Cartes auto-organisatrices pour la classification des données de type intervalle en se basant sur la distance city-block

Chantal Hajjar, Hani Hamdan

École Supérieure d'Électricité (SUPÉLEC)
Département de Traitement du Signal et Systèmes Électroniques
Chantal.Hajjar@supelec.fr, Hani.Hamdan@supelec.fr

Résumé. Les données symboliques permettent de mieux représenter les mesures issues des applications réelles, fournissant ainsi un niveau de connaissance plus élevé qu'avec une représentation en valeurs simples. Les données de type intervalle font partie des données symboliques. Elles sont utilisées pour représenter, selon le cas, la variabilité ou l'incertitude dans les mesures. Dans cet article, nous proposons un algorithme pour l'apprentissage des cartes auto-organisatrices en mode différé (batch) dans le but de classifier des données de type intervalle tout en préservant leur topologie. L'apprentissage de la carte se fait en optimisant un critère basé sur la distance city-block. La méthode proposée est testée et comparée à d'autres méthodes de classification de données intervalles en utilisant deux jeux de données de type intervalle réelles.

1 Introduction

Très souvent, les données réelles ne peuvent pas être modélisées par des valeurs simples, mais nécessitent des modèles plus complexes comme des ensembles de valeurs, des intervalles de valeurs, des distributions, etc. On parle alors de données symboliques (Noirhomme-Fraiture et Brito, 2011). Les données de type intervalle sont un exemple de données symboliques qui reflètent, en fonction du cas, la variabilité ou l'incertitude dans les mesures observées. De nombreux outils d'analyse de données ont déjà été adaptés pour prendre en compte les intervalles : analyse en composantes principales (Cazes et al., 1997), analyse factorielle (Chouakria, 1998), régression (Billard et Diday, 2000), positionnement multidimensionnel (Denœux et Masson, 2000), perceptron multicouche (Rossi et Conan-Guez, 2002), etc. Dans le domaine de la classification, Chavent et Lechevallier (2002) ont proposé un algorithme de nuées dynamiques *(generalized K-means)* pour les données de type intervalle où les prototypes sont des éléments de l'espace de représentation des objets à classer, c'est-à-dire des vecteurs dont les composantes sont des intervalles. Dans cette approche, les prototypes sont définis par l'optimisation d'un critère d'adéquation en se basant sur la distance de Hausdorff. Bock (2003) a construit une carte auto-organisatrice basée sur la distance vertex-type pour la visualisation des données intervalles. Hamdan et Govaert ont développé une théorie sur la classification des données de type intervalle en utilisant les modèles de mélange. Dans ce contexte, ils ont proposé deux approches fondées sur le maximum de vraisemblance : l'approche mélange (Hamdan et Govaert, 2005) et l'approche classification (Hamdan et Govaert, 2004). Bock (2008) a proposé

un modèle probabilistique (approche classification) pour les données symboliques, en particulier pour les données de type intervalle. De Souza et De Carvalho (2004) ont proposé deux algorithmes de nuées dynamiques pour des données intervalles en utilisant la distance city-block adaptative et non adaptative. De Souza et al. (2004) ont proposé deux méthodes de nuées dynamiques, basées sur la distance de Mahalanobis, pour des données intervalles. Dans les deux méthodes, les prototypes sont définis par l'optimisation d'un critère adéquat basé sur une extension de la distance de Mahalanobis. Dans la première méthode, la distance utilisée est adaptative et commune à toutes les classes tandis que dans la seconde méthode, chaque classe a sa propre distance adaptative. El Golli et al. (2004) ont proposé une carte auto-organisatrice pour les données intervalles mais en présentant à la carte un tableau de dissimilarités plutôt qu'un tableau *individus-variables*. De Carvalho et al. (2006) ont appliqué l'algorithme de nuées dynamiques basé sur la distance L_2 pour les données intervalles et ont présenté trois techniques de normalisation pour les variables de type intervalle.

Toujours dans le cadre de la classification de données intervalles, Hajjar et Hamdan ont proposé une carte auto-organisatrice pour les données de type intervalle en utilisant la distance L_2 (Hajjar et Hamdan, 2011b) et la distance de Hausdorff (Hajjar et Hamdan, 2011a), comme ils ont proposé aussi une méthode de classification adaptative pour les données intervalles en utilisant les cartes auto-organisatrices et la distance de Mahalanobis (Hajjar et Hamdan, 2013). De Carvalho *et al.* ont adapté l'algorithme d'optimisation pour les cartes topologiques sur les données de type intervalle en utilisant des distances L_2 (De Carvalho et Pacifico, 2011) et L_1 (De Carvalho et al., 2012) adaptatives et non adaptatives. Cabanes et al. (2013) ont étendu l'algorithme *S2L-SOM (Simultaneous Two-Levels-SOM)* aux données intervalles. Il s'agit d'un algorithme qui permet de faire une projection des données sur une grande carte tout en classifiant les vecteurs prototypes d'une façon simultanée. À la fin de l'exécution de l'algorithme, chaque observation appartiendra à la classe de son neurone vainqueur. L'avantage de cette méthode réside dans le fait que le nombre de classes est détecté automatiquement et des classes de formes quelconques sont reconnues.

Dans cet article, nous proposons un algorithme d'apprentissage d'une carte auto-organisatrice pour la classification des données de type intervalle en optimisant un critère basé sur la distance city-block. Tout d'abord, dans la section 2, nous exposons rapidement le principe des cartes auto-organisatrices classiques de Kohonen. Ensuite, dans la section 3, nous présentons la méthode proposée. Dans la section 4, nous montrons les résultats de la mise en oeuvre de notre approche sur deux jeux de données réelles. Enfin, dans la section 5, nous donnons notre conclusion.

2 Principe des cartes auto-organisatrices

Les cartes auto-organisatrices, désignées en anglais par *Self-Organizing Maps (SOM)*, furent inventées par Kohonen (Kohonen, 1984, 2001). C'est une sorte de réseau de neurones artificiels soumis à un apprentissage non supervisé pour projeter des données de haute dimensionnalité sur un espace de faible dimension dans le but d'en faire des tâches de discrétisation, de quantification vectorielle ou de classification. Une carte auto-organisatrice bidimensionnelle est composée de neurones disposés sur une grille rectangulaire ou hexagonale. La figure 1 représente une carte auto-organisatrice dont les 49 neurones sont disposés suivant une grille carrée. À chaque neurone est associé un vecteur référent (appelé aussi vecteur prototype) dans l'espace

de données (ou espace d'entrée) et un emplacement sur la carte (ou espace de sortie) formé par le numéro de ligne et le numéro de colonne du neurone. L'algorithme d'apprentissage de la carte peut se faire incrémentalement ou en différé (batch). Dans la version incrémentale, chaque itération consiste à présenter à la carte un vecteur de données choisi au hasard, ensuite, sa distance euclidienne de l'ensemble des vecteurs référents est calculée. Le neurone dont le vecteur référent est le plus proche du vecteur d'entrée est appelé le neurone vainqueur ou le *Best Matching Unit (BMU)*. Les vecteurs référents du neurone vainqueur et de ses voisins se déplacent vers le vecteur d'entrée. La valeur de ce déplacement décroît avec le temps et en s'éloignant du neurone vainqueur. Une fonction de voisinage est utilisée pour définir la proximité entre les neurones. Dans le mode différé, à chaque itération, tous les vecteurs d'entrée sont présentés au réseau. Le neurone vainqueur de chaque vecteur d'entrée est déterminé. Chaque vecteur référent est une moyenne pondérée des vecteurs d'entrée, les poids étant les valeurs de la fonction de voisinage définissant la proximité entre le neurone vainqueur du vecteur d'entrée et le neurone dont le vecteur référent est calculé.

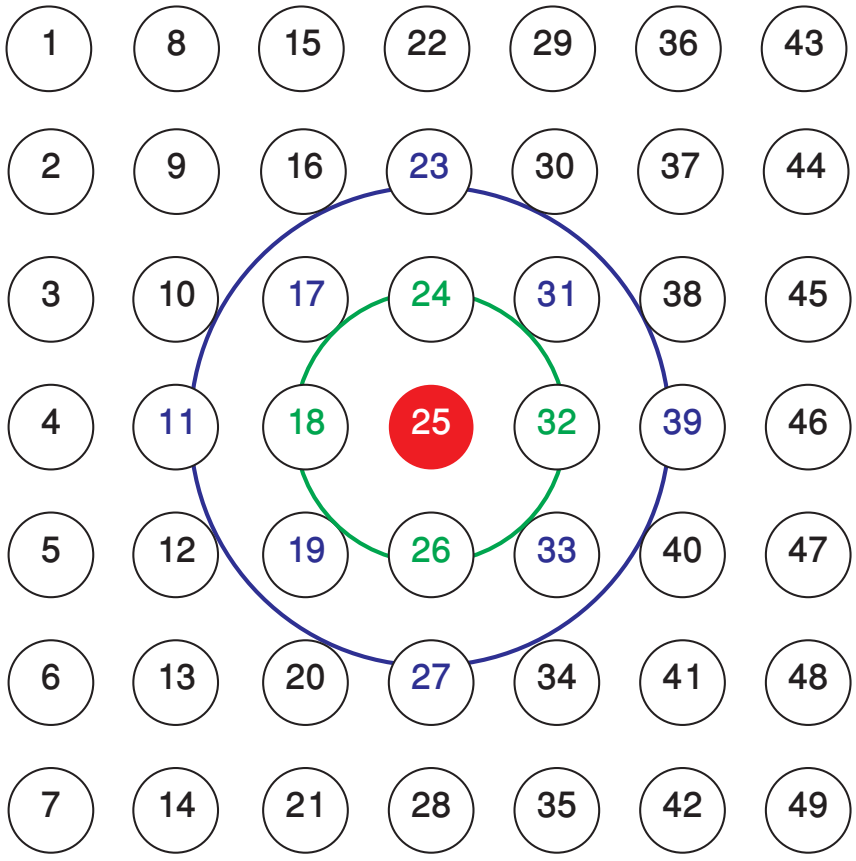

FIG. 1 – *Grille carrée de 49 neurones.*

3 Cartes auto-organisatrices pour les données de type intervalle

Soit $\mathcal{R} = \{\boldsymbol{\mathcal{R}}_1, \ldots, \boldsymbol{\mathcal{R}}_n\}$ un ensemble de n vecteurs de données symboliques décrits par p variables de type intervalle. Chaque individu $\boldsymbol{\mathcal{R}}_i$ est représenté par un vecteur d'intervalles $\boldsymbol{\mathcal{R}}_i = ([a_i^1, b_i^1], \ldots, [a_i^p, b_i^p])^T$ où $[a_i^j, b_i^j] \in I\!I = \{[a, b] \ ; \ a \in I\!R, \ b \in I\!R, \ a \leqslant b\}$. Nous proposons de créer une carte auto-organisatrice de K neurones. À chaque neurone k est associé un vecteur référent $\boldsymbol{\mathcal{W}}_k = ([u_k^1, v_k^1], \ldots, [u_k^p, v_k^p])^T$. L'ensemble des vecteurs prototypes est représenté par $\mathbb{W}$.

Dans l'apprentissage en mode différé proposé par Kohonen, chaque itération de l'algorithme consiste à présenter toutes les observations à la carte, à déterminer le neurone vainqueur de chaque observation et à mettre à jour les vecteurs prototypes des neurones d'une manière heuristique. Dans l'algorithme que nous proposons, l'apprentissage de la carte se fait en minimisant le critère suivant :

$$G_{intSOM}(\mathbb{W}) = \sum_{i=1}^{n} \sum_{k=1}^{K} h_{kc_i} d(\boldsymbol{\mathcal{R}}_i, \boldsymbol{\mathcal{W}}_k) \tag{1}$$

où h_{kc_i} est la fonction de voisinage définie dans l'équation (3), c_i est le neurone vainqueur de l'observation $\boldsymbol{\mathcal{R}}_i$ défini dans l'équation (4), et d est une distance entre deux vecteurs d'intervalles définie dans l'équation (2) en se basant sur la distance L_1 ou la distance city-block :

$$d(\boldsymbol{\mathcal{R}}_i, \boldsymbol{\mathcal{W}}_k)) = \sum_{j=1}^{p} \Big(\big|a_i^j - u_k^j\big| + \big|b_i^j - v_k^j\big| \Big) \tag{2}$$

$$h_{kc_i}(t) = \exp\left(-\frac{\|\boldsymbol{r}_k - \boldsymbol{r}_{c_i}\|^2}{2\sigma^2(t)} \right) \tag{3}$$

où $\boldsymbol{r}_k$ et $\boldsymbol{r}_{c_i}$ sont respectivement les vecteurs représentant les coordonnées dans l'espace $I\!R^2$ des neurones k et c_i, et $\sigma(t)$ est le rayon de voisinage à l'itération t.

Heskes (1999) a prouvé que pour minimiser le critère G_{intSOM}, la recherche du neurone vainqueur doit se faire de la façon suivante :

$$c_i = \underset{k \in \{1,\ldots,K\}}{\arg\min} \sum_{\tau=1}^{K} h_{k\tau}(t) d\big(\boldsymbol{\mathcal{R}}_i, \boldsymbol{\mathcal{W}}_k\big) \tag{4}$$

plutôt qu'à la manière de Kohonen où le neurone vainqueur est le neurone c_i dont le vecteur référent se trouve à une distance minimale de l'observation $\boldsymbol{\mathcal{R}}_i$.

La détermination d'un vecteur prototype $\boldsymbol{\mathcal{W}}_k$ se fait en minimisant le critère suivant :

$$g_k(\boldsymbol{\mathcal{W}}_k) = \sum_{i=1}^{n} h_{kc_i}(t) d\big(\boldsymbol{\mathcal{W}}_k, \boldsymbol{\mathcal{R}}_i\big) \tag{5}$$

g_k sera minimisé s'il l'est pour chaque dimension j :

$$g_k^j(\boldsymbol{\mathcal{W}}_k) = \sum_{i=1}^{n} h_{kc_i}(t) \Big(\big|a_i^j - u_k^j\big| + \big|b_i^j - v_k^j\big| \Big) \tag{6}$$

u_k^j et v_k^j qui minimisent g_k^j sont respectivement la médiane pondérée des $a_i^j, i \in \{1, \ldots, n\}$ et des $b_i^j, i \in \{1, \ldots, n\}$, les poids étant les valeurs de la fonction de voisinage h_{kc_i} (voir annexe A). La mise à jour des vecteurs référents se fait alors comme suit :

$$u_k^j = mediane\_ponderee(a_i^j) \quad i \in \{1, \ldots, n\} \tag{7}$$

$$v_k^j = mediane\_ponderee(b_i^j) \quad i \in \{1, \ldots, n\} \tag{8}$$

À chaque itération t du processus d'apprentissage, un rayon de voisinage est fixé, les valeurs des fonctions de voisinage $h_{kc_i}(t)$ sont calculées et la détermination des vecteurs prototypes des neurones se fait en minimisant le critère G_{intSOM}. Donc, chaque itération du processus d'apprentissage peut engendrer plusieurs sous-itérations s jusqu'à ce que le critère G_{intSOM} soit minimisé (Anouar et al., 1997). Ce qui fait que chaque itération ressemble à l'algorithme des nuées dynamiques avec la différence que les distances sont pondérées par les valeurs de la fonction de voisinage pour assurer la préservation de la topologie. Le rayon de voisinage est initialisé à une valeur suffisamment grande pour activer le plus grand nombre de neurones possible, puis décroît avec les itérations pour atteindre une petite valeur à la fin de l'algorithme de façon à activer seulement le neurone vainqueur. À la fin de l'apprentissage, l'ensemble des observations est partitionné en K classes. Toutes les observations ayant le même neurone vainqueur appartiendront à la même classe. En plus, deux neurones voisins sur la carte représenteront des observations voisines dans l'espace d'entrée.

3.1 Algorithme d'apprentissage

L'algorithme d'apprentissage d'une carte auto-organisatrice pour la classification des données intervalles en optimisant un critère basé sur la distance city-block (intSOM_DYN_city-block) est détaillé comme suit :

— Entrées :
  — Les observations $\boldsymbol{\mathcal{R}}_i = \left([a_i^1, b_i^1], \ldots, [a_i^p, b_i^p]\right)^T, i \in \{1, \ldots, n\}$
  — Dimensions de la grille $(lines, cols)$
  — Valeur initiale (σ_{init}) et finale (σ_{final}) du rayon de voisinage.
— Sorties :
  — Les vecteurs référents auto-organisés $\boldsymbol{\mathcal{W}}_k = ([u_k^1, v_k^1], \ldots, [u_k^p, v_k^p])^T, k \in \{1, \ldots, K\}$
  — Partition finale de l'ensemble des observations : $P_f = \{C_1, C_2, \ldots, C_K\}$

1. Initialisation :
   — $t = 0$, (t : itération liée à la réduction du rayon de voisinage).
   — $s = 0$, (s : sous-itération liée à la mise à jour des vecteurs référents).
   — Le nombre de neurones est $K = lines \cdot cols$.
   — Affecter au rayon de voisinage $\sigma(t)$ la valeur σ_{init}.
   — Initialiser les vecteurs référents $\boldsymbol{\mathcal{W}}_k(0), k \in \{1, \ldots, K\}$ en leur affectant aléatoirement des observations de l'ensemble d'entrée.
2. Calcul des valeurs de la fonction de voisinage $h_{k\tau}(t)$ pour $k = 1$ à K et $\tau = 1$ à K.
3. Détermination des vecteurs référents en optimisant le critère G_{intSOM} :

(a) Calcul des neurones vainqueurs des observations : pour $i = 1$ à n

$$c_i = \underset{k\in\{1,\ldots,K\}}{\arg\min} \sum_{\tau=1}^{K} h_{k\tau}(t) \sum_{j=1}^{p} \Big(\big|a_i^j - u_k^j(s)\big| + \big|b_i^j - v_k^j(s)\big| \Big)$$

(b) Calcul des valeurs de la fonction de voisinage $h_{kc_i}(t)$ pour $i = 1$ à n et $k = 1$ à K.

(c) Mise à jour des vecteurs référents : pour $k = 1$ à K

$$u_k^j(s+1) = mediane\_ponderee(a_i^j)\ i \in \{1,\ldots,n\}$$
$$v_k^j(s+1) = mediane\_ponderee(b_i^j)\ i \in \{1,\ldots,n\}$$

(d) $s = s + 1$

(e) Si $\sum_{i=1}^{n}\sum_{k=1}^{K} h_{kc_i}(t) d\big(\boldsymbol{\mathcal{R}}_i, \boldsymbol{\mathcal{W}}_k(s)\big) \neq \sum_{i=1}^{n}\sum_{k=1}^{K} h_{kc_i}(t)\big(\boldsymbol{\mathcal{R}}_i, \boldsymbol{\mathcal{W}}_k(s-1)\big)$ alors aller à **(a)**.

4. $t = t + 1$
5. Réduire $\sigma(t)$.
6. Si $\sigma(t) > \sigma_{final}$ alors aller à **2.**

3.2 Choix des paramètres de la carte

Le choix des paramètres de la carte doit être fait judicieusement pour que l'apprentissage mène à de bons résultats. Dans ce papier, les paramètres de la carte sont choisis de la façon suivante :

— **Vecteurs référents initiaux.** Tout comme l'algorithme des nuées dynamiques et l'algorithme de Kohonen classique, les résultats produits par cet algorithme dépendent de l'initialisation des vecteurs référents. Pour remédier à ce problème, plusieurs instances de cet algorithme sont lancées avec à chaque fois une initialisation différente, et les résultats produits par l'instance qui permet de minimiser le plus le critère G_{intSOM} sont adoptés.

— **Dimensions de la carte.** Du fait qu'on utilise la carte auto-organisatrice pour la classification des données, les dimensions de la carte sont choisies de façon à ce que le nombre total des neurones corresponde au nombre de classes que l'on désire obtenir.

— **Choix des valeurs initiale et finale du rayon de voisinage.** La valeur initiale du rayon de voisinage doit être choisie de façon à garantir l'activation de la plupart des neurones de la carte en début d'apprentissage (presque la moitié de la plus grande dimension de la carte). Vers la fin de l'apprentissage, le rayon de voisinage doit atteindre une valeur suffisamment petite pour n'activer que le neurone vainqueur seulement.

3.3 Normalisation des données

Chavent et Saracco (2008) ont proposé des techniques de normalisation des variables de type intervalle compte tenu de la distance utilisée. Dans le cas où la distance entre vecteurs

d'intervalles est basée sur la distance L_1, une mesure de dispersion s_j pour chaque variable intervalle j est calculée par :

$$s_j = \sum_{i=1}^{n} \left(|a_i^j - \alpha^j| + |b_i^j - \beta^j| \right)$$
$$\alpha^j = median(a_i^j)\ i \in \{1, \ldots, n\}$$
$$\beta^j = median(b_i^j)\ i \in \{1, \ldots, n\}$$

Chaque intervalle $[a_i^j, b_i^j]$ sera normalisé en $[\tilde{a}_i^j, \tilde{b}_i^j]$ tel que :

$$\tilde{a}_i^j = \frac{a_i^j}{s_j} \quad \tilde{b}_i^j = \frac{b_i^j}{s_j}$$

4 Étude expérimentale

Deux jeux de données intervalles réelles sont utilisés dans le but de valider la méthode proposée. Le premier jeu de données regroupe des informations concernant des modèles de voitures, et le deuxième jeu de données représente les températures minimales et maximales enregistrées dans plusieurs villes du monde durant les 12 mois de l'année.

4.1 Jeu de données *Car models*

Le jeu de données *Car models* (De Carvalho et al., 2006) regroupe des informations sur 33 modèles de voitures décrits par 8 variables de type intervalle. Une partition *a priori* de ce jeu classifie ces modèles en 4 classes (voir tableau 1). Après avoir normalisé les variables suivant la technique décrite dans le paragraphe 3.3, le jeu de données normalisées est présenté à une carte auto-organisatrice où les neurones sont disposés suivant une grille carrée de 4 neurones. L'apprentissage de la carte est fait en deux itérations qui correspondent à un rayon de voisinage initial égal à 1 et final égal à 0,1. 50 instances de cet algorithme sont lancées avec à chaque fois une initialisation différente des vecteurs référents, et les résultats produits par l'instance qui permet d'avoir la plus petite valeur du critère G_{intSOM} sont adoptés. Quand le rayon de voisinage est fixé à 1, l'optimisation du critère G_{intSOM} nécessite 4 sous-itérations et quand le rayon de voisinage est réduit à 0,1, l'optimisation du critère G_{intSOM} nécessite 7 sous-itérations. La partition finale du jeu de données est illustrée dans la figure 2.

Classe	Modèle
Utilitaire	-Alfa 145/U -Audi A3/U -Punto/U -Fiesta/U -Lancia Y/U -Nissan Micra/U -Corsa/U -Twingo/U -Rover 25/U -Skoda Fabia/U
Berline	-Alfa 156/B -Audi A6/B -BMW serie 3/B -Focus/B -Mercedes Classe C/B -Vectra/B -Rover 75/B -Skoda Octavia/B
Sport	-Aston Martin/S -Ferrari/S -Honda NSK/S -Lamborghini/S -MaseratiGT/S -Mercedes SL/S -Porsche/S
Luxe	-Alfa 166/L -Audi A8/L -BMW serie 5/L -BMW serie 7/L -Lancia K/L -Mercedes Classe E/L -Mercedes Classe S/L -Passat/L

TAB. 1 – *Partition a priori du jeu Car models.*

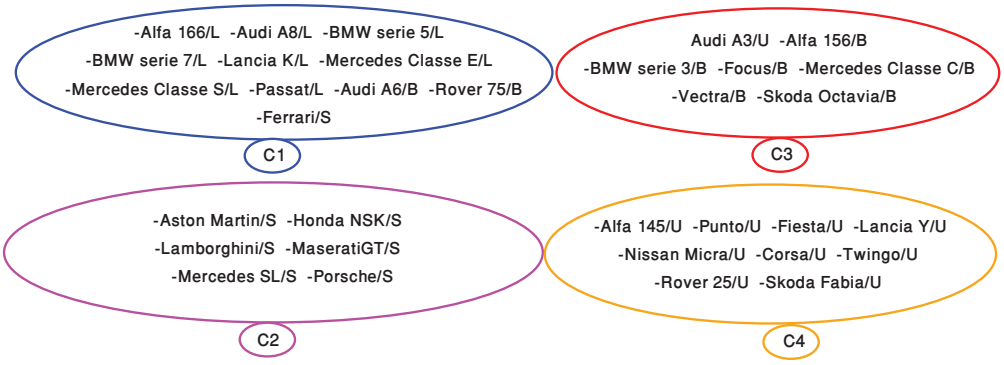

FIG. 2 – *Répartition des modèles de voitures sur les 4 neurones de la carte.*

Tous les modèles de la classe Luxe sont affectés au premier neurone C_1 de la carte. La plupart des modèles de la classe Sport sont affectés au deuxième neurone C_2 de la carte, sauf le modèle Ferrari/S qui est affecté au neurone C_1 voisin du neurone C_2. La majorité des modèles de la classe Berline sont affectés au troisième neurone C_3 sauf les deux modèles Audi A6/B et Rover 75/B qui sont alloués au neurone C_1 voisin du neurone C_3. Presque tous les modèles de la classe Utilitaire sont affectés au neurone C_4 à l'exception du modèle Audi A3/U qui appartient au neurone C_2 voisin du neurone C_4. Ce qui nous mène à conclure que seulement 4 modèles parmi les 33 sont mal classifiés.

Dans le but de donner des résultats plus précis, nous avons évalué les résultats de la classification avec deux critères d'évaluation externes d'une partition : l'indice de Rand corrigé (*CR Index*) (Hubert et Arabie, 1985) et le taux global d'erreur de classification *OERC* qui donne le pourcentage des vecteurs de données mal classifiés. Nous avons obtenu pour l'indice de Rand corrigé une valeur de 0,693 et pour le taux global d'erreur de classification la valeur 12,12%.

Pour comparer la méthode proposée avec d'autres méthodes de classification de données intervalles, nous avons choisi deux méthodes. La première méthode consiste à étendre l'algorithme d'apprentissage en mode différé des cartes auto-organisatrices proposé par Kohonen sur des données de type intervalle en utilisant la distance de Hausdorff entre vecteurs d'intervalles (intSOM_Hausdorff) (Hajjar et Hamdan, 2011a). Dans ce cas, La normalisation des données se fait en calculant une mesure de dispersion conforme avec la distance de Hausdorff (Chavent et Saracco, 2008) avec des paramètres de la carte choisis de la façon suivante :

— le nombre de neurones est $K = 4$ arrangés suivant une grille carrée,
— le rayon initial est $\sigma_{init} = 1$ et le rayon final est $\sigma_{final} = 0,1$,
— le nombre total d'itérations est $totalIter = 100$.

La deuxième méthode (IABSOM-L1), proposée par De Carvalho et al. (2012), consiste à étendre l'algorithme d'optimisation en mode différé des cartes topologiques aux données de type intervalle en utilisant une distance adaptative entre vecteurs d'intervalles basée sur la distance city-block. Le tableau 2 donne le *Corrected Rand Index* et le *OERC* de la méthode proposée dans ce papier et des deux méthodes utilisées comme comparaison.

Nous pouvons donc conclure que la méthode intSOM_DYN_L1 proposée dans ce papier permet une meilleure classification du jeu *Car models* que la méthode IABSOM-L1, alors que des résultats comparables ont été obtenus en appliquant la méthode intSOM_Hausdorff.

Méthode	Indice de Rand corrigé *(CR index)*	Pourcentage des observations mal classifiées (OERC)
intSOM_DYN_L1	0,693	12,12%
intSOM_Hausdorff	0,693	12,12%
IABSOM-L1	0,477	15,2%

TAB. 2 – *Évaluation de la partition obtenue du jeu* Car models *et comparaison avec d'autres méthodes.*

4.2 Jeu de données *City Temperatures*

Le jeu de données *City Temperatures* concerne les températures minimales et maximales enregistrées dans 37 villes du monde durant les 12 mois de l'année (De Carvalho et Pacifico, 2011). Ce jeu est classifié à l'aide d'une carte auto-organisatrice de 9 neurones arrangés suivant une grille carrée. L'apprentissage de la carte est fait en deux itérations : la première correspond à un rayon de voisinage initial égal à 1 et la deuxième correspond à un rayon de voisinage final égal à 0,1. 50 instances de cet algorithme sont lancées avec à chaque fois une initialisation différente des vecteurs référents, et les résultats produits par l'instance qui permet d'avoir la plus petite valeur du critère G_{intSOM} sont adoptés. Quand le rayon de voisinage est fixé à 1, l'optimisation du critère G_{intSOM} nécessite 5 sous-itérations et quand le rayon de voisinage est réduit à 0,1, l'optimisation du critère G_{intSOM} nécessite 4 sous-itérations. La partition finale du jeu de données est illustrée dans la figure 3. Pour chaque classe, la moyenne et l'écart-type des latitudes sont affichés. Nous remarquons que les villes ayant un climat similaire sont affectées au même neurone ou à un neurone voisin sur la carte. Les villes les plus froides sont affectées au neurone C_1, alors que les villes les plus chaudes sont affectées au neurone C_9 l'opposé du neurone C_1 sur la carte.

La figure 4 montre la répartition des villes sur les neurones de la carte quand la méthode IABSOM-L1 est appliquée au jeu de données *City Temperatures* (De Carvalho et al., 2012). En comparant les figures 3 et 4, nous remarquons que les deux méthodes classifient de la même façon la plupart des villes. La somme des écart-types avec la méthode IABSOM-L1 est de 54,83 alors qu'elle est de 49,63 avec la méthode proposée dans ce papier.

5 Conclusion

Dans cet article, nous avons proposé un algorithme pour l'apprentissage d'une carte auto-organisatrice en mode différé pour les données de type intervalle en optimisant un critère basé sur la distance city-bock. Chaque neurone de la carte est représenté par un vecteur d'intervalles. Les vecteurs de données similaires sont affectés au même neurone, ce qui permet d'obtenir une partition de l'ensemble des observations où chaque classe est représentée par un neurone. En plus, deux neurones voisins sur la carte représentent des vecteurs de données proches dans l'espace d'entrée. La méthode proposée a été testée sur deux jeux de données de type intervalle réelles. Le premier jeu de données regroupe des modèles de voiture décrits par des variables intervalles, et le deuxième concerne les températures minimales et maximales enregistrées dans 37 villes du monde. Dans ce contexte, en comparant la méthode proposée à d'autres méthodes, nous avons obtenu des résultats encourageants.

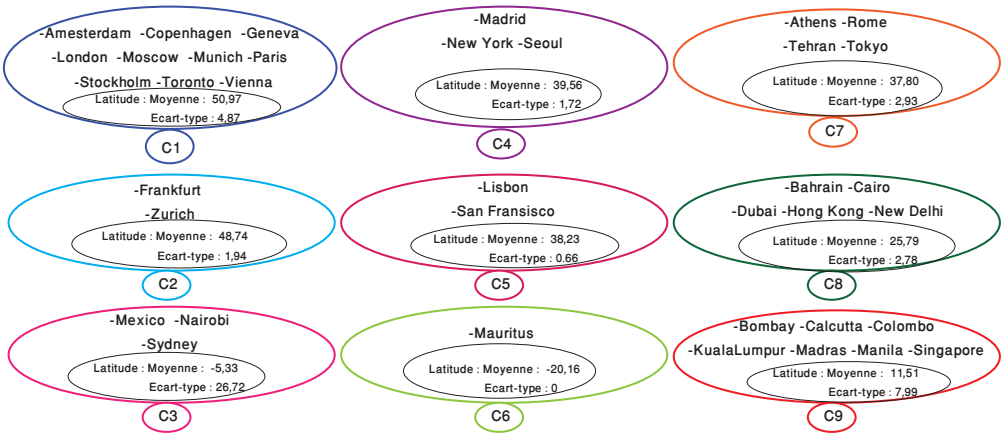

FIG. 3 – *Répartition des villes du monde sur les 9 neurones de la carte avec la méthode proposée.*

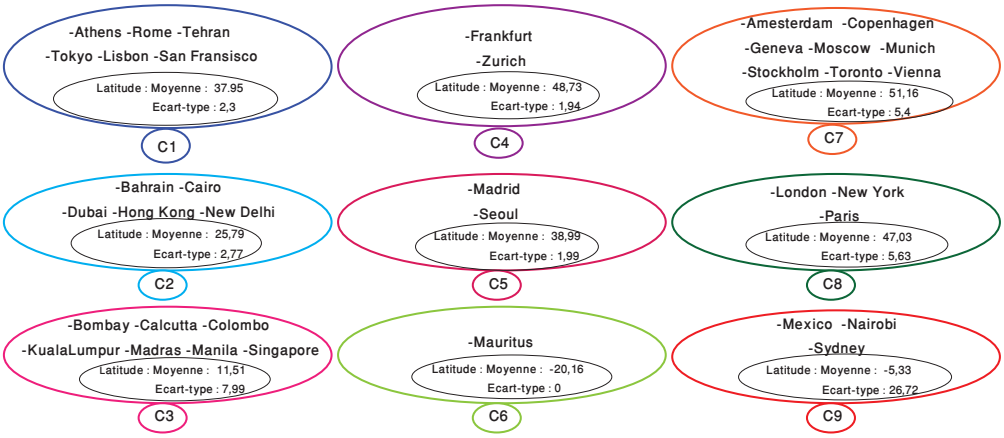

FIG. 4 – *Répartition des villes du monde sur les 9 neurones de la carte avec la méthode IABSOM-L1 (De Carvalho et al., 2012).*

Une perspective de ce travail sera de procéder à une classification en deux étapes. La première étape consiste à projeter l'ensemble des observations sur une large carte auto-organisatrice contenant un grand nombre de neurones, la deuxième étape consiste à classifier les prototypes de la carte pour achever la classification finale des observations. Cette méthode est surtout préconisée quand le nombre de classes n'est pas connu à l'avance et où le nombre de classes optimal est choisi en calculant un critère interne d'évaluation d'une partition comme le critère de Davies-Bouldin (Hajjar, 2014).

A Annexe : Médiane pondérée

Soit à minimiser la fonction suivante :

$$F(w) = \sum_{i=1}^{n} m_i |x_i - w| \tag{9}$$

avec $x_i, w \in I\!R$ et m_i le poids associé au point x_i.

Les dérivées à gauche et à droite de $F(w)$ sont données par :

$$F'_-(w) = \sum_{i:x_i>w} m_i - \sum_{i:x_i\leqslant w} m_i \tag{10}$$

$$F'_+(w) = \sum_{i:x_i\geqslant w} m_i - \sum_{i:x_i<w} m_i \tag{11}$$

Du fait que F est convexe et linéaire par morceaux, une condition nécessaire et suffisante pour que w minimise F est :
$F'_-(w) \leqslant 0$ et $F'_+(w) \geqslant 0$ (Gurwitz, 1990). Ceci entraîne que :

$$\sum_{i:x_i<w} m_i \leqslant \sum_{i:x_i=w} m_i + \sum_{i:x_i>w} m_i \tag{12}$$

$$\sum_{i:x_i>w} m_i \leqslant \sum_{i:x_i=w} m_i + \sum_{i:x_i<w} m_i \tag{13}$$

Donc, w qui minimise F est le point x_i tel que la somme des poids des points qui le précèdent est inférieure ou égale à la somme de son poids et ceux des points qui le succèdent, et la somme de son poids et ceux des points qui le précèdent est supérieure ou égale à la somme des poids des points qui le succèdent (en supposant que les $x_i, i \in \{1, \ldots, n\}$ sont triés par ordre croissant). w est appelé alors la *médiane pondérée* des $x_i, i \in \{1, \ldots, n\}$ (Gurwitz, 1990).

Références

Anouar, F., F. Badran, et S. Thiria (1997). Cartes topologiques et nuées dynamiques. In S. Canu, S. Thiria, O. Gascuel, et Y. Lechevalier (Eds.), *Statistiques et méthodes neuronales*, pp. 190–206. Paris : Dunod.

Billard, L. et E. Diday (2000). Regression analysis for interval-valued data. In H. Kiers, P. Groenen, J.-P. Rasson, et S. M. (Eds.), *Data Analysis, Classification, and Related Methods, Proc. IFCS'2000, Namur, Belgium*. Springer Verlag.

Bock, H. H. (2003). Clustering methods and Kohonen maps for symbolic data. *Journal of the Japanese Society of Computational Statistics 15*(2), 217–229.

Bock, H.-H. (2008). Probabilistic modeling for symbolic data. In P. Brito (Ed.), *COMPSTAT 2008*, pp. 55–65. Physica-Verlag HD.

Cabanes, G., Y. Bennani, R. Destenay, et A. Hardy (2013). A new topological clustering algorithm for interval data. *Pattern Recognition 46*(11), 3030–3039.

Cazes, P., A. Chouakria, E. Diday, et Y. Schektman (1997). Extension de l'analyse en composantes principales à des données de type intervalle. *Revue de Statistique Appliquée XIV*(3), 5–24.

Chavent, M. et Y. Lechevallier (2002). Dynamical clustering of interval data : Optimization of an adequacy criterion based on Hausdorff distance. In K. Jajuga, A. Sokolowski, et H. H. Bock (Eds.), *Classification, Clustering and Data Analysis*, Berlin, Germany, pp. 53–60. Springer.

Chavent, M. et J. Saracco (2008). On central tendency and dispersion measures for intervals and hypercubes. *Communications in Statistics - Theory and Methods 37*(9), 1471–1482.

Chouakria, A. (1998). *Extension des méthodes d'analyse factorielle à des données de type intervalle*. Ph. D. thesis, Université Paris 9 Dauphine.

De Carvalho, F., F. De Melo, et P. Bertrand (2012). Batch self-organizing maps based on city-block distances for interval variables. In *Proceedings of the 20th International Conference on Computational Statistics (COMPSTAT 2012)*, pp. 155–166.

De Carvalho, F. et L. Pacifico (2011). Une version batch de l'algorithme SOM pour des données de type intervalle. In *Actes des XVIIIème Rencontres de la Société Francophone de Classification*, pp. 99–102.

De Carvalho, F. A. T., P. Brito, et H. H. Bock (2006). Dynamic clustering for interval data based on L_2 distance. *Computational Statistics 21*(2), 231–250.

De Souza, R. M. C. R. et F. A. T. De Carvalho (2004). Clustering of interval data based on city-block distances. *Pattern Recognition Letters 25*(3), 353–365.

De Souza, R. M. C. R., F. A. T. De Carvalho, C. P. Tenï£¡rio, et Y. Lechevallier (2004). Dynamic cluster methods for interval data based on Mahalanobis distances. In *Proceedings of the 9th Conference of the International Federation of Classification Societies*, Chicago, USA, pp. 351–360. Springer-Verlag.

Denœux, T. et M. Masson (2000). Multidimensional scaling of interval-valued dissimilarity data. *Pattern Recognition Letters 21*(1), 83–92.

El Golli, A., B. Conan-Guez, et F. Rossi (2004). Self-organizing maps and symbolic data. *JSDA Electronic Journal of Symbolic Data Analysis 2*(1).

Gurwitz, C. (1990). Weighted median algorithms for l1 approximation. *BIT 30*(2), 301–310.

Hajjar, C. (2014). *Cartes auto-organisatrices pour la classification de données symboliques mixtes, de données de type intervalle et de données discrétisées*. Ph. D. thesis, École Supérieure d'Électricité (Supélec).

Hajjar, C. et H. Hamdan (2011a). Self-Organizing Map based on Hausdorff distance for interval-valued data. In *IEEE International Conference on Systems, Man, and Cybernetics*, Anchorage, Alaska, pp. 1747–1752.

Hajjar, C. et H. Hamdan (2011b). Self-Organizing Map based on L_2 distance for interval-valued data. In *IEEE International Symposium on Applied Computational Intelligence and Informatics*, Timisoara, Romania, pp. 317–322.

Hajjar, C. et H. Hamdan (2013). Interval data clustering using Self-Organizing Maps based on adaptive Mahalanobis distances. *Neural Networks 46*, 124–132.

Hamdan, H. et G. Govaert (2004). Int-EM-CEM algorithm for imprecise data. Comparison with the CEM algorithm using monte carlo simulations. In *IEEE International Conference on Cybernetics and Intelligent Systems*, Singapore, pp. 410–415.

Hamdan, H. et G. Govaert (2005). Mixture model clustering of uncertain data. In *IEEE International Conference on Fuzzy Systems*, Reno, Nevada, USA, pp. 879–884.

Heskes, T. (1999). *Kohonen Maps*, pp. 303–315. Amesterdam : Elsevier.

Hubert, L. et P. Arabie (1985). Comparing partitions. *Journal of Classification 2*, 193–218.

Kohonen, T. (1984). *Self Organization and Associative Memory, Second Edition*. Springer Verlag.

Kohonen, T. (2001). *Self-Organizing Maps, Third Edition*. Springer.

Noirhomme-Fraiture, M. et P. Brito (2011). Far beyond the classical data models : Symbolic Data Analysis. *Statistical Analysis and Data Mining 4*(2), 157–170.

Rossi, F. et B. Conan-Guez (2002). Multi-layer perceptron on interval data. In K. Jajuga, A. Sokolowski, et H. H. Bock (Eds.), *Classification, Clustering, and Data Analysis*, pp. 427–436. Berlin, Germany : Springer.

Summary

Symbolic data can better represent the measurements issued from real applications providing thereafter a higher level of knowledge than simple values. Interval data are a kind of symbolic data that are used to represent, as the case may be, the variability or the uncertainty in the observed measurements. In this paper, we propose an algorithm to train the self-organizing maps in batch mode in order to classify interval data while preserving their topology. The learning of the map is done by optimizing an adequacy criterion based on city-block distance. The proposed method is tested and compared to other clustering methods for interval data using two real interval data sets.