Si la température était nulle $T = 0$, la mise à jour deviendrait déterministe et la machine de Boltzmann, si elle n'a pas de nœud caché deviendrait un réseau de Hopfield. En pratique on fait tendre la température vers zéro sans jamais l'atteindre.

Dans une configuration stable, la corrélation entre les états des nœuds quand les nœuds visibles sont fixés est notée :

$$\rho_{ij}^{+} =< s_i, s_j >^{+} .$$

Il s'agit de la moyenne (dite moyenne d'ensemble) des valeurs des états s_i et s_j sur le nombre de données. On réitère ce processus de mise à jour des états en ne fixant que les nœuds visibles d'entrée, et en laissant évoluer librement les valeurs des nœuds cachés et de sortie. La corrélation dans cette phase d'évolution libre est notée :

$$\rho_{ij}^{-} =< s_i, s_j >^{-} .$$

Après le calcul de ces 2 types de corrélations, les poids sont mis à jour :

$$\Delta w_{ij} = \eta(\rho_{ij}^{+} - \rho_{ij}^{-}).$$

où η est le pas d'apprentissage. On répète le processus de mise à jour des poids jusqu'à ce que $\Delta w_{ij} = 0$ pour chaque poids. L'énergie E du réseau

$$E = -\frac{1}{2}\sum_i \sum_j w_{ij} s_i s_j$$

est minimale et l'entraînement terminé. Les valeurs des poids w_{ij} sont alors utilisées pour déterminer les valeurs des nœuds de sortie quand on présente une forme sur les nœuds visibles d'entrée.

Une machine de Boltzmann restreinte est un réseau où les connexions entre nœuds de même type sont supprimées : il n'y a pas de connexion entre nœuds visibles et entre nœuds cachés. Par rapport à une machine de Boltzmann entièrement connectée, la machine restreinte a plutôt une structure en couches. On entraîne comme précédemment mais les nœuds visibles sont utilisés pour mettre à jour les nœuds cachés, puis les nœuds visibles sont mis à jour de nouveau, puis encore les nœuds cachés. Cela permet d'entraîner le réseau de manière plus efficace, plus rapide, et d'entraîner des réseaux de taille importante. Cela permet aussi d'empiler plusieurs réseaux de ce type, les sorties d'un réseau constituant les entrées du suivant. Cette approche est notamment utilisée pour l'extraction automatique de caractéristiques (*feature detector*) dans les réseaux de neurones profonds (DNN *Deep Neural Nets*).

8.4.6 Cartes auto-organisatrices de Kohonen

Les cartes de Kohonen (ou SOMs *Self Organizing Maps*) permettent de visualiser des données multidimensionnelles dans un espace de dimension restreinte. Il y a donc une relation avec la quantification vectorielle. Une carte de Kohonen comprend un ensemble de nœuds placés sur une grille de point. Un nœud est en relation physique (ou logique) avec ses voisins mais il n'y a pas de lien avec les nœuds voisins. A chaque nœud, repéré par ses coordonnées (x, y) de la grille, on associe un vecteur "poids", de

Algorithme 15 Apprentissage d'un réseau de Boltzmann

1. Initialiser les poids du réseau w_{ij} aléatoirement $\in U[-1,1]$ (loi uniforme)

Phase contrainte

2. Pour chaque exemple d'apprentissage, utiliser ses valeurs pour fixer les nœuds d'entrée et de sortie.

3. Recuit simulé avec échantillonnage de Gibbs
pour $T = T_0$ à T_{final}
Mettre à jour chaque état s_j
calculer $v_j = \sum_{i=1, i\neq j}^{N} w_{ij} s_i$ et $P(v_j) = \frac{1}{1+e^{-2v_j/T}}$
tirer aléatoirement $r \in [0,1]$ suivant une loi uniforme :
si $r < P(v_j)$ alors $s_j = 1$
sinon $s_j = -1$
jusqu'à ce que les états ne changent pas.

3b. conserver les états obtenus pour chaque exemple.

4. calculer les corrélations d'états pour la phase contrainte : $\rho_{ij}^{+} =< s_i, s_j >^{+}$
(moyenne du produit des états sur les exemples d'apprentissage)

Phase libre

5. fixer uniquement les nœuds d'entrée, et laisser les nœuds cachés et de sortie libres.
6. mettre à jour les états comme dans l'étape 3 jusqu'à obtenir un état stable.

Mise à jour des poids

7. calculer les corrélations d'états pour la phase libre $\rho_{ij}^{-} =< s_i, s_j >^{-}$,
(moyenne du produit des états sur les exemples d'apprentissage)

8. mettre à jour les poids
$w(n+1) = w(n) + \Delta w_{ij}$ où
$\Delta w_{ij} = \eta(\rho_{ij}^{+} - \rho_{ij}^{-})$.

9. si $\Delta w_{ij} \neq 0$, répéter depuis 2.

dimension d appelé aussi vecteur référent. Les vecteurs poids vont évoluer pendant la phase d'apprentissage pour représenter au mieux les données d'entrée.

A partir d'un signal d'entrée, on recherche au niveau global le nœud le plus proche, appelé le neurone vainqueur ou BMU (*Best Matching Unit*) qui représente le mieux ce signal. Les vecteurs poids de tous les nœuds autour du nœud BMU sont ajustés à des valeurs qui se rapprochent de celles du signal d'entrée. Pendant la phase d'apprentissage, le montant de l'ajustement décroît graduellement avec la distance au nœud BMU, et avec le temps. Au cours de ce processus les poids s'auto-organisent pour offrir une représentation des données d'entrée qui tienne compte de leur proximité. Les étapes d'entraînement d'une carte de Kohonen sont dans l'algorithme 16.

L'algorithme est sensible aux dimensions de la grille, à celle de la fenêtre de voisinage autour du BMU, et au pas d'apprentissage. Un pas et une pondération de la distance qui décroît de 10% de sa valeur au cours de l'apprentissage, donne généralement de bons résultats.

L'entraînement de la carte n'utilise pas d'étiquettes de classe. On est donc dans le cadre de l'apprentissage non supervisé. Cette approche présente aussi des similarités avec l'approche de classification automatique (voir chapitre 6). Si on attribue des étiquettes aux nœuds de la grille, une métrique basée sur la distance peut être appliquée à un vecteur requête sur tous les nœuds étiquetés et on obtient ainsi un classement.

Algorithme 16 Algorithme de Kohonen pour une grille bi-dimensionnelle.

Initialiser D_0, L_0 et λ_L

1. Initialiser chaque nœud (x, y), avec un vecteur poids aléatoire $W_0(x, y)$, de la même dimension que les vecteurs d'apprentissage.
2. Choisir au hasard un vecteur d'apprentissage V_n
3. Déterminer quel nœud (x, y) a le poids le plus proche du vecteur d'apprentissage présenté. Appelons ce nœud le Best Matching Unit (BMU) :
   (x_0, y_0).
4. Déterminer la taille du voisinage du BMU à l'itération n.
   $D_n = D_0 exp(-\frac{n}{\lambda_D})$
5. Déterminer la pondération de la distance de voisinage, proportionnelle à la distance au nœud BMU
   $\Theta(x, y) = \begin{cases} 1 & \text{si } |(x - x_0)^2 + (y - y_0)^2| < D_n^2, \\ 0 & \text{sinon} \end{cases}$
6. Déterminer le poids d'apprentissage pour l'itération courante
   $L_n = L_0 exp(-\frac{n}{\lambda_L})$
7. Ajuster les poids aux nœuds dans le voisinage du BMU, de manière à ce qu'ils soient plus proches du vecteur présenté.
   $W_{n+1}(x, y) = W_n(x, y) + \Theta(x, y) L_n(V_n - W_n(x, y))$
8. Itérer N fois depuis l'étape 2

8.5 Exercices

8.5.1 Exercice sur ordinateur : visualisation des couleurs par carte de Kohonen

Une application populaire des cartes de Kohonen est l'affichage d'un ensemble de couleurs en 2 dimensions. Ecrivez le programme d'apprentissage d'une carte de Kohonen de taille 40×40 en vue de visualiser les couleurs rouge, vert, vert foncé, bleu, bleu foncé , jaune, magenta, orange. utiliser les valeurs RGB comme caractéristiques.
1) Utiliser une pondération de la distance de type gaussien, un voisinage initial de taille 20, un pas d'apprentissage initial de 0.1 et faire $N = 1000$ itérations. Faire décroître la taille du voisinage avec une constante de temps égale à N/3.

2) Utiliser une pondération de distance de type échelon circulaire, un voisinage initial de taille 20, un pas d'apprentissage initial de 0.1 et faire $N = 1000$ itérations Faire décroître la taille du voisinage avec une constante de temps égale à N/3. Comparer les résultats avec la question 1).

8.5.2 Exercice sur ordinateur : visualisation de caractères par carte de Kohonen

Utilisez l'ensemble d'apprentissage de caractères (annexe B) pour construire une carte de Kohonen bi-dimensionnelle de taille 60×60. Après apprentissage, prenez l'ensemble de test et trouver pour chaque caractère de cet ensemble le point (x,y) de la carte le plus proche. Puis attribuer au caractère de test l'étiquette du caractère le plus proche de la carte.

8.5.3 Exercice sur ordinateur : mémoire adressable

1) Enregistrer un exemplaire d'image de 0 et un de 7 dans un réseau de Hopfield. Ce images seront prises dans l'ensemble d'apprentissage de l'annexe B.2. On remplacera les valeurs de 0 (fond) par -1.

2) Entrez ensuite des images de caractères de classe 0 et 7, prises dans l'ensemble de test. Visualiser la sortie du réseau pour chaque caractère entré.

8.6 Solutions des exercices

Solution de l'exercice 8.5.1

La grille entraînée avec une pondération de distance de type gaussien aura des frontières plus douces que celle entraînée avec une pondération de type échelon.

Solution de l'exercice 8.5.2

Les réponses vont varier. Un exemple de sortie est montré en figure 8.14.

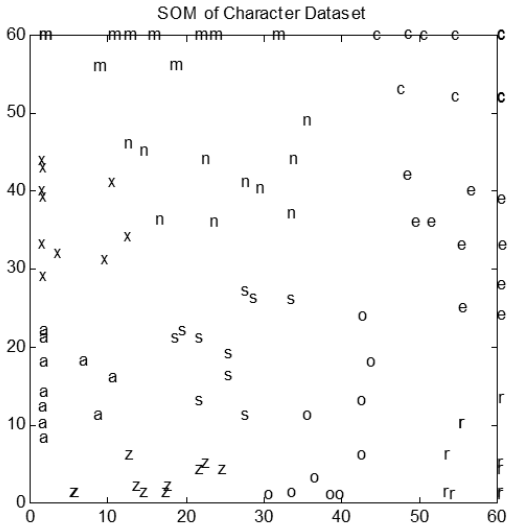

FIGURE 8.14 – Exemple de carte de Kohonen pour les données caractères.

Solution de l'exercice 8.5.3

1) On veut stocker en mémoire une image de zero et une image de sept. Ces images sont de taille 5 lignes $\times$ 28 colonnes. Le nombre de nœuds du réseau est donc égal à 140. On met chaque image d'apprentissage sur une ligne de la matrice xa, les lignes des images de chiffres étant mises bout à bout. Les valeurs de pixels représentant le fond des images, de valeur 0, sont décalés sur la valeur -1.

```
m=5; % taille des images de chiffres
n=28;
 num0=double(xa0_1');
 xa(1,:)=num0(:);
 num7=double(xa7_1');
 xa(2,:)=num7(:);
  % affichage
 subplot(1,2,1);image(255*reshape(xa(1,:),n,m)');
 subplot(1,2,2);image(255*reshape(xa(2,:),n,m)');
 % mettre des -1 a la place des zeros
xa(find(xa==0))=-1;
```

Apprentissage de la matrice de poids w. Elle est symetrique et à diagonale nulle :

```
w=zeros(nrow*ncol,nrow*ncol);
for i=1:size(xa,1)
    input_vec=xa(i,:);
    w=w+input_vec'*input_vec;
end
for i=1:nrow*ncol
    w(i,i)=0;  % pas de liaison d'un noeud sur lui meme
end
```