# TP 1 NMI

Ben Amira Rawia

27 février 2022

**Exercice 1.** *Rappeler le lien entre l'information mutuelle normalisée I(X,Y) de deux variables aléatoires X et Y et l'entropie de celles-ci.*

L'information mutuelle de deux variables aléatoires est une quantité qui mesure la dépendance entre ces deux variables . L'information mutuelle normalisée permet de comparer des clusterings ayant des nombres de clusters différents. L'entropie mesure la quantité moyenne d'information d'un ensemble d'évènements,et son incertitude.

la relation entre entropie et information mutuelle normalisée est :

$$NMI = \frac{2 * MI(C,Y)}{H(C) + H(Y)}$$

avec MI : information mutuelle H(C) : entropie de la premiere variable C H(Y) : entropie de la deuxieme variable Y

**Exercice 2.** *Considérons l'image "lena.jpeg" comme une variable aléatoire, calculer et afficher la densité de probabilité associée*

```
import numpy as np
from PIL import Image
import matplotlib.pyplot as plt
##########################################EXERCICE 1 #################
Im = Image.open('lena.jpeg')
Imarray = np.asarray(Im)
#numpy.asarray() function is used when we want to convert input to an array.

plt.figure()
#Create a new figure, or activate an existing figure.
plt.imshow(Imarray, cmap='gray')
#The imshow() is used to display data as an image

# Calculer et afficher la densité de probabilité de l'image
xe = np.asarray(range(np.amaImarray[:])+2))
H1, xe = np.histogram(Imarray.reshape(-1), bins=xe)
#Compute the histogram of a dataset.
P1 = H1/Imarray.size
print(H1)
#P1 est la liste de probabilité des couleurs
#le truc a afficher c est P1
plt.figure()
plt.plot(P1)
```

la densite de probabilité est decrite par une matrice geante ( P1) de 253 elements( ci desoous )

```
[1.22070312e-04 3.05175781e-05 6.10351562e-05 7.62939453e-05
 1.83105469e-04 2.59399414e-04 2.13623047e-04 4.11987305e-04
 4.57763672e-04 5.18798828e-04 8.39233398e-04 7.47680664e-04
 1.09863281e-03 1.29699707e-03 1.49536133e-03 1.55639648e-03
 1.78527832e-03 2.24304199e-03 2.27355957e-03 3.08227539e-03
 3.03649902e-03 3.75366211e-03 4.53186035e-03 5.05065918e-03
 5.24902344e-03 5.38635254e-03 6.40869141e-03 6.07299805e-03
 6.89697266e-03 6.39343262e-03 6.71386719e-03 7.23266602e-03
 6.57653809e-03 6.80541992e-03 5.87463379e-03 6.59179688e-03
 6.30187988e-03 5.52368164e-03 5.05065918e-03 4.91333008e-03
 4.42504883e-03 3.86047363e-03 3.64685059e-03 3.69262695e-03
 3.58581543e-03 3.32641602e-03 3.08227539e-03 3.09753418e-03
 2.80761719e-03 2.79235840e-03 2.30407715e-03 2.63977051e-03
 2.48718262e-03 2.51770020e-03 2.42614746e-03 2.68554688e-03
 2.59399414e-03 2.57873535e-03 2.77709961e-03 2.88391113e-03
 2.88391113e-03 2.96020508e-03 2.62451172e-03 3.35693359e-03
 2.77709961e-03 3.09753418e-03 3.14331055e-03 2.80761719e-03
 3.35693359e-03 3.11279297e-03 2.79235840e-03 3.15856934e-03
```

**Exercice 3.** *Calculer et afficher l'entropie de l'image*

```python
import math as mth
# Calculer et afficher l'entropie de l'image
def entropie(liste_pro):
 """
 Calcule et affiche l entropie d une image à partir de la liste contenant
 les differentes probabilitées
 rappelons que la formule c est \uD835\uDC3B(\uD835\uDC3C) = - somme(\uD835\uDC5
 """
 entropi = 0
 for el in liste_pro :
 if el != 0 :
 entropi = entropi + el * mth.log2(el)
 return -entropi

entrop = entropie(Pl)
print(entrop)
#l entropie est egale a 7.653048611238887 pour cette image
```

l entropie est egale a 7.653048611238887

**Exercice 4.** *Utiliser l'image seuillée lena4 et calculer la densité de probabilité jointe entre lena et lena4 ainsi que l'entropie jointe.*

```
from itertools import combinations

Im4 = Image.open('lena4.jpeg')
Imarray4 = np.asarray(Im4)

plt.figure()
#Create a new figure, or activate an existing figure.
plt.imshow(Imarray4, cmap='gray')
#The imshow() is used to display data as an image

# Calculer et afficher la densité de probabilité de l'image
y = np.asarray(range(np.amaImarray4[:])+2))
#Compute the histogram of a dataset.
H , xe , y = np.histogram2d(Imarray.reshape(-1),Imarray4.reshape(-1),bins=(256,2
#Pl est la liste de probabilité des couleurs
#le truc a afficher c est Pl
plt.figure()
plt.plot(Pl)
# Densité de probabilité jointe
print(H)
P = H/Imarray.size
P = P.reshape(-1)
# Entropie jointe
entJointe = entropie(P)
print(entJointe)
```

l'entropie jointe est egale à 10.362911876757801, il suffit de faire appel a la fonction qui calcule l entropie mais avec des valeurs de probabilté jointe qu on extrait grace a la fonction histogram2d

**Exercice 5.** *Ecrire une fonction NMI(I1, I2) qui renvoie l'information mutuelle normalisée entre les images Im1 et Im2*

```
def NMI(Im1, Im2):
 # Im1: image 1 (2D np array)
 Imarray1 = np.asarray(Im1)
 # Im2: image 2 (2D np array)
 Imarray2 = np.asarray(Im2)

 # Densité de probabilité et entropie de chaque image
 x1 = np.asarray(range(np.amax(Imarray1[:])+2))
 H1, x1 = np.histogram(Imarray1.reshape(-1), bins=x1)

 x2 = np.asarray(range(np.amax(Imarray2[:])+2))
 H2, x2 = np.histogram(Imarray2.reshape(-1), bins=x2)

 P1 = H1/Imarray1.size
 P2 = H2/Imarray2.size

 entr1 = entropie(P1)
 entr2 = entropie(P2)

 # Densité de probabilité jointe et entropie jointe
 H , x1 , x2 = np.histogram2d(Imarray1.reshape(-1),Imarray2.reshape(-1),bins=(256,256))
 P = H/Imarray1.size # Imarray1 ou Imarray2 dans notre cas c'est 256
 P = P.reshape(-1)
 # Entropie jointe
 entJointe = entropie(P)
 # Calcul de l'information mutuelle normalisée
 mi = entr1 + entr2 - entJointe
 nmi = 2*mi /(entr1 + entr2)

 return nmi
```

comme on vient de le rappeler dans le premier exercice , le calcule de l information mutuelle normalisé se fait :

$$NMI = \frac{2 * MI(C, Y)}{H(C) + H(Y)}$$

```
# NMI entre les deux images
Im1 = Image.open('lena.jpeg')
Im2 = Image.open('lena4.jpeg')

nmi = NMI(Im1,Im2)
print(nmi)
```

```
0.29239682709028103
```

**Exercice 6.** *On a vu en cours que l'information mutuelle normalisée est maximale lorsque celles-ci sont recalées "au mieux". Ecrire une fonction registration(I1, I2, niter) qui calcule itérativement la translation optimale permettant de registrer les deux images.*

```
[29] def registration(Im1cent, Im2array, niter=100, crop=200, depv=0, deph=0):
       # Im1cent: partie centrale de l'image 1 (2D np array)
       # Im2array: image 2 (2D np array)
       # niter: nombre d'itérations (default: 100)
       # crop: marge pour le déplacement (default: 200)
       # depv: déplacement vertical initial (default: 0)
       # deph: déplacement horizontal initial (default: 0)
       vec_nmi = []
       # Itérer niter fois
       for i in range(niter):
           print(f'Iteration: {i}')
           print(f'Deplacement vertical: {depv}')
           print(f'Deplacement horizontal: {deph}')

           # Extraire la partie de l'image 2 centrée sur le déplacement actuel et calculer la nmi
           Im2cent = Im2array[crop:-crop,crop:-crop]
           nmi0 = NMI(Im1cent,Im2orig)

           # Extraire chaque partie de l'image 2 décalée d'un pixel dans chacune des 4 directions
           Im1 = Im2cent[crop+depv+1:-crop+depv+1,crop+deph:-crop+deph]
           Im2 = Im2cent[crop+depv:-crop+depv,crop+deph+1:-crop+deph+1]
           Im3 = Im2cent[crop+depv-1:-crop+depv-1,crop+deph:-crop+deph]
           Im4 = Im2cent[crop+depv:-crop+depv,crop+deph-1:-crop+deph-1]

           # Calculer l'information mutuelle normalisée entre l'image 1 et chacune des images décalées
           nmi1 = NMI(Im1cent,Im1)
           nmi2 = NMI(Im1cent,Im2)
           nmi3 = NMI(Im1cent,Im3)
           nmi4 = NMI(Im1cent,Im4)

           # Décider de la direction dans laquelle déplacer l'image et incrémenter depv et deph d'un pixel en fonction
           nmi = max(Im1,Im2,Im3,Im4)
           vec_nmi.append(nmi)
           if nmi == Im1 :
             deph = depv + 1
           elif nmi == Im2 :
             deph = deph + 1
           elif nmi == Im3 :
             depv = depv - 1
           else :
             deph = deph - 1
       return Im2cent, vec_nmi, depv, deph
```

l idee de cet algorithme est de partir d une certaine position et de faire des deplacements a chaque fois jusqu'a alignement parfait , arrivée a l alignement parfait l information mutuelle normalisée est maximale donc a chaque essais de depacment horizontale ou verticale , on calcule la NMI entre l image de depart et la nouvelle image deplacé , on compare les 4 NMI calculé pour les 4 sens ( haut , bas , droit gauche ) , la plus grande valeur de NMI sera choisie parmis les 4 , et l image sera deplacé dans le sens qui a la plus grande NMI