependant, le paramètre le plus critique demeure la topologie de la carte. Nous présentons donc ici les méthodes *Neural Gas* et *Growing Neural Gas* qui visent à supprimer le choix de cette topologie. Nous présentons également la méthode *Parameter-Less SOM* qui vise spécifiquement à réduire le nombre de paramètres des cartes auto-organisatrices.

2.3.1 *Neural Gas*, *Growing Neural Gas*

Dans certains cas, une topologie adaptée à la forme de l'espace est difficile à obtenir, d'autant plus lorsqu'aucune information sur la distribution des données n'est disponible. Pour répondre à ce problème, Martinetz et Schulten (1991, 1994) ont proposé la méthode *Neural Gas* qui ne requiert aucune topologie mais simplement une définition du nombre de modèles utilisés. En contrepartie, cette méthode ne permet pas d'obtenir une projection de l'espace d'apprentissage mais simplement une quantification vectorielle doublée d'une structure de voisinage entre les modèles. L'apprentissage se déroule de manière très proche de celui réalisé par les cartes auto-organisatrices puisque la fonction de modification d'un modèle i lors de la présentation de l'exemple $x(t)$ est donnée par :

$$m_i(t+1) = m_i(t) + \alpha(t) N_v(t, k_i) \left(x(t) - m_i(t)\right) \tag{2.3}$$

Cette règle est quasi-similaire à celle de l'équation 2.1 des cartes. La différence principale est dans la fonction de voisinage qui n'est plus paramétrée par la distance sur la carte du modèle i au modèle vainqueur (il n'y a plus de carte) mais plutôt par le rang k de ce modèle, obtenu en classant les modèles par ordre croissant suivant leur distance à l'exemple. Une fois ce rang déterminé, la fonction de voisinage est donnée par :

$$N_v(t, k_i) = e^{-\frac{k_i}{\sigma_t}}$$

La structure de voisinage entre les modèles est constituée de liens maintenus au cours de l'apprentissage. La création d'un lien entre deux modèles, ainsi que leur mise à jour, a lieu à chaque fois que ces deux modèles sont ceux qui représentent le mieux l'exemple en cours d'apprentissage (soit les modèles de rang $k = 0$ et $k = 1$). Lorsqu'un lien n'a pas été mis à jour pendant un certain nombre d'itérations, il est supprimé. Ce nombre d'itérations est déterminé par une fonction $T(t)$, décroissante suivant t, afin de rendre compte de la plus grande stabilité des liens attendue en fin d'apprentissage.

Si l'algorithme *Neural Gas* résout le problème de la topologie de la carte, il demande encore de connaître le nombre de modèles utilisés. Pour faire coller au mieux la quantification vectorielle aux données, Fritzke (1994, 1995) proposent la méthode *Growing Neural Gas* qui modifie *Neural Gas* pour pouvoir l'utiliser sans choisir au préalable le nombre de modèles. Ceux-ci sont ajoutés au cours de l'apprentissage dans les régions de l'espace

où la représentation est la plus mauvaise. Cependant, cet algorithme est notablement plus compliqué que le précédent.

Ainsi, pour chaque modèle, cet algorithme associe à la fois des connections aux autres modèles, un âge à chacune de ces connections et une mesure de qualité qui évalue la projection de ce modèle. À chaque itération, où un nouvel exemple est présenté, cet algorithme recherche les deux modèles qui lui sont les plus proches et crée une connexion d'âge 0 entre eux. Ensuite, tous les modèles connectés au modèle vainqueur sont rapprochés de l'exemple et la qualité de projection associée à ce vainqueur est diminuée d'autant plus que sa distance à l'exemple est importante. Enfin, l'âge des connexions partant du modèle vainqueur est incrémenté de 1 et toutes les connexions (pas seulement celles mises à jour précédemment) dont l'âge dépasse un certain seuil sont supprimées. Ces étapes assurent l'organisation des modèles existants. La création d'un nouveau modèle a lieu, quant à elle, à intervalle régulier de λ itérations. À ce moment là, un modèle est créé comme le barycentre des deux modèles dont la qualité est la plus faible. La qualité de ces deux modèles est alors augmentée d'un taux ϵ_1 et la qualité des autres modèles d'un taux ϵ_2.

Comme nous l'avons fait remarquer plus haut, ce qui permet à ces deux algorithmes de supprimer le recours à une topologie des modèles dans un espace de petite dimension est qu'aucun des deux ne réalise de projection. Leur intérêt réside donc essentiellement dans la quantification vectorielle des données, quantification associée à une structure de voisinage (les liens entre modèles).

2.3.2 *Parameter-Less SOM*

La méthode *PL-SOM* (Berglund et Sitte, 2006), au contraire, conserve la topologie des modèles et cherche plutôt à simplifier les cartes en supprimant les paramètres relatifs au taux d'apprentissage α et à l'étendue du voisinage σ. Dans l'algorithme original, ces deux paramètres décroissent avec le temps de manière à relativiser la modification d'un modèle en fonction du moment où cette mise à jour a lieu. Plutôt que d'utiliser le temps pour déterminer la valeur de ces paramètres, les auteurs de *PL-SOM* remplacent l'indicateur temporel (le numéro de l'itération) par un indicateur d'erreur fonction de la distance entre l'exemple à apprendre et le modèle qui doit le représenter :

$$\epsilon(t) = \frac{d(x(t), m_c(t))}{r(t)}$$

$$\begin{aligned} r(t) &= \max\left(d(x(t), m_c(t)),\ r(t-1)\right) \\ r(0) &= d(x(0), m_c(0)) \end{aligned}$$

Cet indicateur va permettre à la carte de s'adapter automatiquement à l'exemple. Si l'exemple est mal représenté sur la carte, $\epsilon(t)$ est grand et les modèles seront fortement

modifiés. A l'inverse, si la carte représente convenablement l'exemple, $\epsilon(t)$ est petit et peu de modifications seront engendrées. L'étendue du voisinage σ est également exprimée en fonction de $\epsilon(t)$:

$$\sigma(\epsilon(t)) = \beta \times \ln(\epsilon(t)(e+1)+1)$$

avec β un paramètre du modèle et e le nombre d'Euler. L'étendue du voisinage est ensuite utilisée dans une fonction de voisinage $N_v(t)$ similaire à celle de l'équation 1.2. De cette manière, le voisinage du modèle vainqueur modifié à chaque itération sera d'autant plus important que la distance de l'exemple à la carte sera grande.

Cet algorithme ne supprime donc pas complètement les valeurs de α et σ mais évite à l'utilisateur des cartes de devoir les fixer *a priori*. Leurs valeurs sont déterminées automatiquement en fonction de la carte plutôt qu'en référence à une valeur initiale. Grâce à ce mécanisme, il n'est plus nécessaire de définir une période d'apprentissage au-delà de laquelle les cartes n'apprennent plus. Au contraire, elles sont capables de s'adapter à de nouveaux exemples, même si ceux-ci arrivent en fin d'apprentissage.

3 Adaptation à l'apprentissage supervisé

Les sections précédentes ont montré comment les cartes auto-organisatrices réalisent un apprentissage non-supervisé des données qui conduit à la fois à une quantification vectorielle et à une projection. Leur simplicité et leur rapidité de fonctionnement ont amené plusieurs auteurs à proposer des adaptations de cet algorithme pour l'apprentissage supervisé. Nous présentons ici trois familles d'adaptation, l'apprentissage par quantification vectorielle (*Learning Vector Quantization, LVQ*, section 3.1), le modèle LASSO (section 3.2) et les méthodes de type cartes étiquetées (section 3.3). Ces familles se différencient sur la combinaison de deux critères. Le premier de ces critères concerne leur capacité de projection et le second a trait à l'indépendance entre la position des modèles (que l'algorithme doit déterminer) et leur étiquette. Après une présentation succincte des différentes familles, la section 3.4 propose un récapitulatif de l'implémentation de ces critères par chaque famille.

3.1 Apprentissage par quantification vectorielle

L'apprentissage par quantification vectorielle (ou *LVQ* pour *Learning Vector Quantization*) est la première adaptation des cartes auto-organisatrices à l'apprentissage supervisé (Kohonen, 1990, 2001). Cette adaptation vise à tirer parti des capacités de quantification vectorielle des cartes dans le domaine de l'apprentissage supervisé. Ainsi, elle permet d'obtenir une quantification qui associe à chaque modèle une étiquette représentative de sa

région de quantification. De cette manière, la prédiction d'un nouvel exemple est assurée simplement en lui associant l'étiquette du modèle qui lui est le plus proche. Comme seul l'aspect quantification vectorielle des cartes est adapté, les modèles des *LVQ* ne sont pas reliés par une structure de voisinage dans un espace de projection (autrement dit, il n'existe pas de carte).

En pratique, les *LVQ* sont composés d'un ensemble de modèles dans l'espace d'apprentissage, chacun affecté à l'une des étiquettes à apprendre. À l'instar des cartes, un exemple x est représenté par le modèle m_c si c est tel que :

$$c = argmin_i(d(x, m_i))$$

À chaque itération t de l'algorithme, un exemple $x(t)$ est présenté à l'algorithme. À partir de celui-ci, les modèles sont modifiés. Il existe 3 versions des *LVQ* qui utilisent chacune une règle différente de modification des modèles. Dans la première version *LVQ1*, seul le modèle vainqueur m_c qui est le plus proche de l'exemple est modifié conformément à la règle suivante :

$$m_c(t+1) = \begin{cases} m_c(t) + \alpha(t) \times (x(t) - m_c(t)) & \text{si le modèle a la même étiquette} \\ & \text{que l'exemple (cas 1);} \\ m_c(t) - \alpha(t) \times (x(t) - m_c(t)) & \text{sinon (cas 2).} \end{cases}$$

Cette règle est très proche de la règle de modification des poids des cartes auto-organisatrices, excepté sur deux points. D'abord, elle ne définit aucune notion de voisinage (pas de fonction $N_v(t)$) et ensuite, elle intègre la classe de l'exemple pour déterminer si le modèle doit être rapproché (cas 1) ou éloigné de l'exemple (cas 2). Ainsi, la position des modèles dans l'espace d'apprentissage est déterminée en fonction de la classe qui leur est affectée en début d'apprentissage.

Dans une deuxième version *LVQ2*, Kohonen propose de modifier les deux modèles les plus proches de l'exemple $x(t)$ à condition que l'un d'eux (noté m_j) soit de même classe que l'exemple et l'autre (noté m_i) de classe différente. Par ailleurs, pour que la modification ait lieu, $x(t)$ doit se situer dans une certaine fenêtre w autour des modèles de telle sorte que :

$$\min\left(\frac{d_i}{d_j}, \frac{d_j}{d_i}\right) > s, \text{ avec } s = \frac{1-w}{1+w}$$

avec $d_i = d\left(x(t), m_i(t)\right)$. Sous ces conditions, la règle de modification des modèles est alors :

$$\begin{aligned} m_i(t+1) &= m_i(t) - \alpha(t) \times (x(t) - m_i(t)) \\ m_j(t+1) &= m_j(t) + \alpha(t) \times (x(t) - m_j(t)) \end{aligned}$$

Enfin, Kohonen affine encore la version précédente en rajoutant une règle de modification pour les cas où m_i, m_j et x sont de même classe. Dans ces cas, Kohonen propose d'appliquer la règle suivante à la fois à m_i et m_j (*LVQ3*) :

$$m_k = m_k(t) - \epsilon\alpha(t) \times (x(t) - m_k(t))$$

avec ϵ, un hyper-paramètre de l'algorithme jouant sur l'importance des modifications. Cette règle permet de rapprocher très faiblement les modèles de l'exemple, même dans le cas où les modèles semblent bien représenter cet exemple (*ie* les deux modèles qui lui sont les plus proches sont de même classe).

3.2 Modèle LASSO

Le modèle LASSO[4] (Midenet et Grumbach, 1994) cherche, au contraire des *LVQ*, à adapter les cartes à l'apprentissage supervisé, aussi bien en ce qui concerne la quantification vectorielle que la projection et la visualisation.

Pour ce faire, ces auteurs proposent d'étendre l'espace sur lequel les cartes réalisent l'apprentissage – et donc la projection – à l'aide des étiquettes à prédire de manière à garder intacte la phase d'apprentissage des cartes. L'espace d'apprentissage est ainsi composé de deux espaces distincts, celui des attributs d'une part et celui des classes d'autre part. Ce deuxième espace représente les étiquettes sous une forme disjonctive complète. Pour un exemple donné, chaque dimension de cet espace représente ainsi la probabilité qu'il appartienne à une étiquette particulière. Chacune de ces dimensions prend donc pour valeur 1 si l'étiquette qu'elle représente correspond à celle de l'exemple et 0 si l'étiquette ne correspond pas. Par exemple, en utilisant cette représentation avec des exemples à apprendre décrits par 8 attributs et répartis en 4 classes, l'espace d'apprentissage se compose de 12 dimensions, puisqu'aux 8 dimensions relatives aux attributs sont ajoutées une dimension par étiquette. Ainsi, dans l'espace des classes, un exemple ayant la 2^e classe est représenté par le vecteur (0100), chaque composante de ce vecteur représentant la probabilité *a priori* que l'exemple ait l'étiquette concernée (figure 2.6.a).

À partir de ce nouvel espace, l'apprentissage de la carte se déroule de manière standard. Cependant, au terme de celui-ci, la carte est prête pour associer à un nouvel exemple son étiquette. Pour attribuer cette étiquette, l'algorithme recherche d'abord le modèle vainqueur sur la carte. Cette recherche a recours uniquement à l'espace des attributs pour calculer la distance entre un modèle et l'exemple. La position dans l'espace des classes du modèle vainqueur correspond alors aux probabilités *a posteriori* de prédire chaque étiquette pour cet exemple, l'étiquette ayant la plus grande probabilité étant attribuée à l'exemple (figure 2.6.b).

Cette stratégie permet de conserver la projection de l'espace d'apprentissage par les cartes et donc leurs capacités de visualisation. En particulier, après un apprentissage par LASSO, Fessant et Aknin (2000) visualisent une carte par étiquette où chaque modèle est colorié en fonction de sa probabilité de prédire l'étiquette d'intérêt. Visuellement, ils peuvent ainsi juger de la pertinence de la représentation obtenue en vérifiant que des

[4] à ne pas confondre avec la méthode LASSO de sélection de variables en régression linéaire proposée par Tibshirani (1996).

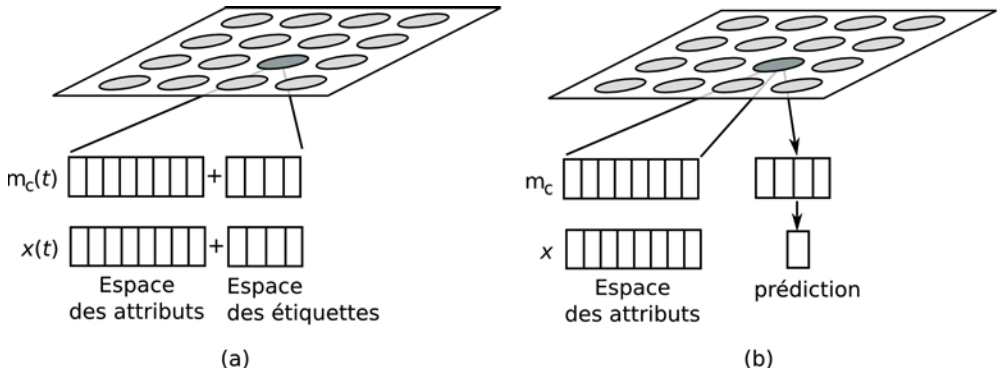


FIG. 2.6: Les modèles de LASSO sont des vecteurs dans un espace composé de celui des attributs et de celui des classes. (a) En phase d'apprentissage, le modèle vainqueur est recherché dans cet espace. (b) En phase de prédiction, la recherche du modèle vainqueur ne se fonde plus que sur la partie de l'espace qui correspond aux attributs. L'espace des classes détermine alors l'étiquette de la prédiction.

modèles voisins sur la carte représentent bien des étiquettes identiques. Par ailleurs, comme pour les *LVQ*, la position des modèles dans l'espace des attributs est déterminée en fonction de l'étiquette des exemples puisque la recherche du vainqueur lors de l'apprentissage est réalisée en partie sur cette étiquette.

3.3 Cartes étiquetées

Alors que les deux premières familles présentées transforment la phase d'apprentissage des cartes auto-organisatrices pour obtenir des algorithmes d'apprentissage supervisé, les méthodes de type cartes étiquetées dotent les cartes de capacité de prédiction sans modifier ni leur phase ni leur espace d'apprentissage.

Ces méthodes préparent la prédiction au terme de l'apprentissage des exemples par la carte. Elles procèdent alors à une phase d'étiquetage des modèles en fonction des exemples qu'ils représentent. Trois cas sont alors possibles pour un modèle :

- les exemples représentés par le modèle sont majoritairement d'une même classe. Le modèle est alors étiqueté en fonction de cette classe,
- les exemples représentés par le modèle ne sont pas majoritairement d'une même classe. Le modèle est alors étiqueté comme modèle ambigu,
- le modèle ne représente aucun exemple. Il est alors étiqueté comme vide.

Après cette phase d'étiquetage, la carte est prête à être utilisée en prédiction. Il existe alors trois méthodes différentes pour associer son étiquette à un nouvel exemple, Kohonen-KNN (pour *k-Nearest Neighbor*), Kohonen-WI (pour *Weight Interpretation*) et Kohonen-Opt. Ces

trois méthodes divergent sur la fonction de prédiction en charge d'associer une étiquette pour un exemple représenté par un modèle indéterminé, c'est à dire vide ou ambigu.

3.3.1 Kohonen-KNN et Kohonen-WI

La méthode Kohonen-KNN proposé par Zupan et al. (1994) fonde la prédiction d'un modèle indéterminé sur l'étiquette majoritaire portée par ses voisins directs sur la carte. Song et Hopke (1996) expriment deux remarques sur cette stratégie. D'abord, elle perd les informations de voisinage réel dans l'espace d'apprentissage pour lui substituer le voisinage issu de la prédiction. Ensuite, elle conduit dans certains cas à de nouvelles indéterminations si aucune étiquette n'est majoritaire dans le voisinage du modèle. Afin d'éliminer ce problème, ils proposent la fonction Kohonen-WI qui associe comme étiquette à un modèle indéterminé, celle du modèle étiqueté qui lui est le plus proche dans l'espace d'apprentissage. En pratique, ils montrent que cette stratégie donne de meilleurs résultats que Kohonen-KNN sur leur base de données d'intérêt.

3.3.2 Kohonen-Opt

La stratégie Kohonen-WI a l'avantage d'être relativement simple puisqu'il suffit de connaître l'étiquette du plus proche voisin pour prendre une décision. La contrepartie de cette simplicité est la perte de deux types d'informations, celles relatives au voisinage sur la carte et celles relatives aux modèles ambigus. Pour prendre en compte l'ensemble de ces informations, nous avons suggéré Kohonen-Opt (Prudhomme et Lallich, 2005b,a). Lors de la prédiction d'un exemple par un modèle d'indice c indéterminé (ou vide), nous proposons d'associer une mesure de vraisemblance à chaque étiquette. Cette mesure intègre deux types d'informations : celles relatives à la présence de l'étiquette dans le voisinage du modèle c (par la suite, on note V_{m_c} l'ensemble des indices des modèles voisins de m_c) et celles relatives à la distance de chaque voisin au modèle c. L'étiquette ayant la plus grande vraisemblance est alors retenue pour prédire l'exemple. Pour un modèle d'indice i voisin du modèle m_c et une étiquette y, on commence par calculer une pondération $p_{m_i}(y)$. Cette pondération est fonction de la proportion d'exemples représentés par ce modèle portant l'étiquette y, proportion notée $q_{m_i}(y)$ et de la distance normalisée du modèle i à l'exemple x à prédire soit :

$$p_{m_i}(y) = \left(1 - \frac{d(x, m_i)}{\max_{i \in V_{m_c}}(d(x, m_i))}\right) * q_{m_i}(y)$$

Cette pondération prend des valeurs proches de 0 pour des modèles éloignés qui représentent peu d'exemples d'étiquette y. Elle tend vers 1 pour les modèles les plus proches de l'exemple qui représente une grande proportion d'exemples d'étiquette y. L'étiquette c retenue pour

la prédiction est alors définie par :

$$\begin{cases} c &= \max_h(P(x,y)) \\ P(x,y) &= \sum_{i \in V_{m_c}} p_{m_i}(y) \ . \end{cases}$$

Afin de valider Kohonen-Opt, nous l'avons testé face à Kohonen-WI sur plusieurs jeux de données. Les résultats sont détaillés en section 6.

3.3.3 Hyper-paramètres

Le paramétrage des cartes auto-organisatrices en apprentissage supervisé diffère légèrement de celui donné en section 2.3. En particulier, le choix des dimensions de la carte ne suit pas les mêmes critères. Ici, ces dimensions doivent être suffisamment grandes pour garantir une bonne représentation des zones frontières entre deux classes. En effet, comme l'apprentissage n'utilise pas l'étiquette pour déterminer la position des modèles, ces zones ne sont pas représentées avec une grande précision. Par exemple, Song et Hopke (1996) utilisent une carte de taille 15×15 pour l'apprentissage de 250 exemples décrits par 8 attributs. Cependant, plus la carte est grande, plus le nombre de modèles vides augmente – et donc plus le choix d'une fonction de prédiction adaptée devient important. Par ailleurs, Song et Hopke (1996) remarquent que le nombre de modèles vides peut être modulé suivant la valeur minimale de σ, l'étendue du voisinage. Ainsi, finir l'apprentissage avec un voisinage de 1 réduit notablement la présence de modèles vides (l'apprentissage classique se termine généralement avec un voisinage de 0, c'est-à-dire que seul le modèle vainqueur est modifié).

3.4 Récapitulatif

Finalement, parmi les 3 familles de méthodes proposées, seules les méthodes de type cartes étiquetées répondent aux critères représentationnels que nous nous sommes fixés en section 1 (le tableau 2.1 résume ces critères pour les 3 familles).

En effet, les *LVQ* sont essentiellement orientés vers la prédiction et ne projettent pas les exemples dans un espace visualisable. Par ailleurs, que ce soient les *LVQ* ou le modèle LASSO, la position des modèles à la fin de l'apprentissage n'est pas indépendante de leur étiquette puisque celle-ci sert à la mise à jour des poids. Aussi, il est impossible de visualiser le lien entre la position des exemples et leur étiquette à l'aide de ces deux approches.

Les méthodes de type cartes étiquetées, en revanche, conservent toutes les propriétés des cartes auto-organisatrices dont elles reprennent l'apprentissage. Elles restent donc adaptées à la représentation des données en apprentissage supervisé. Par la suite, l'apprentissage supervisé des cartes fera toujours référence à ces méthodes.

	Méthodes		
	LVQ	**LASSO**	**Cartes étiquetées**
Projection	non	oui	*oui*
Indépendance modèle-classe	non	non	*oui*
Représentation	non	non	**oui**

TAB. 2.1: Tableau récapitulatif des propriétés des différentes familles d'adaptation des cartes à l'apprentissage supervisé.

4 Validation statistique

Au terme de l'apprentissage des cartes auto-organisatrices, qu'il soit non-supervisé ou supervisé (type modèle LASSO ou carte étiquetée), on obtient un ensemble de modèles qui ont une position à la fois dans l'espace d'apprentissage et sur la carte, ce qui permet de projeter les exemples de l'un vers l'autre. Comme nous l'avons mentionné en section 2.3, la qualité de cette projection est liée au paramétrage initial de la carte. Aussi, peut-il être nécessaire de la valider avant de s'en servir pour exploiter les données d'apprentissage. En outre, même avec une bonne projection, l'algorithme des cartes reste stochastique et peut engendrer des résultats différents lors de deux apprentissages successifs sur les même données. Dans certains cas, il convient donc de s'assurer que l'interprétation est bien issue des données et non le fait du hasard.

Dans la suite, nous présentons plusieurs travaux en lien avec la validation statistique des cartes auto-organisatrices, d'abord en apprentissage non-supervisé, puis en apprentissage supervisé. Dans ce dernier cas, nous nous sommes uniquement intéressé à la validation d'un apprentissage de type carte étiquetée (puisque nous souhaitons l'utiliser pour la représentation des données) pour lequel nous avons proposé une mesure statistique.

4.1 Apprentissage non-supervisé

Nous avons distingué trois sortes de mesures pour valider l'apprentissage non-supervisé d'une carte auto-organisatrice : les mesures de distorsion, les mesures de stabilité du voisinage et enfin les mesures de préservation de la topologie.

4.1.1 Mesure de distorsion.

Comme nous l'avons mentionné en section 1.2, les cartes auto-organisatrices font partie des méthodes de quantification vectorielle. À ce titre, elles minimisent la distorsion qui résulte de la quantification de l'espace d'apprentissage par les modèles de la carte. Pour une carte, cette distorsion s'écrit :

$$\Delta = \sum_{i}^{M} \sum_{x_j \in C_i} \|x_j - m_i\|^2 \ .$$

avec C_i la région de Voronoï du modèle i. Cette mesure peut servir directement comme indicateur de la validité de l'apprentissage. Grâce à cette mesure, Polani et Uthmann (1993) guident un algorithme génétique dont chaque phénotype encode une topologie particulière de carte afin de déterminer la meilleure topologie possible pour représenter les données d'apprentissage.

Bodt et al. (2002) se sont intéressés aux variations de cette mesure entre plusieurs apprentissages, notamment pour étudier son risque de converger vers des minima locaux. Pour cela, ils ont estimé le cœfficient de variation de la mesure de distorsion en répétant 100 fois l'apprentissage d'une carte sur les données d'apprentissage (méthode de Monte-Carlo). Ils ont comparé ce cœfficient de variation suivant trois cas d'initialisation des modèles à chaque itération de la méthode de Monte-Carlo : initialisation au hasard, initialisation à partir du résultat de la première itération, initialisation à partir du résultat de la première itération avec néanmoins quelques interversions des modèles.

En menant des expérimentations sur trois jeux de données, dont un réel, ils ont constaté que le cœfficient de la mesure de distorsion reste stable quel que soit le cas dans lequel il a été calculé. Ce cœfficient ne semble donc pas affecté par le mode d'initialisation de la carte.

Dans un deuxième temps, ils ont cherché à déterminer le nombre de modèles des cartes à l'aide de ce cœfficient de variation. Cependant, dans ce cas, ce cœfficient a été estimé à partir d'échantillons *bootstraps* des données d'apprentissage (et seulement avec une initialisation au hasard des modèles). À l'arrivée, les variations de la mesure de distorsion semblent d'autant plus importantes que le nombre de modèles est sur-dimensionné par rapport aux classes naturelles des données.

4.1.2 Force du lien de voisinage.

Dans le même article que précédemment – et avec la même démarche méthodologique – les auteurs se sont intéressés à la stabilité des relations de voisinage entre deux exemples x_i et x_j sur la carte (Bodt et al., 2002). Ils ont proposé comme mesure de cette stabilité :

$$stab_{i,j}(r) = \frac{\sum_{b=1}^{B} \Lambda_{i,j}^{b}(r)}{B} \ .$$

avec $\Lambda^b_{i,j}(r) = 1$ si i et j sont voisins dans un rayon de r sur la b^{e} carte des B *bootstraps*. Cette mesure prend pour valeur 1 (resp. 0) pour chaque paire d'exemples (x_i, x_j) s'ils sont voisins (resp. non voisins) sur la carte. En outre, assimilant le comportement de $stab_{i,j}$ à celui d'une variable binomiale sous l'hypothèse nulle où le voisinage entre x_i et x_j est déterminé de manière aléatoire, les auteurs sont capables d'associer une *p-value* à cette mesure. Ainsi, sur des données réelles, cette mesure permet de déterminer la force d'un lien entre deux exemples et la probabilité que ce lien ne soit pas dû au hasard. Cet outil donne ainsi à l'utilisateur l'assurance que l'interprétation qu'il tire de sa carte est bien issue des données et non le fait d'une convergence vers un minima local lors de l'apprentissage.

4.1.3 Préservation de la topologie.

La mesure de distorsion valide la quantification vectorielle des cartes mais ne s'intéresse pas à la qualité de la projection de l'espace d'apprentissage. Or, si cette projection est de bonne qualité, elle permet d'accéder à la topologie de l'espace d'apprentissage à travers celui de la carte (ce qui assure, entre autres, une visualisation adéquate des données). Bauer et al. (1999) rappellent que la topologie est fonction de la similarité entre les points de l'espace. Ils catégorisent la préservation de la topologie (ou topographie) suivant que la notion de similarité s'assimile à :

- une structure de voisinage entre les points et les modèles,
- un ordonnancement des distances entre les points et les modèles,
- la conservation des relations de distances entre les points et les modèles.

Ces différentes notions conduisent chacune à des définitions différentes de la préservation de la topologie par les cartes (voir Polani (1997); Bauer et al. (1999) pour des revues sur ces mesures). À titre d'illustration, nous présentons ici deux mesures qui se rapportent aux deux premières acceptations données pour la notion de similarité, respectivement la fonction topographique (Villmann et al., 1994) et le produit topographique (Bauer et Pawelzik, 1992) par lequel nous commençons.

Produit topographique. Ce produit, proposé par Bauer et Pawelzik (1992), cherche à rendre compte de la préservation de la topologie au sens d'une conservation de l'ordonnancement des distances entre les deux espaces. Pour cela, ces auteurs définissent d'abord deux variables $n^A_k(i)$ et $n^V_k(i)$, qui représentent l'indice du k^{e} voisin du modèle i respectivement dans l'espace d'apprentissage (exposant A) et dans celui de la carte (exposant V). Ils considèrent alors les rapports :

$$Q_1(i,k) = \frac{d(w_i, w_{n^A_k(i)})}{d(w_i, w_{n^V_k(i)})} \text{ et } Q_2(i,k) = \frac{d(i, n^A_k(i))}{d(i, n^V_k(i))} \ .$$

Ces rapports sont choisis tels que $Q_1(i,k) = Q_2(i,k) = 1$ si les distances entre le k^{e} voisin de i au sens du voisinage de la carte et au sens du voisinage de l'espace d'apprentissage

correspondent, distances mesurées respectivement dans l'espace d'apprentissage (Q_1) et dans l'espace de la carte (Q_2). Ils s'intéressent alors au produit de ces rapports (qu'ils normalisent puis moyennent sur l'ensemble des modèles) pour obtenir le produit topologique :

$$P = \frac{1}{n^2 - n} \sum_{i}^{m} \sum_{j}^{m-1} \log \left(\prod_{k}^{j} Q_1(i,k) \times Q_2(i,k) \right)^{\frac{1}{2j}} \tag{2.4}$$

Suivant cette mesure, lorsque la préservation de la topologie est de bonne qualité $P \approx 0$, $P < 0$ lorsque l'espace de la carte est trop grand et $P > 0$ sinon (Bauer et al., 1999).

Fonction topographique. Contrairement au produit topographique, la fonction topographique contrôle la préservation de la topologie en s'intéressant à la conservation des distances selon les structures de voisinage des points. Cette mesure commence donc par la construction, dans l'espace d'apprentissage, d'un graphe G des modèles selon la méthode des polygones de Voronoï (où deux points sont reliés par une arête s'ils partagent un coté commun sur leur polygone). La distance d_G entre deux modèles correspond alors à leur distance sur le graphe. Ces distances servent ensuite à évaluer la déviation entre les voisinages des deux espaces, à travers deux fonctions qui calculent chacune pour un modèle i le nombre de ses voisins dans un espace (resp. l'espace d'apprentissage et l'espace de la carte), qui ne le sont plus dans l'autre (resp. l'espace de la carte et l'espace d'apprentissage) à une distance supérieure à k, soit $f_i(k)$ et $f_i(-k)$ telles que

$$f_i(k) = \#\{j \mid \|r_i - r_j\|_{max} > k \,;\; d_G(w_i, w_j) = 1\}$$

$$f_i(-k) = \#\{j \mid \|r_i - r_j\| = 1;\; d_G(w_i, w_j) > k\}$$

En définissant $\Phi(k)$ comme la moyenne de ces fonctions sur l'ensemble des modèles, ils obtiennent comme fonction topographique :

$$\Phi = \Phi(1) - \Phi(-1)$$

Tout comme le produit topographique, cette fonction vaut 0 lorsque la topographie est parfaitement préservée. Mais, à l'inserve du produit, elle prend des valeurs positives lorsque l'espace de la carte est trop petit, et négatives sinon.

Bauer et al. (1999) évaluent ces fonctions, et d'autres, sur différents jeux de données et montrent qu'elles sont capables de faire des approximations relativement correctes de la dimension intrinsèque de données réelles. Ils notent également que l'utilisation d'une structure de voisinage pour calculer les distances dans l'espace d'apprentissage (comme c'est le cas pour la fonction topographique) n'apporte pas de réel avantage, notamment à cause de l'éparsité des données en grandes dimensions qui rend peu opérante cette structure de voisinage.

4.2 Apprentissage supervisé

Un apprentissage supervisé réalisé à l'aide d'une carte étiquetée se déroule en deux étapes, à savoir un apprentissage non-supervisé de la carte suivi de l'étiquetage de chaque modèle. La conséquence directe est que la qualité de cet apprentissage supervisé dépend de deux facteurs. Le premier concerne la qualité de l'apprentissage topologique de la carte, c'est-à-dire la qualité de la projection de l'espace d'apprentissage sur l'espace de la carte. Les différentes mesures que nous venons de présenter permettent de valider ce premier facteur. Le deuxième facteur à jouer un rôle déterminant dans la qualité de la prédiction est la représentation par l'espace de la carte (et, en supposant que la projection de la carte soit de bonne qualité, par l'espace d'apprentissage) des différentes valeurs de l'étiquette.

Pour valider ce deuxième point, nous proposons d'adapter une mesure de validation de la représentation des graphes de voisinage en apprentissage supervisé à la représentation obtenue par les cartes étiquetées (Prudhomme et Lallich (2005b) ainsi que Prudhomme et Lallich (2005a)). Cette mesure est issue des outils de l'auto-corrélation spatiale et nous en formulons trois versions pour les cartes étiquetées.

4.2.1 Statistique d'auto-corrélation spatiale

L'auto-corrélation spatiale s'intéresse à la répartition d'une étiquette sur des sites répartis dans un espace à 2 ou 3 dimensions (Cliff et Ord, 1981). Il existe une auto-corrélation spatiale positive entre ces sites si des sites voisins ont tendance à avoir une même étiquette, auto-corrélation négative s'ils ont tendance à avoir une étiquette différente et nulle si leurs étiquettes ne dépendent pas les uns des autres. L'hypothèse d'absence d'auto-corrélation H_0 peut être testée à l'aide de la statistique *cross-product* de Mantel (1967); Getis (1991) :

$$\Gamma = \sum_{i}^{n} \sum_{j}^{n} W_{ij} U_{ij}$$

où, pour les n sites, W_{ij} mesure la proximité entre les sites i et j dans l'espace et U_{ij} la proximité entre ces sites au sens de l'étiquette.

4.2.2 Statistique du poids des arêtes coupées

Dans le contexte de données supervisées, Sebban et Zighed (1996); Zighed et al. (2001) ont cherché à connaître a priori la séparabilité des classes dans l'espace d'apprentissage à partir d'un graphe de voisinage des exemples où chaque sommet est étiqueté suivant la classe à prédire. Lallich (2002) ramène ce problème à un test d'auto-corrélation spatiale entre la position des exemples dans l'espace d'apprentissage et leur étiquette. Il définit ainsi

une statistique *cross-product* J du poids des arêtes coupées tel que :

$$J = \sum_{i=1}^{n} \sum_{j=1}^{n} W_{ij} U_{ij} \ . \tag{2.5}$$

Chaque terme W_{ij} de la matrice $W_{n \times n}$ représente le poids de l'arête entre les exemples i et j de sorte que plus les exemples sont proches, plus ce poids est important :

$$W_{ij} = \begin{cases} (1 + d(x_i, x_j))^{-1} & \text{si } i \text{ et } j \text{ sont voisins,} \\ 0 & \text{sinon .} \end{cases}$$

La matrice $U_{N \times N}$ mesure les proximités au sens des étiquettes et son terme U_{ij} vaut 1 si les exemples i et j sont d'étiquettes différentes, 0 sinon. Ainsi, le produit $W_{ij}U_{ij}$ est différent de 0 seulement si i et j sont deux exemples d'étiquettes différentes reliés par une arête. Pour ne conserver sur le graphe que des relations de voisinage entre exemples de même étiquette, ces arêtes doivent être coupées. Au final, cette statistique teste donc la force du lien qui peut exister entre des exemples d'étiquettes différentes en fonction des distances entre ces exemples. Enfin, cette statistique peut être normalisée en la divisant par le poids total de toutes les arêtes $S_0 = \sum_i \sum_j W_{ij}$.

Pour appliquer un test à cette statistique, Lallich (2002) considère l'hypothèse nulle où les exemples sont étiquetés indépendamment suivant une même distribution de probabilité $(\pi_r)_r$ où π_r désigne la proportion d'exemples d'étiquette r. La *p-value* unilatérale à gauche de J (*ie* la probabilité d'obtenir sous H_0 une valeur de J aussi petite que celle observée) s'obtient alors par approximation normale (Cliff et Ord, 1981) (on se reportera à Lallich (2002) pour connaître les expressions de $\mu = E(J|H_0)$ et $\sigma^2 = Var(J|H_0)$ nécessaires à cette approximation).

Zighed et al. (2001) ont vérifié le bien-fondé de cette statistique en comparant sa version normalisée avec le taux d'erreur en généralisation obtenu par différentes méthodes. Ils ont ainsi pu montrer une corrélation forte entre la statistique du poids des arêtes coupées et le taux d'erreur en généralisation. Par ailleurs, sur la quasi-totalité des jeux de données, la *p-value* de J était inférieure à 5%. Cette statistique valide donc bien la qualité au sens de l'étiquette de la structure de voisinage des exemples dans l'espace d'apprentissage, nous informant ainsi sur la qualité de la prédiction que l'on peut espérer.

4.2.3 Adaptation aux cartes étiquetées

En tant que représentation, les cartes étiquetées fournissent une modélisation des relations de voisinage entre les exemples, obtenue indépendamment de leur étiquette de classe. Pour accéder à la qualité de cette représentation, on propose d'adapter la statistique du

poids des arêtes coupées afin de tester la force du lien entre la position sur l'espace de la carte et l'étiquette. Si ce lien est fort, on pourra en déduire que la carte représente convenablement l'information à prédire.

L'espace que l'on souhaite auto-corréler est maintenant celui de la carte. Il devient alors possible de raisonner à deux niveaux différents pour mesurer cette auto-corrélation, celui des exemples et celui des modèles. Pour les exemples, seule la matrice W du poids des arêtes doit être redéfinie afin d'intégrer les proximités dans l'espace de la carte. Nous envisageons deux solutions, soit les matrices $T'_{N\times N}$ et $T''_{N\times N}$. Pour la matrice T', le poids de l'arête entre les exemples i et j est :

$$T'_{ij} = \begin{cases} 1 & \text{si } i \text{ et } j \text{ sont représentés par le même modèle,} \\ 0 & \text{sinon .} \end{cases}$$

Néanmoins, cette mesure exploite uniquement la quantification vectorielle de la carte et ne prend pas en compte sa topologie. Pour y remédier, nous définissons également la matrice T'' telle que :

$$T''_{ij} = \begin{cases} 1 & \text{si } i \text{ et } j \text{ sont représentés par le même modèle,} \\ 1 - \frac{d(m_c(i),m_c(j))}{\max_{k,l} d(m_l,m_k)} & \text{si } i \text{ et } j \text{ sont représentés par des modèles voisins,} \\ 0 & \text{sinon .} \end{cases}$$

De cette manière, le voisinage des exemples au sens de la carte est pris en compte dans l'estimation de la statistique.

En raisonnant au niveau des modèles, la carte étiquetée ne fournit pas une structure de voisinage directement exploitable puisque des modèles sans étiquette résultent de l'apprentissage. Pour ne pas les prendre en compte, nous proposons de construire un graphe de Toussaint des modèles qui ne connecte entre eux que des modèles étiquetés (voir section 2.2 pour la définition des graphes de Toussaint). La figure 2.7 illustre l'obtention de ce graphe à partir de la carte étiquetée. Il nous permet de définir la matrice $T'''_{m\times m}$ des proximités entre modèles telle que :

$$T'''_{ij} = \begin{cases} 1 - \frac{d(m_i,m_j)}{\max_{k,l} d(m_l,m_k)} & \text{si } i \text{ et } j \text{ sont des modèles reliés par une arête} \\ 0 & \text{sinon .} \end{cases}$$

Enfin, pour mesurer la proximité des modèles au sens des étiquettes, nous définissons la matrice $U'_{m\times m}$ dont le terme U'_{ij} vaut 1 si deux modèles sont de même étiquette, 0 sinon.

À partir de ces matrices, nous formons 3 statistiques. J' et J'' estiment la statistique du poids des arêtes coupées de la carte à partir des exemples et J''' estime cette statistique à partir des modèles. Le tableau 2.2 donne le détail de ces trois statistiques.

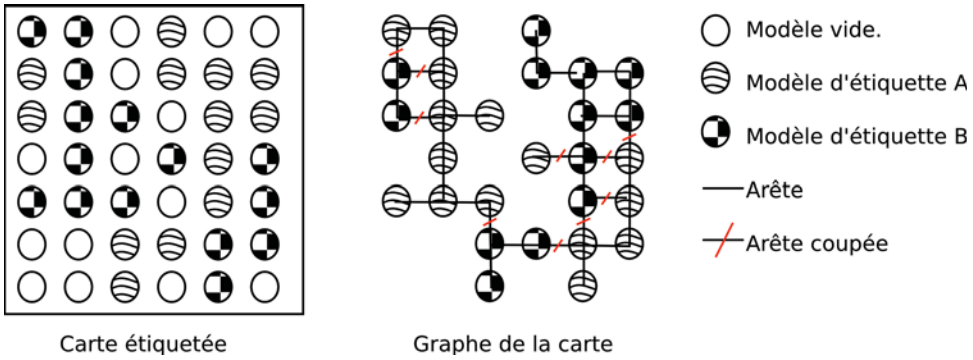


FIG. 2.7: Arêtes coupées de la statistique J''' sur une carte étiquetée avec deux classes.

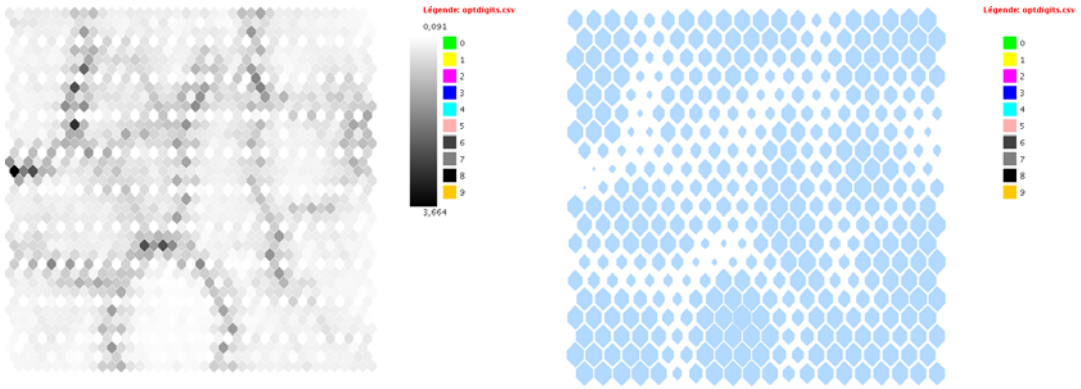
J'	J''	J'''
$\sum_i^N \sum_j^N T'_{ij} U_{ij}$	$\sum_i^N \sum_j^N T''_{ij} U_{ij}$	$\sum_i^M \sum_j^M T'''_{ij} U'_{ij}$

TAB. 2.2: Tableau récapitulatif des statistiques du poids des arêtes coupées pour les cartes étiquetées.

Ces statistiques varient entre 0 et S_0 (le poids total des arêtes). Elles prennent des valeurs d'autant plus faibles que le lien entre l'étiquette des exemples (ou des modèles dans le cas de J''') et leurs positions sur la carte est fort. Dans la section 6.2, nous étudions ces trois statistiques en mesurant leur corrélation avec le taux d'erreur en généralisation des cartes.

5 Visualisation de l'apprentissage

Une fois que l'apprentissage a été validé, ses résultats sont exploitables à travers une visualisation de la carte. Diverses méthodes ont été proposées pour assurer cette visualisation (voir Vesanto et Alhoniemi (2000) pour une revue). Dans la suite, nous présentons ces méthodes en les distinguant suivant l'information qu'elles donnent à voir, à savoir la structure de la carte, les attributs, la réponse aux exemples ou encore l'étiquette de classe. Les figures illustratives de cette section ont été obtenues sur les données *optical digit*. Ces données comportent 5619 exemples décrits par 64 attributs et représentent 10 classes.



Tab. 2.3: Visualisation des distances entre modèles d'une carte apprise sur les données *optical digit*. À gauche : visualisation en U-matrice. Les régions claires entourées de régions sombres correspondent aux classes naturelles des données. À droite : visualisation de la matrice des distances entre modèles. Les classes naturelles des données sont entourées de modèles de plus petite taille.

5.1 En contexte non-supervisé

5.1.1 Visualisation de la structure

Pour visualiser la structure de la carte et comprendre la position des modèles les uns par rapport aux autres, il est possible soit de visualiser sur la carte la distance entre ces modèles, soit de projeter les modèles et leurs relations dans un espace à 2 ou 3 dimensions.

Pour visualiser les distances entre modèles dans l'espace de la carte, Ultsch et Siemon (1990) proposent la visualisation de type U-matrice qui consiste à placer entre deux modèles voisins une nuance de gris représentative de la distance qui les sépare dans l'espace d'apprentissage. De cette manière, si des modèles sont proches les uns des autres mais éloignés du reste de la carte (autrement dit, si ces modèles représentent une classe naturelle), alors ils seront reliés les uns aux autres par des couleurs claires et séparés du reste de la carte par des couleurs sombres (à condition que les couleurs sombres soient affectées aux distances les plus grandes). Cette visualisation fait ainsi apparaître les séparations entre classes. La figure 2.3 donne une illustration de cette visualisation sur les données *optical digit*.

Une manière plus simple, bien que moins informative, consiste à résumer cette information en affichant seulement la distance moyenne d'un modèle à ses voisins. Dans ce cas, plutôt que d'utiliser une variation de couleur pour illustrer cette distance, une visualisation plus intuitive revient à déterminer la taille d'un modèle de manière inversement proportionnelle à sa distance moyenne au voisinage. La figure 2.3 donne un exemple de cette visualisation, toujours sur les données *optical digit*. Cette fois, la séparation entre les

classes apparaît sous la forme d'une série de modèles plus petits que leur voisinage.

S'il est possible de distinguer les frontières entre classes à partir d'une visualisation de type U-matrice, cette dernière ne permet pas de visualiser la structure générale de la carte dans l'espace d'apprentissage. Pour y parvenir, il est courant de recourir à une projection dans un espace de visualisation de la position des modèles dans l'espace d'apprentissage, à l'aide des méthodes de projection présentées en section 2.2.3 du chapitre 1. La projection dans un espace en 2D – par la méthode de Sammon – d'une carte apprise sur les données *optical digit* montre ainsi clairement leur structure arrondie (figure 2.8). Ce type de visualisation permet de mieux appréhender la topologie de la carte.

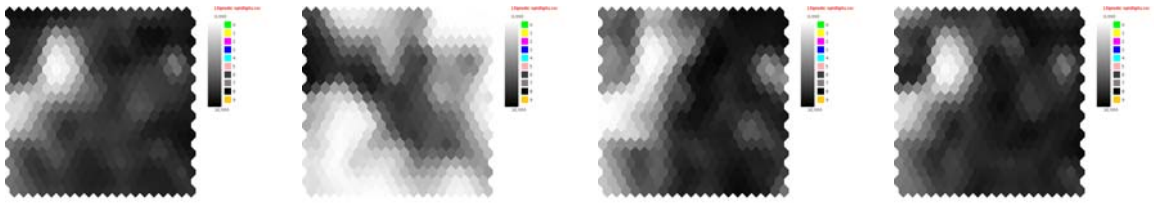


Fig. 2.8: Projection par la méthode de Sammon d'une carte apprise sur les données *optical digit*. Bien qu'il semble difficile d'accéder à la topologie exacte de la carte, la projection montre clairement la structure en arrondie de cette dernière.

5.1.2 Visualisation des attributs

Par rapport à l'ACP, la projection des cartes présente l'avantage d'être non-linéaire. En revanche, le repère construit lors de la projection n'est pas défini en fonction des attributs de départ, comme c'est le cas pour l'ACP. Il n'est donc pas possible de déterminer le rôle exact des attributs dans l'espace de projection. Cet aspect rendant plus difficile l'interprétation de la position des modèles dans l'espace de la carte, plusieurs méthodes ont été proposées pour visualiser l'implication d'un attribut dans la position d'un modèle.

La première solution consiste à établir une carte par attribut, pour laquelle chaque modèle est colorié en fonction de sa valeur sur l'attribut d'intérêt. Cette visualisation permet d'associer facilement les valeurs d'un attribut avec les régions de la carte. En comparant cette visualisation avec la visualisation en U-matrice, il est également possible de relier un intervalle de valeurs d'un attribut avec certaines classes des données. Par



Tab. 2.4: De gauche à droite, cartes des attributs numéro 4, 6, 11 et 60 sur les données *optical digit*. Les attributs 4, 11 et 60 sont corrélés entre eux ce qui se traduit par des colorations similaires de leur carte respective. Par ailleurs, en rapprochant leurs cartes de la visualisation en U-matrice, on s'aperçoit que les valeurs basses de ces attributs (en claires) sont relativement spécifiques de certaines classes. L'attribut 6 représente, en revanche, une information différente.

exemple, la carte de gauche du tableau 2.4 représente les valeurs de l'attribut 4 sur les données *optical digit*. Les valeurs basses de cet attribut (en clair) sont bien localisées sur des régions spécifiques de la carte. En rapprochant cette carte de celles produites à partir des distances entre modèles, on s'aperçoit que ces valeurs basses sont également localisées sur une classe naturelle des données.

En outre, les cartes coloriées à partir des valeurs d'un attribut sont d'autant plus similaires qu'elles sont construites sur des attributs corrélés. Le tableau 2.4 présente 4 cartes apprises sur le même jeu de données mais coloriées à partir d'attributs différents. Avec cette visualisation, il est facile de remarquer que 3 de ces cartes – et donc les attributs correspondants – représentent une information similaire.

Pour se rapprocher du fonctionnement de l'ACP, Rauber (1999) propose d'étiqueter automatiquement chaque modèle en fonction des attributs qu'il représente le mieux. Pour déterminer ces attributs, Rauber se sert de l'erreur de quantification associée à chacun, soit pour l'attribut k du modèle d'indice i (avec C_i sa région de Voronoï) :

$$q_{ik} = \sum_{x_j \in C_i} \sqrt{(m_{ik} - x_{jk})^2}$$

Pour un modèle donné, seuls les attributs dont l'erreur de quantification est inférieure à un certain seuil sont conservés. Avec cette visualisation, chaque modèle se voit ainsi étiqueté par plusieurs attributs.

5.1.3 Visualisation des réponses aux exemples

Outre la contribution des attributs à la carte, il est également possible de visualiser la représentation des exemples par celle-ci. Les deux manières de procéder consistent soit à

s'intéresser au nombre de réponses par modèle, soit à visualiser l'erreur de quantification de chaque modèle.

Pour avoir une idée de la répartition des réponses suivant les modèles, il suffit de visualiser le nombre d'exemples représentés par chaque modèle. Une solution courante consiste à associer un histogramme (resp. une couleur) à chaque modèle dont la taille (resp. la saturation) est proportionnelle au nombre d'exemples. Avec cette visualisation, il est facile de détecter les ensembles de modèles qui représentent des régions denses de l'espace (voir figure 2.9 pour un exemple avec les données *optical digit*).

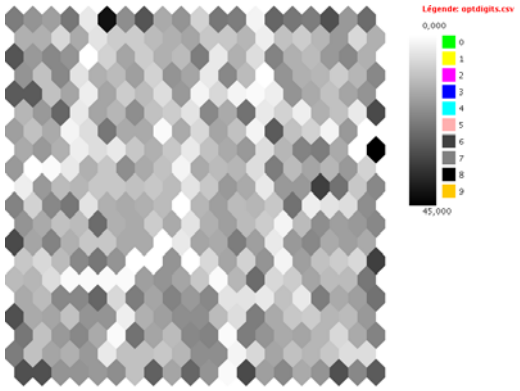


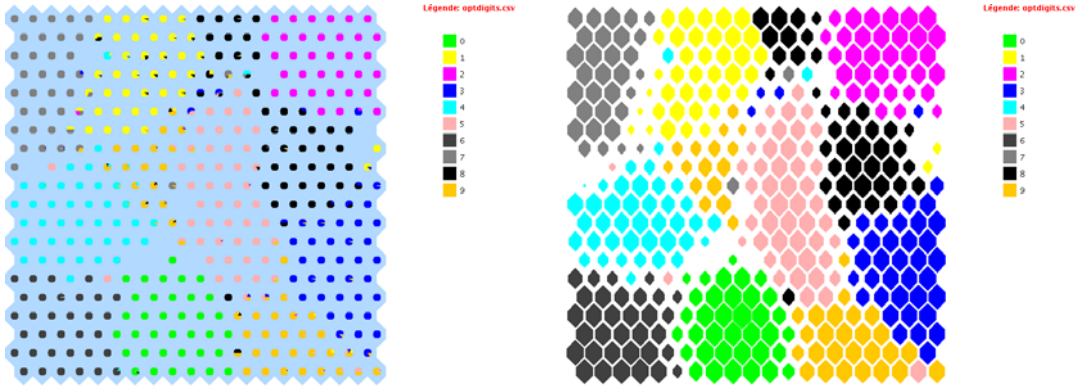
Fig. 2.9: Visualisation du nombre d'exemples représentés par chaque modèle d'une carte apprise sur les données *optical digit*. Plus un modèle est sombre, plus il représente un grand nombre d'exemples.

Pour visualiser l'erreur de quantification de chaque modèle, le plus simple consiste à faire apparaître une barre verticale d'une hauteur proportionnelle à l'erreur sur une carte vue en 3d isométrique. De cette manière, les modèles représentant le moins bien leurs données se distingueront des autres par des barres de grande taille.

5.2 En contexte supervisé

La visualisation d'une carte étiquetée commence par la visualisation de sa structure au travers des stratégies présentées dans la section précédente. À ces visualisations s'ajoutent celles qui permettent de représenter la répartition de la classe sur la carte ainsi que celles qui ont trait à la prédiction d'un nouvel exemple.

La répartition des étiquettes de classe se visualise facilement en associant à chaque étiquette une couleur puis en coloriant chaque modèle avec la couleur de l'étiquette majoritaire parmi les exemples qu'il représente. La figure de gauche du tableau 2.5 indique la classe prédite (parmi les 10 possibles) pour chaque modèle d'une carte. Cette information peut se coupler aisément avec les visualisations décrites à la section précédente. Toujours



Tab. 2.5: À gauche : visualisation de la répartition des étiquettes de classes d'une carte apprise sur les données *optical digit*. Le blanc est associé aux modèles vides (qui ne représentent aucun exemple) et le gris le plus clair aux modèles ambigus. À droite : couplage entre l'information de classe des modèles et la représentation de la matrice des distances entre voisins.

dans le tableau 2.5, la figure de droite reprend la visualisation de la matrice des distances du tableau 2.3 en coloriant chaque modèle avec la couleur de l'étiquette qu'il représente. Il devient alors facile de déterminer les classes les mieux représentées par la carte ainsi que les zones frontières entre classes qui risquent de poser problème lors de la prédiction d'un nouvel exemple. Par ailleurs, il est possible d'affiner ces deux types de visualisation en faisant apparaître, dans chaque modèle, la proportion d'exemples de chaque classe plutôt que la couleur de la classe majoritaire.

Ces visualisations donnent une idée *a priori* de la qualité de la prédiction que l'on s'attend à obtenir en fonction du modèle qui réalise cette prédiction. Cependant, pour un même modèle, cette qualité varie suivant les exemples qu'il représente. Pour visualiser de manière plus spécifique la qualité de la représentation par la carte d'un exemple, Vesanto et Alhoniemi (2000) proposent de s'intéresser à la surface de réponse de cet exemple sur la carte. Cette visualisation ne s'intéresse pas seulement au modèle vainqueur mais regarde la distance de l'exemple à prédire à chacun des modèles. L'idée est qu'un exemple bien représenté sera très proche d'un modèle particulier et de ses voisins mais éloigné du reste de la carte. En revanche, un exemple que la carte a du mal à représenter risque d'avoir une proximité semblable avec plusieurs modèles répartis à différents endroits sur la carte. Pour visualiser cette information, Vesanto et Alhoniemi (2000) mesurent pour chaque modèle i la grandeur $g_i = 1/(1 + q_i^2)$ où q_i est la distance de l'exemple au modèle. Le tableau 2.6 montre la surface de réponse d'une carte apprise sur les données *optical digit* pour quatre exemples différents.



Tab. 2.6: Visualisation de la surface de réponse d'une carte apprise sur les données *optical digit* pour 4 exemples. Plus un modèle est sombre, plus il est proche de l'exemple. Si les prédictions visualisées par les cartes 1 et 4 sont certainement de bonne qualité, il n'en est pas de même pour la deuxième carte où aucun modèle ne semble fortement associé à l'exemple.

5.3 Contextualisation et navigation

Nous distinguons deux niveaux de contextualisation d'un exemple par une carte auto-organisatrice suivant que l'on s'intéresse à la position de l'exemple sur la carte (niveau global) ou à la position de cet exemple dans la région de l'espace que représente son modèle (niveau local). À partir d'une des visualisations décrites à la section précédente, contextualiser un exemple à un niveau global revient simplement à mettre visuellement en avant le modèle qui le représente le mieux.

Au niveau local, il s'agit de positionner l'exemple à contextualiser non plus par rapport aux modèles de la carte, mais par rapport aux exemples qui lui sont proches. Dans ce cas, la carte sert à définir la notion de proximité entre exemples. Ainsi, les exemples proches de celui à contextualiser sont définis comme ceux représentés par le modèle vainqueur, éventuellement étendu à son voisinage (dans le cas où le modèle vainqueur représente trop peu d'exemples). Cette contextualisation permet de représenter rapidement les liens d'un exemple à son voisinage en se limitant aux exemples d'intérêt. En pratique, une fois les exemples définis, il suffit de réaliser une ACP sur les seuls exemples sélectionnés (ou tout autre méthode de représentation).

L'avantage de ces deux niveaux de contextualisation est de donner très rapidement à l'utilisateur une idée du contexte général de son exemple d'intérêt en même temps qu'une représentation précise de l'environnement de cet exemple. Cet avantage se retrouve dans la navigation. Grâce aux cartes, l'utilisateur a un outil permettant de localiser les régions d'intérêt en visualisant la carte dans sa globalité. De cette manière, la quantité d'information mise à sa disposition est d'abord réduite aux traits les plus saillants, représentés ici par les modèles. Lorsque l'utilisateur identifie une région d'intérêt, par exemple en cliquant sur un modèle, la navigation se poursuit par une contextualisation locale des exemples du modèle et de son voisinage. L'utilisateur se déplace alors, dans la limite de la région définie, à travers les exemples. L'avantage de cette stratégie est d'exploiter à la fois les propriétés

Base	Attributs	Classes	Exemples	Taille	Cycles
(1) Abalone	8	29	4177	25×25	90000
(2) Balance Scale	4	3	625	15×15	60000
(3) Breast Cancer	9	2	699	20×20	90000
(4) Glass Indent	9	6	214	10×10	10000
(5) Haberman	3	2	306	10×10	20000
(6) Ionosphère	34	2	351	10×10	20000
(7) Iris	4	3	150	10×10	2000
(8) Italian Olive Oil	9	9	572	15×15	45000
(9) Liver	6	2	345	10×10	35000
(10) Yeast	8	10	1484	25×25	90000

TAB. 2.7: Description des bases et des hyper-paramètres associés

de quantification vectorielle et de projection d'une carte auto-organisatrice pour assurer un accès facile à l'information.

6 Expérimentations

Nous présentons ici des expérimentations menées sur l'utilisation des cartes auto-organisatrices en apprentissage supervisé. Dans un premier temps, nous avons testé la fonction de prédiction Kohonen-Opt que nous avons proposée en section 3.3. Ensuite, nous avons validé la statistique présentée en section 4 en analysant son lien avec le taux d'erreur (Prudhomme et Lallich, 2005b). Dans une dernière partie de ces expérimentations, nous nous sommes intéressés aux capacités de passage à l'échelle des cartes auto-organisatrices (Prudhomme et Lallich, 2005a).

Le tableau 2.7 décrit les données que nous avons utilisées pour mener à bien ces expérimentations. Ces données proviennent du site de l'Université de Californie à Irvine (UCI, Blake et Merz (1998)) sauf *Italian Olive Oil* qui est tiré de Hopke et Massart (1993).

6.1 Carte étiquetée : Kohonen-Opt

Pour valider notre fonction de prédiction Kohonen-Opt, nous avons comparé son taux d'erreur à la fois à celui de la méthode Kohonen-WI (une amélioration de Kohonen-KNN) et d'un arbre de décision (méthode ID3, Quinlan (1990)) sur les 10 jeux de données présentés

Base	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)
T.E. Opt	73,8	17,3	3,2	34,2	24,1	11,6	4,7	7,7	37,5	47,5
T.E. WI	74,4	19,5	3,5	34,5	24,7	12,3	5,3	8,0	43,2	48,9
T.E. ID3	74,2	23,4	4,2	29,0	26,5	8,5	4,00	5,9	42,0	39,8

TAB. 2.8: Taux d'erreur (en %) pour les méthodes Kohonen-Opt, Kohonen-WI et ID3

précédemment. Les fonctions Kohonen-Opt et Kohonen-WI ont été testées avec des cartes paramétrées de la même manière. Le tableau 2.7 donne pour chaque jeu de données la taille de la carte utilisée et le nombre de cycles d'apprentissage. Pour tous les jeux de données, les cartes utilisées possèdent des modèles carrés, un pas d'apprentissage qui décroît linéairement à partir de 1 et une fonction de voisinage rectangulaire. Les résultats de cette expérience sont reportés dans le tableau 2.8.

Sur l'ensemble des jeux de données, Kohonen-Opt assure une meilleure performance que Kohonen-WI. Par rapport à ID3 (Quinlan, 1990), les taux d'erreur de Kohonen-Opt sont sensiblement équivalents (taux d'erreur moyen de 26,17 pour Kohonen-Opt contre 25,76 pour ID3) avec autant de cas favorables que défavorables. Kohonen-Opt exploite donc mieux les informations disponibles dans le voisinage d'un modèle vainqueur, vide ou ambigu. Grâce à cela, cette stratégie assure une meilleure prédiction que Kohonen-WI. Par ailleurs, ces résultats montrent que les cartes fournissent une prédiction fiable, comparable à celle d'un arbre de décision, avec l'avantage d'une représentation des données.

6.2 Validation statistique

Les 3 statistiques de type J présentées en section 4.2.3 ont été construites pour valider statistiquement la représentation obtenue par la carte étiquetée. Nous avons testé ces statistiques sur les 10 jeux de données précédents. Pour chaque jeu, nous avons estimé la valeur de ces statistiques ainsi que la *p-value* associée. Afin de pouvoir comparer le comportement des statistiques entre les différents jeux de données, nous avons normalisé ces statistiques en les divisant par S_0, le poids total des arêtes. Le tableau 2.9 présente les résultats obtenus, ainsi que les taux d'erreur (T.E.) déterminés avec la méthode Kohonen-Opt.

Comme nous pouvons le voir, les *p-value* associées aux statistiques J' et J'' sont significatives ($p < 0,05$, à l'exception de la base *Haberman* pour laquelle $p = 0,08$). Pour la

Base	$\frac{J'}{S_0}$	**p-value**	$\frac{J''}{S_0}$	**p-value**	$\frac{J'''}{S_0}$	**p-value**	**T.E.**
(1)	79,96	0	79,88	0	81,97	1	73,86
(2)	21,66	0	23,68	0	22,28	0	17,3
(3)	0,40	0	1,10	0	8,30	0	3,21
(4)	43,29	0	44,64	0	61,65	0,02	34,21
(5)	34,57	0,078	32,51	0,005	28,20	0,022	24,06
(6)	10,92	0	14,76	0	36,11	0,022	11,6
(7)	3,60	0	4,70	0	8,50	0	4,67
(8)	41,40	0	8,05	0	20,06	0	7,69
(9)	60,58	0,0031	0,3750	0	58,28	0,9989	37,53
(10)	50,00	0	52,40	0	64,52	0	47,53
Moyenne	31,64	0,0081	29,92	0,0005	27,99	0,2045	

TAB. 2.9: Résultats sur 10 bases des statistiques de test de type J, de leur *p-value* et du taux d'erreur en généralisation avec Kohonen-Opt (T.E.)

statistique J''', deux *p-values* sont proches de 1. Dans ces deux cas, la carte étiquetée issue de l'apprentissage comporte un grand nombre de modèles ambigus. Or, cette catégorie de modèles donne lieu à des arêtes coupées au sens de la statistique J'''. Finalement, lorsque le nombre de modèles ambigus augmente, la statistique J''' se trouve dans une situation proche de l'indépendance entre l'étiquette des modèles et leur position. Comme le montre le tableau 2.9, cette situation ne se produit que dans des cas où le taux d'erreur est élevé (il s'agit là d'une autre conséquence d'un nombre important de modèles ambigus).

Dans un deuxième temps, nous avons mesuré la corrélation linéaire entre le taux d'erreur en généralisation des cartes et les statistiques J. Le tableau 2.10 détaille les résultats obtenus et les corrélations de J'' et J''' sont représentées graphiquement dans la figure 2.10.

J'	J''	J'''
$r^2 = 0,78$	$r^2 = 0,98$	$r^2 = 0,88$

TAB. 2.10: Cœfficient de corrélation entre les statistiques J et le taux d'erreur

En ne s'intéressant qu'aux exemples représentés par un même modèle, la statistique J' ne tient pas compte de la proximité entre modèles et donc de la topologie de la carte. C'est la raison de sa faible corrélation avec le taux d'erreur en généralisation. La statistique J'', au contraire, intègre le voisinage des modèles dans sa validitation de la représentation,

d'une part en considérant des arêtes entre exemples représentés par des modèles voisins et d'autre part, en pondérant le poids d'une arête en fonction de la distance entre les modèles. Cette statistique présente la meilleure corrélation avec le taux d'erreur. L'inconvénient majeur de cette statistique est qu'elle nécessite de considérer les relations entre chaque paire d'exemples. La statistique J''' propose une abstraction de J'' en raisonnant directement sur les paires de modèles, ce qui présente un avantage computationnel certain. La corrélation de cette statistique avec le taux d'erreur en généralisation est intermédiaire aux deux précédentes. Comme pour J'', cette statistique prend en compte la topologie locale de la carte en considérant le voisinage des modèles. En revanche, en raisonnant sur les modèles et non plus sur les exemples, cette statistique souffre de l'approximation engendrée par la projection.

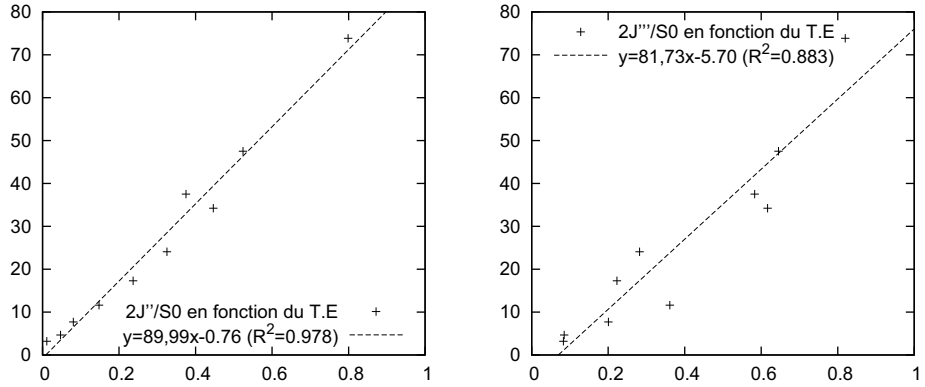


Fig. 2.10: T.E. en généralisation en fonction de $2J''/S_0$ (figure de gauche) et de $2J'''/S_0$ (figure de droite). Sur chaque figure, la droite en pointillé représente la droite de régression entre le T.E. et la statistique.

6.3 Passage à l'échelle

Pour contrôler les capacités de passage à l'échelle des cartes auto-organisatrices, que ce soit en termes du nombre d'exemples ou du nombre d'attributs, nous nous sommes servis de données synthétiques. Pour simuler une augmentation du nombre d'individus, nous nous sommes servis du générateur de données *wave* disponible sur le site de l'UCI. Il génère automatiquement des exemples dans un espace à 40 dimensions en gardant une erreur bayésienne globale d'environ 14%. Nous avons ainsi généré des jeux de données variant entre 1250 et 1280000 exemples pour lesquels nous avons mesuré le taux d'erreur en généralisation d'une carte étiquetée avec Kohonen-Opt, ainsi que les valeurs des statistiques J'' et J''' (tableau 2.11).

Le taux d'erreur obtenu reste stable lorsque le nombre d'individus augmente (environ 15%), de même que les valeurs des statistiques J'' et J'''. Ces dernières sont néanmoins

Taille	$2J''/S_0$	$2J'''/S_0$	**T.E.**	**Temps (s)**
1250	26,27	17,80	16,03	2
2500	26,09	13,39	15,70	5
5000	24,92	9,87	15,46	10
10000	24,36	9,44	15,57	20
20000	23,68	7,42	14,92	41
40000	23,04	7,78	14,84	78
80000	23,22	7,40	15,25	127
160000	23,03	7,45	14,93	245
320000	23,37	7,92	15,04	500
640000	23,09	7,18	15,17	1073
1280000	22,94	7,98	15,22	2014

TAB. 2.11: Statistique J'' et J''', taux d'erreur (T.E.) et temps d'apprentissage en fonction de l'augmentation du nombre d'individus du jeu de données *wave*

soumises à une légère baisse qui s'explique par le temps d'apprentissage, plus long pour les grands jeux de données, qui conduit à une meilleure représentation.

Pour simuler une augmentation du nombre d'attributs, nous avons généré des données. Pour cela nous avons mis en place un programme qui nous permet de choisir le nombre d'individus, le nombre de prédicteurs, l'intervalle de valeurs de ces prédicteurs et le nombre de classes du jeu de données. Pour chaque classe, deux types d'attributs sont définis : des attributs dont les valeurs sont générées aléatoirement sur l'intervalle de valeurs (ces attributs n'apportent donc aucune information sur la variable de classe) et des attributs dont les valeurs sont générées suivant une loi normale dont on spécifie la moyenne et l'écart-type (ces attributs sont caractéristiques de la classe). En utilisant ce programme, nous avons généré 5 jeux de données de 40000 individus et 3 classes avec un nombre de prédicteurs variant de 20 à 100. Chaque classe est caractérisée par cinq attributs dont les valeurs sont générées par une loi normale. Le tableau 2.12 les décrit plus en détails.

Pour chacun des jeux générés, nous avons calculé le taux d'erreur obtenu par Kohonen-Opt avec une 10-validation croisée et les valeurs des statistiques J'' et J'''. Nous avons également mesuré le temps nécessaire à l'apprentissage. Les résultats sont reportés dans le tableau 2.13.

Classe	Attributs				
C1	1	2	3	4	5
	N(25,5)	N(55,5)	N(50,10)	N(85,5)	N(30,10)
C2	2	3	6	7	8
	N(20,5)	N(70,10)	N(25,5)	N(85,5)	N(30,10)
C3	7	8	9	10	11
	N(55,5)	N(50,10)	N(25,5)	N(20,5)	N(70,10)

TAB. 2.12: Données générée : description des lois normales utilisés pour la génération des attributs caractéristiques des différentes classes

Le taux d'erreur en généralisation augmente avec le nombre de dimensions. Cette augmentation va de pair avec une augmentation des statistiques J'' et J'''. Le temps d'apprentissage, quant à lui, augmente linéairement. La dimensionnalité des données a donc un impact direct sur la qualité de la prédiction, même si la dimension intrinsèque des données ne varie pas. Les dimensions non corrélées avec la classe constituent un bruit qui dégrade la projection effectuée par l'algorithme des cartes auto-organisatrices.

Taille	$2J''/S_0$	$2J'''/S_0$	T.E.	Temps (s)
20	22,87	16,46	17,12	10
40	25,10	17,50	19,05	16
60	27,70	19,91	21,02	23
80	28,62	21,35	22,38	29
100	27,32	22,24	22,20	36

TAB. 2.13: Statistiques J'' et J''', taux d'erreur (T.E.) et temps d'apprentissage en fonction de l'augmentation du nombre d'attributs

7 Bilan et perspectives

Dans ce chapitre, nous avons considéré les cartes auto-organisatrices sous l'angle de la représentation des données en apprentissage supervisé. Les cartes étiquetées sont un moyen simple et rapide pour obtenir une telle représentation. En outre, nous avons montré qu'elles possèdent de bonnes capacités de généralisation, capacités qu'il est possible de valider statistiquement. Néanmoins, les cartes auto-organisatrices souffrent de l'augmentation de la

dimension puisque plus la dimensionnalité des données est grande, plus l'erreur de leur projection dans un plan sera importante. Par ailleurs, l'expérimentation que nous avons menée montre que même si la dimension intrinsèque des données ne varie pas, la présence d'attributs sans lien avec l'étiquette détériore la prédiction.

Pour faire face à des données en grandes dimensions, il a été proposé de modifier la mesure de distance entre exemples (Aggarwal et al., 2001; Fessant et al., 2001), afin d'éviter que cette distance ne perde son pouvoir discrimimant (comme c'est le cas pour la distance euclidienne, voir section 4.1.2). Ces auteurs ont proposé de la remplacer par des distances fractionnaires. Ces distances, tout comme la distance euclidienne, sont des cas particuliers de la distance de Minkowski. Entre deux exemples $X = (x_1, x_2, \ldots, x_d)$ et $Z = (z_1, z_2, \ldots, z_d)$, la distance de Minkowski se définit telle que :

$$d(X, Z) = \left(\sum_{i=1}^{d} |x_i - z_i|^p \right)^{1/p}$$

Pour obtenir la distance euclidienne, on choisit $p = 2$ et pour les distances fractionnaires $p < 1$.

Aggarwal et al. (2001) se sont intéressés à la qualité du voisinage défini par chacune de ces distances. Pour cela, ils ont calculé le nombre de plus proches exemples de même étiquette d'un exemple cible, à partir de chacune de ces deux distances. En trouvant davantage d'exemples de même étiquette, les distances fractionnaires se sont montrées plus pertinentes dans ce contexte. Doherty et al. (2004) ont reproduit cette expérience en incluant une normalisation des données (par centrage-réduction). Celle-ci a eu pour effet de réduire notablement l'avantage des distances fractionnaires sur les distances euclidiennes. Ils ont par ailleurs testé l'usage de ces deux distances dans des algorithmes d'apprentissage supervisé (k-PPV et *neural gas*) et n'ont alors constaté aucune différence sur la qualité de la prédiction.

Dans des espaces de grandes dimensions, la mesure donnée par les distances fractionnaires garde donc un sens pour des valeurs de distance plus élevées que la distance euclidienne. Néanmoins, il est possible d'améliorer le résultat de la distance euclidienne en appliquant une normalisation. Ces résultats ne signifient pas que les distances fractionnaires soient plus adaptées aux espaces de grandes dimensions. En effet, pour mesurer des distances proches, la distance euclidienne conserve son pouvoir discriminant. Or, pour la plupart des algorithmes d'apprentissage, ce sont ces valeurs qui déterminent la qualité du résultat final.

Devant la dégradation de la représentation de données en grandes dimensions, nous avons envisagé une autre solution fondée sur l'apprentissage puis l'agrégation de plusieurs

représentations. Pour chaque représentation, l'idée est de contourner la dimensionnalité en ne représentant qu'un sous-espace de l'espace d'apprentissage. La totalité de l'espace continue à être représenté mais en étant réparti sur les différentes représentations. Enfin, l'agrégation permet l'exploitation de ces multiples représentations sous la forme d'une représentation unique et/ou d'une prédiction.

La conception d'un ensemble de classifieurs pour l'apprentissage et la prédiction d'un jeu de données est une référence directe aux méthodes ensemblistes. Ces méthodes, désignées par le terme d'ensembles ou de comités, ont connu un regain d'attention ces dernières années pour devenir aujourd'hui un champ d'étude à part entière. Nous les introduisons dans le chapitre suivant avant de les mettre en œuvre pour l'apprentissage d'un comité de cartes visant la représentation de données en apprentissage supervisés et en grandes dimensions.