TP n°4 IA : Cartes de Kohonen Apprentissage auto-supervisé

Ishak AYAD

1 Cartes topologiques de Kohonen

Apparues dans les années 60, les méthodes neuronales inspirées du fonctionnement du cerveau humain ont suscité l'intérêt des mathématiciens et notamment des statisticiens. En général, on peut les utiliser pour réaliser deux types d'apprentissage statistique :

— Apprentissage supervisé : classification et régression
— Apprentissage non supervisé : classification automatique

Dans ce TP, j'ai implémenté des cartes auto-organisatrices Kohonen (Self-Organizing Map, SOM) appelées aussi cartes topologiques de Kohonen. Il s'agit d'une méthode neuronale développée par Teuvo Kohonen en 1982, utilisée pour réaliser des tâches de classification automatique.

Elles sont composées d'éléments simples (ou neurones formels) assemblés en réseaux de neurones. Leur fonctionnement est fortement influencé par la connexion des éléments entre eux. Les cartes topologiques sont souvent utilisées pour analyser des données observées dont on cherche à comprendre la structure.

1.1 Principe de la méthode

La méthode proposée par Kohonen consiste à projeter l'espace des données observées (de grande dimension) sur un espace de faible dimension ; en général 1, 2 ou 3D, appelé carte. Cette dernière est formée d'un ensemble de neurones connectés entre eux selon la notion de voisinage.

Chaque neurone possède :

— Des coordonnées fixes sur l'espace de la carte
— Des coordonnées adaptables sur l'espace des données appelées aussi vecteurs référents, responsables d'une zone de cet espace

Cette projection doit respecter la topologie des données. Elle est basée sur des méthodes de quantification vectorielle qui cherchent à partitionner l'ensemble des observations disponibles en groupements similaires par des algorithmes d'apprentissage. Ces groupements sont caractérisés par leurs structures de voisinage qui peuvent être matérialisées à l'aide d'un espace discret appelé « carte topologique ». Cet espace forme un treillis de faible dimension sur lequel les structures de voisinages sont prises en considération par le modèle. Le choix de la topologie dépend de la nature des problèmes. Les figures ci-dessous montrent quelques exemples de topologies tirés du package R kohonen.

```c
//structure de donnees d'un neuron
typedef struct {
    //vecteurs referents
    double *weights;
    //coordonnees fixes sur la carte
    int x, y;
    int num_weights;
}Neuron;

//structure de donnees representant une carte de kohonen
typedef struct {
    Neuron *lattice;
    int latice_size;
    //nombre de ligne/colonne
    int sideX, sideY;
}Map;

```

Listing 1 – Structures de données nécessaire pour représenter une carte de kohonen

2 Algorithme

Une fois la topologie de carte et les fonctions d'activation choisies, les vecteurs référents sont obtenus par apprentissage à partir des données. Je vais maintenant présenter l'algorithme utilisé pour l'apprentissage de la carte.

— Etape 0 : Initialisation des vecteurs référents W.

— Etape 1 : A chaque itération :

On présente à l'entrée de la carte, un exemple d'apprentissage X(n) choisi.
On compare l'exemple à tous les vecteurs référents, le neurone gagnant i est celui dont le vecteur.
référents $W_i(n)$ est le plus proche de l'entrée X(n) :

$$i^* = argmin(d(W_i(n), X(n)) \quad (1)$$

Où i désigne tour à tour chaque neurone du réseau et Wi son vecteur référent.

— On évalue le voisinage du neurone gagnant

$$K_{i^*}^T(i) = K^T(\delta(i, i^*)) \quad (2)$$

— Mise à jour des poids pour tous les neurones de la carte, l'adaptation est d'autant plus forte que les neurones sont voisins de i^* :

$$W_i(n+1) = W_i(n) + \alpha(t) * K_{i^*}^T(i)[X(n) - W_i(n)] \quad (3)$$

— Pour tout i appartenant au voisinage de i^*

— α est un paramètre d'apprentissage appelé pas d'apprentissage, qui décroît en fonction de n, et est inférieur à 1.

Cette fonction décroissante α est un élément important car permettant de trouver un compromis entre la vitesse de convergence et la qualité du dé-pliement de la carte.

— Utilisation d'un critère d'arrêt : nombres d'itérations, différence entre les vecteurs poids, erreur de quantification ...

Les phases 1, 2 et 3 s'appellent respectivement :

Compétition : Après sélection d'un exemple dans la banque de données, on cherche le neurone qui lui ressemble le plus. Ce neurone dit « neurone gagnant » est souvent noté BMU (Best Matching Unit).

Coopération : Dans cette étape, on détermine le voisinage du neurone gagnant. Il désigne la région de la carte la plus active et qui s'approche le mieux, au sens de la distance utilisée, de l'observation. La taille du voisinage du neurone gagnant est contrôlée par le rayon d'apprentissage.

Adaptation : On modifie le neurone gagnant pour qu'il ressemble plus à l'exemple puis on diffuse l'information sur son voisinage. Cette modification se fait à l'aide de la fonction d'activation qui permet de contrôler l'influence du neurone gagnant sur son voisinage.

3 Exercice n°1

Dans cet exercice nous supposerons le réseau de neurones de taille fixe : 20 neurones. Nous supposerons que l'ensemble des données est composées de 20 vecteurs de deux dimensions. Chaque neurone possède par conséquence deux poids en entrée. Nous utiliserons une fonction ϕ(dist) définit tel que dvp=1 et dvn=2, soit $\phi(0)$=1 $\phi(1)$=λ et $\phi(2)$=-β où $\lambda = 0.5$ et $\beta = 0.2$.

```
1  //Un vecteur d'entree de taille size
2  typedef struct{
3    int size;
4    double* input;
5    //controleur pour checker si on a tirer cette DATA
6    int drawn;
7  }DATA;
8
9  //Un ensemble de donnees compose de size * DATA de taille num_in
10 typedef struct{
11    int size;
12    int num_in;
13    DATA* entries;
14 }TRAINING_DATA;
15
```

Listing 2 – Structure de données nécessaire à la représentation d'un vecteur d'entrée

3.1 Avantages et inconvénients

Les cartes topologiques de Kohonen présentent de nombreux avantages entre autres :

- C'est une méthode facile à mettre en œuvre, à expliquer
- regroupement automatique des données
- Avec ces nombreuses sorties graphiques, les résultats sont faciles à comprendre et à expliquer

Parmi les limites de cette approche, on peut noter :

- La convergence vers une solution globale n'est pas assurée
- Parfois les résultats ne correspondent pas à celle souhaitée

Avec un dvp et un dvn bien plus large les valeurs en sortie vont dépasser les valeurs en entrées, et donc les valeurs de sorties seront erronés.

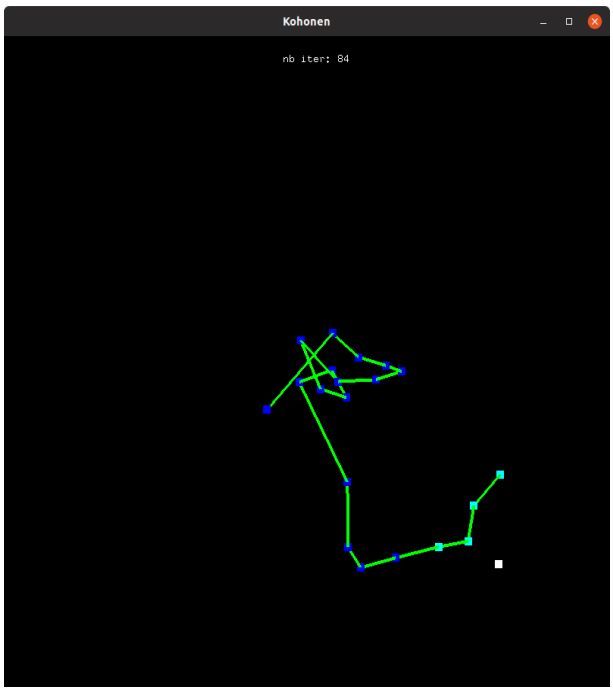

FIGURE 1 – Résultat du premier exercice, avec des entrées entre [0,200] on remarque que la carte topologique se rapproche de plus en plus vers des valeurs comprises entre [0,200]

4 Exercice n°2

Le problème du voyageur de commerce est un problème d'optimisation historique qui est difficile à résoudre de façon peu coûteuse et optimale.

Le problème du voyageur de commerce est un problème classique, aussi connu sous le nom de travevelling salesman problem. Le voyageur de commerce doit effectuer la tournée de ses magasins le plus rapidement possible. Il doit donc déterminer un chemin passant par toutes les villes qui soit le plus court possible.

Combien de neurones sont nécessaires ici pour résoudre le problème : On doit disposer de minimums 2×le nombres de villes, car sachant que l'algorithme (2) vas modifier les poids des voisins selon la fonction ϕ, et donc si on prend (un nombre de neurones ≤ le nombre de villes) au moment de modification des poids on peut changer des cordonnées qui sont déjà correctes, donc pour évité cela on pause le nombre de nuerons = 2× le nombre de villes.

Quel est la partie du réseau qui sera le chemin à suivre pour notre voyageur decommerce : C'est notre carte topologique formée de neurones, donc le graphe de neurones est notre chemin à suivre.

Remarque : il s'applique de préférence à un nombre de villes inférieur à unes ou deux centaines. Au-delà, lorsque les villes sont trop proches des unes des autres, cet algorithme retourne une solution où les erreurs apparaissent de manière évidente.

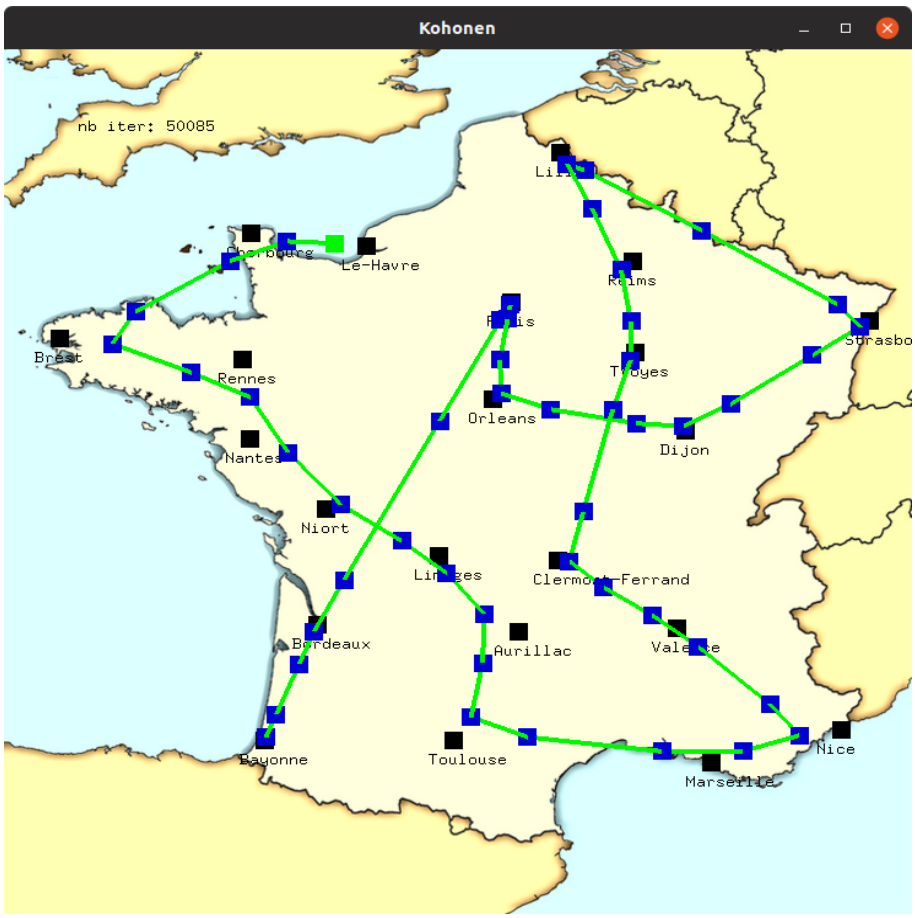

FIGURE 2 – Résultat du deuxième exercice, avec des poids initiés entre [50,750] on remarque qu'on obtiens un résultat assez rapidement que quand on lance avec des poids en 0.

L'influence de l'initialisation des poids sur le résultat c'est que le temps de calcul de la meilleure solution serra optimale (12000 itérations en moyenne), et si on initialise nos poids aléatoirement on aura un temps de calcul un peu plus élevé (20100 itérations en moyenne).

5 Exercice n°3

On souhaite compresser une image en couleur, pour ce faire nous allons diminuer le nombre de couleurs présentes dans l'image pour n'en garder qu'un nombre déterminé.Pour cet exercice nous commencerons par compresser l'image avec 16 couleurs.

Comment utiliseriez-vous Kohonen pour résoudre ce problème : On utilise une carte Kohonen 2D, $(\frac{nombredecouleur}{2})^2$.

Pour l'entraînement je prends notre image de base en convertissant les pixels en vecteurs d'entraînements, et en ce qui concerne la reconstruction je regarde pour chaque pixel de l'image originale la meilleure valeur sur ta carte c'est-à-dire la plus proche je fais une distance euclidienne et puis je prends la meilleure couleur et je la mets dans l'image à reconstruire.

FIGURE 3 – Résultat de la compression avec 16 couleurs après 10000 itération

Vous pouvez changer le nombre de couleurs souhaité pour la compression sur le fichier Kohonen_compression.c il s'agit de la constante NB_COLOR.

Pour compresser votre image laisser l'apprentissage se dérouler et puis taper quand vous voulez le touche [C] l'apprentissage va s'arrêter et le processus de compression va se lancer et il va prendre du temps selon la taille de l'image et le nombre de couleurs utilisé pour la compression.

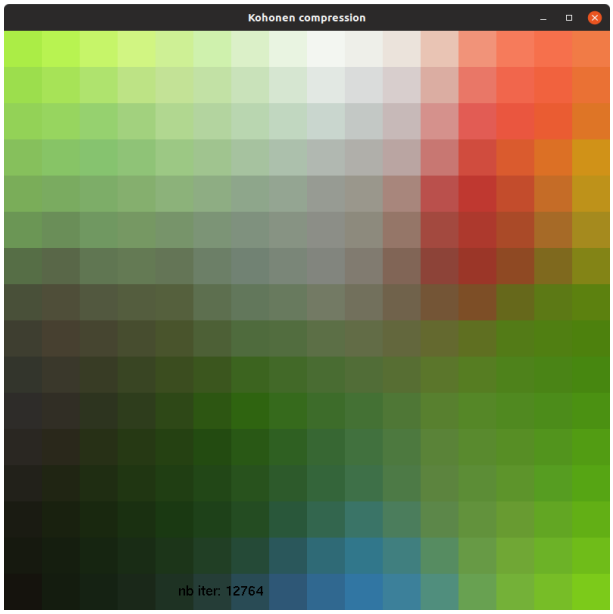

FIGURE 4 – Affichage des valeurs des nuerons pour une compression de 32 couleurs.

6 Conclusion

Contrairement aux méthodes classiques qui ont montré leurs limites, les réseaux de neurones ont montré leurs tendances à s'adapter à des problèmes complexes grâce à leur grande capacité de calcul et d'apprentissage.

Ils sont l'objet d'utilisation dans les différents domaines tels que : La reconnaissance des formes et le traitement des images.

Le grand avantage caractérisé dans les réseaux de neurones de Kohonen est que ces derniers sont léger en coût et en calcule et sont portable dans différents domaines, ce qui les rend les plus simples à utiliser et les plus rapides.