I61_DAUSSY_2023

DAUSSY Fabio

April 2023

1 TP I61

Dans la première partie de ce TP nous allons dans un premier temps étudier un algorithme qui va nous permettre de calculer l'entropie d'une séquence, soit une entropie conjointe de n éléments où n représente la longueur de la séquence, en utilisante la règle de la chaine.

Pour cela, nous allons réalisé les étapes suivantes :

- Construire l'**hypercube** de la matrice d'entrée
- Écrire un algorithme qui va **projeter** cet hypercube dans des dimensions inférieures (pour calculer des probabilités jointes avec des variables aléatoire en moins).
- **Tester** le code pour voir si tout cela fonctionne.

Voici le code qui permet de construire l'hypercube :

```
def hypercube(OBS):
    """ Construit l'hypercube de la matrice """

    r = len(OBS) # On recupere le nombre de realisation

    dico = {} # On initialise le dictionnaire, dont les cles sont les sequences
    # Ce dictionnaire va compter les occurences des sequences.

    # Pour chaque sequence, on ajoute sa concatenation en cle du dico si
    # elle n'y est pas, sinon on ajoute 1 a la cle existente.
    for i in range(r):

        if tuple(OBS[i]) not in dico.keys(): # On transtype en tuple car pas de liste
        # en cle de dictionnaire
            dico[tuple(OBS[i])] = 1
        else:
            dico[tuple(OBS[i])] += 1
```

```
    # On calcule ensuite les probabilités jointes de nos sequences et on les stocke dans
    # le dictionnaire
    for k in dico.keys():
        dico[k] = dico[k] / r
    return dico
```

Grâce à l'hypercube, il est très facile de calculer l'entropie jointe de notre séquence et ses réalisations :

```
    def entropie_jointe(OBS):
    """
    Calcule l'entropie jointe de la séquence et des ses réalisations, stockée dans
    la matrice OBS
    """
    somme = 0
    hcube = hypercube(OBS)

    for key in hcube.keys():

        somme += hcube[key] * log2(hcube[key])

    return - somme
```

Ensuite, il faut implanter le code qui permet de réduire l'hypercube pour pouvoir calculer les entropies conditionnelles :

```
    def reduire(hypercube):
    """
    Cette fonction va permettre de réduire l'hypercube de une dimension
    """

    nouveau = {} # Le nouveau dictionnaire qui va stocker les probabilites jointes
    # de hypercube avec une dimension de moins


    for key in hypercube.keys():
        prob = hypercube[key] # On memorise la probabilite courante
        # On regarde si la nouvelle cle existe dans le nouveau dico
        if key[:-1] not in nouveau.keys():
            # si elle n'y est pas on l'ajoute
            nouveau[key[:-1]] = prob # avec la probabilite memorisee !

        else:
            # si elle y est on fait la somme des probabilites
            nouveau[key[:-1]] += prob

    return nouveau
```

Par manque de temps, je n'ai pas pu implanter la fonction qui permet de calculer l'entropie dirigée:

$$IM(X^n \to Y^n) = \sum_{i=1}^{n} IM(X^i; Y_i | Y^{i-1}) \tag{1}$$

Mais qui prochainement fait grâce à la règle de la chaîne et grâce à la propriété suivante :

$$IM(X; Y|Z) = H(X|Z) + H(Y|Z) - H(X, Y|Z) \tag{2}$$

qui peut se généraliser aux séquences.