Sorbonne Univérsité - ISUP

Réseaux Symétriques de Hopfield en SageMath

Adnane EL KASMI

21 Novembre 2021

1 Préliminaire

[19]:
```
# La matrice des poids synaptiques W (transition):

def  W(vecteur):
    if  len (vecteur)  ==  1 :
        print  ( "vecteur invalide" )
        return
    else :
        w = [[0 for i in range(len(vecteur))] for i in range(len(vecteur))]
        for i in  range (len(vecteur)):
            for j in  range (i,len(vecteur)):
                if i == j:
                    w[i][j] = 0
                else:
                    w[i][j] = vecteur[i]*vecteur[j]
                    w[j][i] = w[i][j]
        return w
```

[20]:
```
# Fonction pour avoir une convertion binaire

def binaire(i):
    q = -1
    resultat = []
    while q != 0:
        q = i // 2
        r = i % 2
        resultat.append(r)
        i = q
    return resultat
```

[21]:
```
# Fonction pour creer tous les états initiaux

def etats_initiaux(taille_W):
    tous_etats_initiaux = {}
    for i in range(2**taille_W):
        b = binaire(i)
```

```
        a = [0 for j in range(taille_W - len(b)) ]
        tous_etats_initiaux["b""{}".format(i+1)] = (binaire(i) + a)
    return tous_etats_initiaux
```

2 Le cas Synchrone

[22]:
```
# La fonction activation pondéré du neurone

def e(W,a):
    W_initale = [0 for i in range(len(W))]
    for i in range(len(W)):
        W_initale[i] += sum( x*y for x,y in zip(W[i],a))

# ici x et y sont les élements de W[i] et a respectiviment pour tous i.

    return  W_initale
```

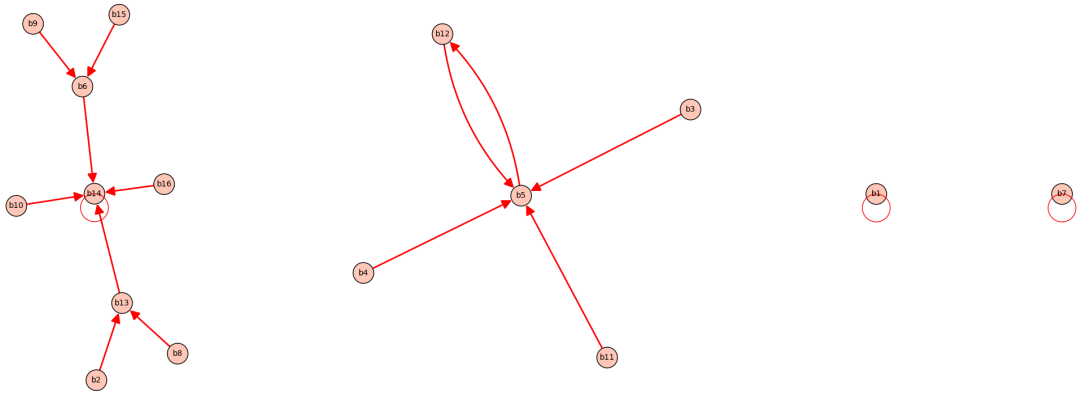
[23]:
```
#la fonction d'activation

def a(activation_pondere):
    fonction_activation = []
    for i in range(len(activation_pondere)):
        if activation_pondere[i] > 0:
            fonction_activation.append(1)
        else:
            fonction_activation.append(0)
    return fonction_activation
```

[24]:
```
# Fonction permettre de prendre un etat initial et regarder ou cela va etre␣
↪(pour tous les etats initiaux)

def synchrone_stationaire(W,tous_etats_initiaux):
    point_fixe = {}
    for cle, valeur in tous_etats_initiaux.items():
        a1 = a(e(W,valeur))
        for cle_1, valeur_1 in tous_etats_initiaux.items():
            if valeur_1 == a1:
                point_fixe.update({ cle:[cle_1]})
    return  point_fixe
```

2.1 Représentation graphique

[25]:
```
# Exemple de matrice W

W = [[0,-1,1,1],[-1,0,1,-1],[1,1,0,1],[1,-1,1,0]]
```

```
# Initialiser le graphe

Graphe = DiGraph(synchrone_stationaire(W,etats_initiaux(len(W))))

# Afficher le graphe

Graphe.plot(color_by_label = True).show(figsize = (20,20))
```



3 Le cas Asynchrone

[26]:
```
# Fonction qui permet de réaliser un produit de deux vecteurs

def produit_vecteur(vecteur1,vecteur2):
    return sum(i*j for i,j in zip(vecteur1,vecteur2))

# Fonction qui permet de réaliser un produit d'une matrice et d'un vecteur

def produit_matrice_vecteur(matrice,vecteur):
    resultat_vecteur = []
    for i in range(len(matrice)):
        resultat_vecteur.append(produit_vecteur(matrice[i],vecteur))
    return resultat_vecteur

# Fonction pour copier

def copier(liste):
    copie = []
    for element in liste:
        copie.append(element)
    return copie
```

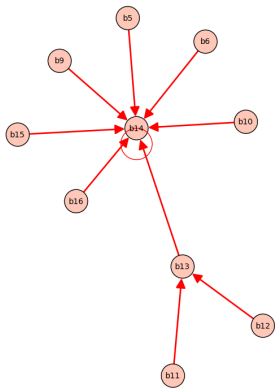
[27]:
```
# La fonction energie

def energie(W,a):
    return -produit_vecteur(a,produit_matrice_vecteur(W,a))

# fonction pour l'activation d'un seul élement

def activation_un_element(x):
    return 1 if x>0 else 0
```

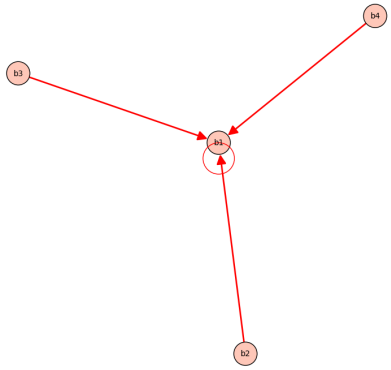
[28]:
```
# Fonction couche suivante

def couche_suivante(W,a_0):
    copie_a = copier(a_0)
    for k in range(len(W)):
        resultat = []
        for i, j in zip(W[k], copie_a):
            resultat.append(i*j)
        copie_a[k] = activation_un_element(sum(resultat))
    return copie_a
```

[29]:
```
# fonction synchrone stationaire

def asynchrone_stationaire(W,tous_etats_initiaux):
    point_fixe = {}
    for cle, valeur in tous_etats_initiaux.items():
        a1 = couche_suivante(W, valeur)
        for cle_1, valeur_1 in tous_etats_initiaux.items():
            if valeur_1 == a1 :
                point_fixe.update({ cle:[cle_1]})
    return  point_fixe
```

3.1 Representation graphique

[31]:
```
# Exemple de matrice W

W = [[0, -1, 1, 1], [-1, 0, 1, -1], [1, 1, 0, 1], [1, -1, 1, 0]]

# Initialiser le graphe

Graphe = DiGraph(asynchrone_stationaire(W,etats_initiaux(len(W))))

# Afficher le graphe

Graphe.plot(color_by_label = True).show(figsize = (20,20))
```


b5
b6
b9
b14
b10
b15
b16
b13
b12
b11


b4
b3
b1
b2


b8
b7