TP3

June 8, 2018

1 TP3 - Automatic Segmentation of Mails

- Davide GALLITELLI
- Carlotta CASTELLUCCIO

This Lab aims to build an email segmentation tool, dedicated to separate the email header from its body. It is proposed to perform this task by learning a HMM (A, B, π) with two states, one (*state 1*) for the header, the other (*state 2*) for the body. In this model, it is assumed that each mail actually contains a header : the decoding necessarily begins in the state 1.

Knowing that each mail contains exactly one header and one body, each mail follows once the transition from 1 to 2.

1.0.1 Q1 : Give the value of the vector of the initial probabilities

$$\pi^T = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

Because the initial probability of being in the state 0 (header) is always true.

1.0.2 Q2 : What is the probability to move from state 1 to state 2 ? What is the probability to remain in state 2 ? What is the lower/higher probability ? Try to explain why

Given the matrix:

$$A = \begin{bmatrix} 0.999218078035812 & 0.000781921964187974 \\ 0 & 1 \end{bmatrix}$$

The probability of moving from state 1 to state 2 is $P(2|1) = 0.000781921964187974$; the probability of state 2 is $P(2|2) = 1$.

It is normal for $P(2|2)$ to be higher, since once a character belonging to the *body* of the mail has been found, all the following observations will belong to the same state : no other *header* is found after the *body* in a mail.

1.0.3 Q3 : What is the size of the corresponding matrix ?

Because the ASCII characters are 256 ($N = 256$), the size of the corresponding matrix will be $256x2$, where each row represents the discrete probability distribution of the character c given the state s.

In [1]:
```
import numpy as np

# Viterbi - test
O = range(0,256)
S = [0,1]
pi = [1,0]
A = [[0.999218078035812, 0.00078192196418797 4],[0, 1]]
B = np.loadtxt('P.text', dtype=float)
```

In [2]:
```
# Implementation of Viterbi algorithm
"""
Implementation of the Viterbi algorithm.
Finds the best segmentation of the text provided,
and returns the most probable states sequence
Parameters explaination:
:param O: = characters codification (ASCII, range(0,256))
:param S: = possible states (header = 0, body = 1)
:param A: = state transition probability matrix
:param B: = matrix probability of character c being in state s (256x2, contained in P.
:param pi: = initial probability vector
:param Y: = observation in the mail (mail.dat)
:return X: = most likely hidden state sequence
"""
def ViterbiAlgorithm(O, S, A, B, pi, Y):
    A = np.matrix(A)
    T = len(Y)
    T1 = np.zeros((len(S), T))
    T2 = np.zeros((len(S), T))
    for s in S:
        T1[s,0] = pi[s]*B[Y[0]][s] # vector of most likely path so far
        T2[s,0] = 0 # most likely path for previous observation
    for t in range(1,T):
        for s in S:
            result = [a*b*B[Y[t],s] for a,b in zip(T1[:, t-1],A[:,s])]
            result += np.finfo(np.double).tiny
            T1[s, t] = np.max(result)
            T2[s, t] = np.argmax(result)
#         Rescale output to solve too small number errors
        T1[:,t] = T1[:,t]*(10**(-1*(int(np.log10(np.max(T1[:,t])))+1)))
    Z = [0 for t in Y]
    X = [0 for t in Y]
    T = T - 1
    Z[T] = int(np.argmax(T1[:,T]))
    X[T] = S[Z[T]]
```

```
        for i in range(T, 0, -1):
            Z[i-1] = int(T2[Z[i], i])
            X[i-1] = S[Z[i-1]]
        return X
```

1.0.4 Q4: Print the track and present and discuss the results obtained on mail11.txt to mail30.txt

```
In [3]: Y = np.loadtxt('dat/mail11.dat',dtype=int)

        X = ViterbiAlgorithm(O, S, A, B, pi, Y)
        print("Observation size: "+str(len(Y)))
        print("Expected change point in mail11.dat : ~2850")
        print("Obtained change point : "+str(np.bincount(X)[0]))
```

```
Observation size: 3475
Expected change point in mail11.dat : ~2850
Obtained change point : 2851
```

```
In [4]: Y = np.loadtxt('dat/mail30.dat',dtype=int)

        X = ViterbiAlgorithm(O, S, A, B, pi, Y)
        print("Observation size: "+str(len(Y)))
        print("Expected change point in mail30.dat : ~2250")
        print("Obtained change point : "+str(np.bincount(X)[0]))
```

```
Observation size: 5160
Expected change point in mail30.dat : ~2250
Obtained change point : 2173
```

The results seem consistent with the expected values. The expected values have been obtained by opening with a text editor the mail*.txt files and counting the characters before the body.

For *mail11.txt*, the delta in recognising the body is just one character from the expected value, which is great. For *mail30.txt*, the delta is a little bit higher, but nevertheless acceptable.

1.0.5 Q5 : How would you model the problem if you had to segment the mails in more than two parts (for example : header, body, signature) ?

The hidden model would not change: the changes to be done would be on the inputs. In particular, the states space would now have size of 3 (header = 0, body = 1, signature = 2), and therefore the vector of initial probabilities π would have 3 elements

$$\pi = (1, 0, 0)$$

the transition matrix `A` would have size `3x3`, with form:

$$A = \begin{bmatrix} p_{11} & p_{12} & 0 \\ 0 & p_{22} & p_{23} \\ 0 & 0 & 1 \end{bmatrix}$$

Even `B`, conditional probability of observation given the state, would change, becoming a matrix with size `256x3`.

1.0.6 Q6 : How would you model the problem of separating the portions of mail included, knowing that they always start with the character ">".

We would then need a 4th state, *mail_included*, which comes before the *signature* of the mail. The inputs would change, including this 4th state: for example the transition matrix `A` would now be:

$$A = \begin{bmatrix} p_{11} & p_{12} & p_{13} & 0 \\ 0 & p_{22} & p_{23} & p_{24} \\ 0 & p_{32} & p_{33} & p_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

with the initial probability vector

$$\pi = (1, 0, 0, 0)$$

Moreover, knowing that the mail included always starts with character ">", the probability of the character belonging to state 3 is higher than in the other states - this information should be included in the conditional probability matrix `B`.