TP : RN-AlgoGN

fonctions Matlab :

M=imread('Fry.bmp') ;↦ lit l'image 'Fry.bmp' dans dossier courant de MATLAB et l'enregistre dans la matrice M (valeur logique : 0 ou 1)

imshow(M) ;↦ affiche l'image associé à la matrice de valeurs logiques M
imwrite(M,'test.bmp') ;↦ enregistre l'image 'test.bmp' (associée à la matrice M) dans dossier courant de MATLAB

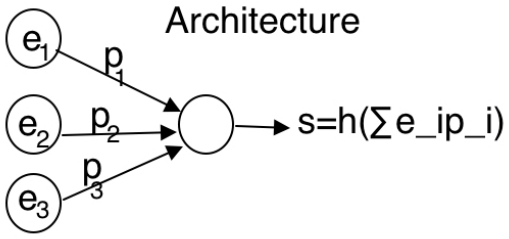
Q=double(M) ;↦ transforme la matrice M de valeur logique en matrice Q de réels

M=logical(Q> $seuil$) ;↦ $M(i,j) = 1$ si $Q(i,j) > seuil$ et 0 sinon (M est une matrice de valeur logique)
fonctions Python :Voir Module skimage

Exercice 1 : On se donne l'image 'Fry.bmp'. On souhaite faire un réseau de neurones qui agissent sur les neurones d'entrées associés à l'image 'Fry.bmp'. Un neurone est un pixel de l'image qui est à 0 ou à 1.

Pour chaque pixel i, on renvoie $e_i = 1$ si $\sum_{j\ voisin\ du\ pixel} a_{ji} * e_j > seuil$, 0 sinon.
1) Cas 1 : les $a_{voisin} = 1$ faire varier le seuil, observer le résultat et expliquer le.
2) Cas 2 : les $a_{ji,\ voisin\ j\neq i} = -1$, $a_{ii} = \alpha$, faire varier α et la valeur du seuil, observer le résultat et expliquer le.

Exercice 2 : Implanter le réseau associé à l'architecture (ci dessous, à droite)



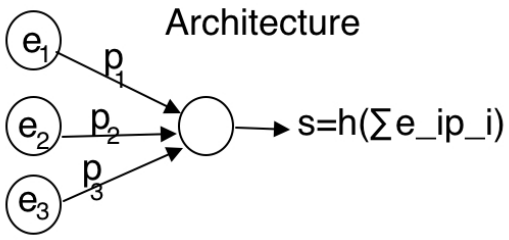
Ce que l'on veut

Entrées			Sortie
e_1^{th}	e_2^{th}	e_3^{th}	s^{th}
0	0	1	0
1	1	1	1
1	0	1	1
0	1	1	0

Les valeurs des poids sont entre -20 et 20 et la fonction d'activation h sera Heaviside, Linéaire $x \mapsto min(1, max(0, Tx))$ avec $T > 0$, Sigmoïde $x \mapsto 1/(e^{-Tx} + 1)$ (avec $T = 1$). L'erreur du réseau sera la norme 2 : $err(P) = norm2(h(\sum p_i e_i^{th}) - s^{th})$.
1) Créer la fonction d'erreur qui a une matrice de poids $Q = \begin{pmatrix} q_1 \\ q_2 \\ q_3 \end{pmatrix}$ et une fonction d'activation h renvoie l'erreur $err(Q)$.
2) Créer une population de poids, c'est-à-dire une matrice $P = (Q_1, Q_2, \cdots, Q_{taille\ pop.})$ et un vecteur d'erreur $err = (err_1, \cdots err_{taille\ pop.})$ avec $taille\ pop.$ entre 10 et 20.
3) Créer un algorithme génétique dont l'objectif est de minimiser l'erreur du réseau. Tester avec différentes fonctions d'activation et valeur de T. [1]

1. Attention, l'algo. géné doit contenir les mutation d'un seul poids synaptique

Exercice 2bis : Implanter le réseau associé à l'architecture (ci dessous, à droite)

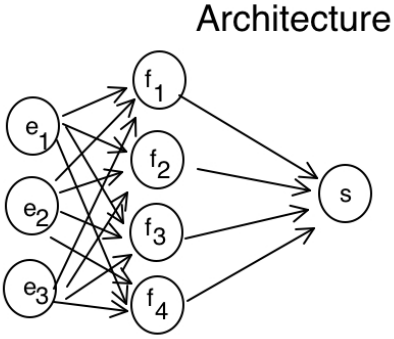


Ce que l'on veut

Entrées			Sortie
e_1^{th}	e_2^{th}	e_3^{th}	s^{th}
0	0	1	0
1	1	1	0
1	0	1	1
0	1	1	1

Essayer de faire de même. Qu'observe t'on ?

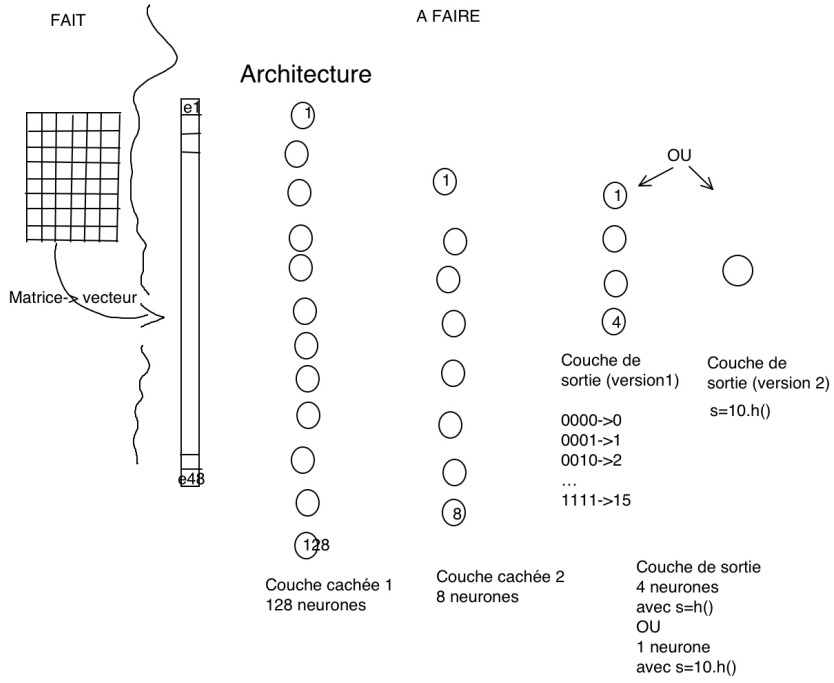
On ajoute une couche cachée de quatre neurones, chercher des valeurs de poids P_1 entre la couche d'entrée et la couche cachée et P_2 entre la couche cachée et la couche de sortie. Optimiser les poids de manière à minimiser l'erreur du réseau.



Ce que l'on veut

Entrées			Sortie
e_1^{th}	e_2^{th}	e_3^{th}	s^{th}
0	0	1	0
1	1	1	0
1	0	1	1
0	1	1	1

Exercice 3 : Le fichier train-only-input-48x218.data contient 218 vecteurs de taille 48 représentant des chiffres de 0 à 9 dont la représentation est donnée dans train-only-output-1x218.data



Optimiser le réseau pour minimiser l'erreur et tester sur test.data (contenant quelques exemples entrée sortie).

Exercice 4 : On se donne l'image 'Alien.jpg' ayant $144 * 144$ pixels (voir sur le site du cours dans pédagogie). On souhaite faire un réseau de neurones qui agissent sur les neurones d'entrées associés à l'image 'Alien.jpg'. Un neurone est un pixel de l'image qui est entre 0 et 255 (niveau de gris).

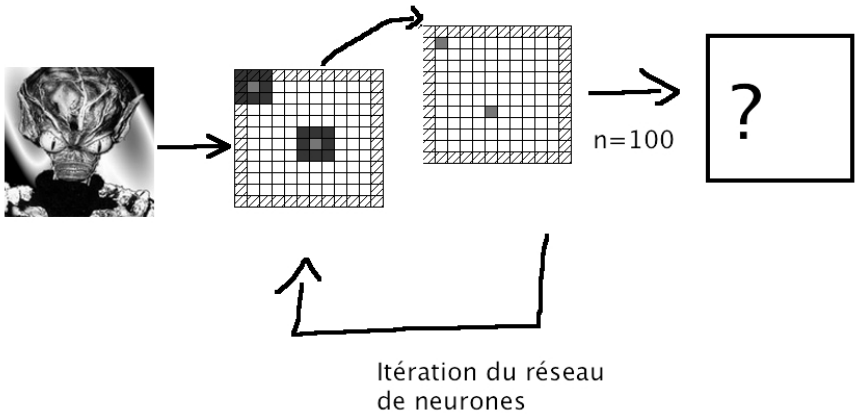
Pour chaque pixel (i, j) qui ne se trouve pas sur le bord, c'est à dire, $i \neq 1; 144$ et $j \neq 1; 144$, on applique un réseau de neurones qui transforme la matrice $M_1 = double(imread('Alien.jpg'))$ en une matrice M_2 de taille $144 * 144$ (la même que M_1) :

$$M_2(i,j) = s(\sum_{i'=i-1}^{i+1} \sum_{j'=j-1}^{j+1} P_{i,j} M_1(i,j)),$$

$$M_2(1,j) = M_1(1,j), \;\; M_2(144,j) = M_1(144,j), \;\; M_2(i,1) = M_1(i,1), \;\; M_2(i,144) = M_1(i,144),$$

avec $s(z) = \begin{cases} 0, & z \leq 0, \\ 255, & z \geq 255, \\ \text{Partie entière}(z), & z \in [0, 255]. \end{cases}$ *et* $P = \begin{bmatrix} -1.6 & 0.4 & -1.5 \\ 0.1 & 6.7 & 0.1 \\ -1.5 & 0.4 & -1.6 \end{bmatrix}$

On applique à nouveau le réseau de neurones sur la matrice M_2 pour créer une matrice M_3. On itère le procédé en ré-injectant la matrice M_n dans le réseau pour créer une matrice M_{n+1}.



Pour n assez grand (cela devrait s'observer assez vite $n \leq 100$), que dire de $M_{n+1} - M_n$?
En déduire que le réseau admet une matrice invariante.

Exercice 5 : Amélioration de l'exo 1-

1) Construire un réseau de neurones capable de détecter les contours d'une image en dégradé de gris (et en couleur)
2) Construire un réseau de neurones faisant une moyenne locale (blur gaussien) d'une image en dégradé de gris