SOM & programmation GPGPU

SOM THEORIE

PRINCIPE :
Une carte auto-adaptative est composée d'un réseau de neurone de dimension n, ici 2. Chaque neurone a une position propre et contient un vecteur de poids, donné aléatoirement, de même dimension et de même type que le vecteur d'entrée.

APPRENTISSAGE :

Activité

Les neurones vont être mis en « compétition » pour savoir quel vecteur de poids se rapproche le plus du vecteur d'entrée. Pour cela, plusieurs calculs de distance sont possibles mais pour l'instant, nous n'avons utilisé pour l'instant que la distance Euclidienne. On a :

$$d = \sqrt{\sum_{i=0}^{i=k} (V_i - W_i)^2}$$

avec V le vecteur d'entrée, W le vecteur de poids du neurone et k la taille du vecteur de poids du neurone.

Alpha, Beta : initialisation, taux et période

Une fois le vainqueur déterminé, deux nouvelles variables rentrent en jeu, données par l'utilisateur :

- Alpha qui est l'amplitude du changement du poids. Le vecteur de poids du neurone vainqueur et celui de ses voisins vont être modifiés pour mieux se rapprocher du vecteur d'entrée précédemment donné. La modification est proportionnelle à la distance entre un neurone et le vainqueur. $\alpha \in]0; 1]$
- Beta qui définit l'étendu du voisinage. Lors de la modification du vecteur de poids, les voisins du vainqueur vont recevoir des modifications, il faut donc définir qui sont ses voisins. $\beta \in]0; \max(largeur, longueur)]$

Au fur et à mesure des itérations de l'apprentissage, alpha et beta vont décroitre pour une meilleure précision selon un taux donné par l'utilisateur. Dans les calculs suivants, $\theta_1 \in]0; 1]$ et $\theta_2 \in]0; 1]$ sont respectivement les taux de modification de alpha et beta, avec $\theta_1 < \theta_2$. De plus alpha et beta peuvent ne pas décroître à chaque itération, mais selon une période.

Amplitude (alpha)

Alpha décroît tel que :

$$\alpha(t) = \alpha * e^{(-\frac{t}{\theta_1})}$$

avec θ_1 et α choisi par l'utilisateur, et t le nombre d'itérations réalisées.

On remarque : $\lim_{t \to +\infty} \alpha(t) = 0 < \lim_{t \to 1} \alpha(t)$

Distance voisinage (beta)

On a le rayon du cercle de voisinage qui suit la fonction suivante :

$$\beta(t) = \beta * e^{(-\frac{t}{\theta_2})}$$

avec θ_2 et β choisi par l'utilisateur, et t le nombre d'itérations réalisées.

On remarque : $\lim_{t \to +\infty} \beta(t) = 0 < \lim_{t \to 1} \beta(t)$

Modification proportionnelle selon la distance entre un neurone et le vainqueur

Plus un neurone sera éloigné du neurone vainqueur, plus la modification de son vecteur de poids sera moindre. On a :

$$\varphi(t) = e^{(-\frac{d^2}{2\beta^2(t)})}$$

La distance est calculée avec les coordonnées des neurones.

On remarque :

$\lim_{dist \to +\infty} \varphi(t) = 0$ et $\lim_{dist \to 0} \varphi(t) = 1$

$\lim_{\beta(t) \to 0} \varphi(t) = 0 < \lim_{\beta(t) \to \beta} \varphi(t)$

Modification des poids

$$W(t+1) = W(t) + \alpha(t)\varphi(t)(V(t) - W(t))$$

Ainsi, la zone du maillage qui correspond au vecteur d'entrée est optimisée pour mieux y ressembler. Après beaucoup d'itérations, le réseau va se stabiliser et l'apprentissage s'arrêtera quand alpha sera proche de 0.