TD/TP HMM B

Romain Hérault

Automne 2015

1 Apprentissage des HMM

Dans le fichier `TD4b-X1.npz`, vous trouverez un ensemble de `sequences` issu d'un même processus modelable par un HMM à `nStates` états et à `nObs` observations. En vous aidant de la bibliothèque `yahmm`, retrouver les paramètres de ce HMM par apprentissage de Baum-Welch.

2 HMM à observations continues

Dans le fichier `TD4b-X2.npz`, vous trouverez une séquence `y` à étiqueter.

Les deux états possibles 0 et 1 ont respectivement une espérance de durée de 50 et 25. A l'état 0, les observations suivent une loi normale de centre -1 et d'écart-type 1. A l'état 1, les observations suivent une loi normale de centre 1 et d'écart-type 2.

Retrouver la séquence d'états la plus vraisemblable.

A Annexes

Pour vous aider à créer un modèle `yahmm` depuis ou vers une représentation matricielle.

```
1  import yahmhelper as yhh
2
3  def genData():
4      p0=np.array([0.6,0.4])
5      A=np.array([[0.8,0.2],[0.2,0.8]])
6      B=np.array([[0.9,0.1],[0.1,0.9]])
7
8
9      genmodel=yhh.modelFromLambda(p0,A,B)
10     genmodel.bake()
11
12     nS=100;
13     nT=50;
14     sequences=[]
15     for s in range(nS):
16         sequences.append(genmodel.sample(nT))
17
18     nStates=A.shape[0]
19     nObs=B.shape[1]
20
21     return (nStates,nObs,sequences)
22
```

```
23 def train(nStates,nObs,sequences):
24
25     s=0.25
26
27     p0=0.5+s*np.random.rand(nStates)
28     p0/=p0.sum()
29
30     A=0.5+s*np.random.rand(nStates,nStates)
31     A/=A.sum(axis=1,keepdims=True)
32
33     B=0.5+s*np.random.randn(nStates,nObs)
34     B/=B.sum(axis=1,keepdims=True)
35
36     trainmodel=yhh.modelFromLambda(p0,A,B)
37     trainmodel.bake()
38
39
40     trainmodel.train(sequences,min_iterations=10)
41
42     p0est,Aest,Best=yhh.modelToLambda(trainmodel)
43
44     return p0est,Aest,Best
```