Inspiration et modélisation du vivant - M2 Informatique AVR
TD : DNF et attention visuelle

Introduction

L'objectif de ce TD est de vous faire mieux comprendre les mécanismes internes aux Champs Neuronaux Dynamiques (Dynamic Neural Fields, DNF) et de vous les faire appliquer.

Quelques consignes :

— Vous travaillerez par groupes de 2 étudiants.
— Vous utiliserez de préférence le langage `python`.
— Vous pourrez avoir besoin des bibliothèques numpy, scipy, matplotlib et cv2. Vous pouvez les installer sur vos machines avec la commande suivante : `pip3 install numpy scipy matplotlib opencv-python`
— L'ensemble de vos valeurs devront être normalisées entre 0 et 1.
— Testez vos fonctions au fur et à mesure.

1 CNFT

On considère dans un premier temps des DNF dérivés de la théorie des champs neuronaux continus (CNFT), en utilisant l'équation suivante :

$$\tau \frac{du(x,t)}{dt} = -u(x,t) + \int_{y\in[0,1]^2} f(u(y,t))\omega(||x-y||)dy + \gamma Input(x,t) + h$$

avec le noyau d'interaction calculé comme une différence de gaussiennes

$$\omega(d) = c_{exc}\exp(-\frac{d^2}{2\sigma_{exc}^2}) - c_{inh}\exp(-\frac{d^2}{2\sigma_{inh}^2})$$

et f est par exemple la fonction sigmoïde logistique $\frac{1}{1+e^{-(2x-1)}}$

(en `python` vous pouvez utiliser `scipy.special.expit`).

Une version discrétisée en temps (selon la méthode d'Euler) et en espace ($N \times N$ neurones) donne :

$$u(p,t+dt) = u(p,t) + \frac{dt}{\tau}\left(-u(p,t) + \frac{1}{N^2}\sum_q f(u(q,t))\omega(||p-q||) + \gamma Input(p,t) + h\right)$$

avec $||p-q||$ la distance normalisée entre les positions p et q dans la grille de neurones.

On fournit les fonctions `euclidean_dist(x,y)` et `gaussian(d,sigma)` qui calculent respectivement la distance euclidienne entre deux points de $[0,1]\times[0,1]$, et la fonction gaussienne centrée en 0 et d'écart-type σ en fonction d'une distance d.

1. Programmez une fonction `omega`$(d, c_{exc}, c_{inh}, \sigma_{exc}, \sigma_{inh})$ qui calcule la valeur du noyau d'interaction (en différence de gaussiennes) appliqué à une valeur réelle d.
2. Programmez une fonction `gaussian_activity(p,q,sizes,sigma)` qui génère une "image" contenant deux bulles d'activité gaussiennes (à la manière des exemples vus en cours) respectivement centrées en p et q. L'image sera donnée sous la forme d'un tableau 2D de taille *sizes*. Pensez à normaliser les coordonnées et les amplitudes (pic maximum à 1). Trouvez une valeur `sigma` qui permette de former deux bulles d'activités bien séparées et formées lorsque les positions p et q sont suffisamment éloignées. En `python`, vous pouvez par exemple visualiser ce qui est obtenu en vous inspirant du code suivant :

```
import numpy as np
import matplotlib.pyplot as plt
a = gaussian_activity(...)
plt.imshow(a, cmap=binary, interpolation='nearest')
plt.show()
```

3. Définissez une classe `DNF` correspondant à un champ neuronal 2D, chacun des neurones étant caractérisé par un potentiel (valeur réelle non normalisée). La taille (paramétrable) par défaut de cette carte sera 45×45. Les différents paramètres du DNF seront des attributs de cette classe.
4. Programmez une fonction `update_neuron(position)` qui effectue la mise à jour du neurone à la position indiquée dans le champ neuronal, selon l'équation discrétisée en temps et en espace.
5. Programmez une fonction `update_map()` qui fait évoluer le champ neuronal de façon synchrone pendant `nbiter` itérations de pas de temps dt. L'entrée sera une image statique (1 pixel d'entrée par neurone, les neurones étant organisés de manière rétinotopique). Visualisez le résultat.

   Remarque : il se peut que votre programme soit lent, en raison essentiellement du calcul des interactions latérales. Pour améliorer cela, vous pouvez calculer simultanément les interactions latérales de tous les neuones en utilisant le calcul optimisé de convolution `scipy.signal.fftconvolve`. Attention à vérifier que vous obtenez bien les mêmes résultats qu'avec le calcul que vous avez implanté initialement dans la fonction `update_neuron()`.
6. Vérifiez que vous pouvez bien obtenir l'émergence d'une unique bulle d'activité dans le DNF lorsque l'entrée fournie contient deux bulles d'activité gaussiennes. Vérifiez que le comportement ne change pas lorsque vous augmentez le nombre de neurones.

   Remarque : si votre DNF ne fait apparaître aucune activité significative, vous pouvez éventuellement inclure dans le code l'augmentation progressive du paramètre γ qui renforce l'influence des stimuli.

2 Spiking DNF

En vous aidant de votre programme préparé auparavant, et du premier TD, programmez une version impulsionnelle du DNF précédent, en utilisant des neurones CUBA-LIF.

Vérifiez que les propriétés d'émergence d'une bulle d'activité sont maintenues.

Testez le comportement de votre DNF impulsionnel lorsque les bulles d'activité en entrée se déplace. Si les impulsions restent localisées au même endroit ou si elles disparaissent ou s'étendent sur une zone de plus en plus grande, modifiez les paramètres pour que le DNF puisse suivre une bulle.