JANUARY 1, 2016

Assignment 2

PARAMETER ESTIMATION AND DECODING IN HMMS.

CHEN, CHEN

IMPERIAL COLLEGE LONDON

EM for HMM

*In the initialization phase, I randomly generated emission probability **B**, transition matrix **A**, and $\boldsymbol{\pi}$ the hidden state probability at first time frame. And then I used them to do EM algorithm, finding the parameters $\boldsymbol{A}$, $\boldsymbol{\pi}$, **B** by maximizing the probability $p(\boldsymbol{x}_{1..T}, \boldsymbol{z}_{1..T}|\theta)$.*

As EM algorithm doesn't guarantee to reach the global optimiser, I run the above EM for multiple times and chose the one who has maximum log likehood over evaluation as the model. Each time, the EM algorithm will iteratively bring the function to its local minimum and convergence at some point (see figure 1learning curve).

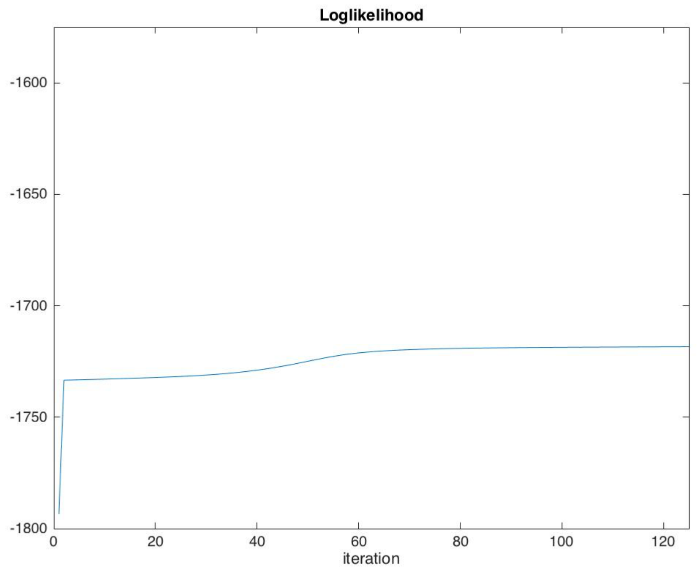

figure 1

Note that the two cases: Discrete Observation Models and Continuous Observation Models use the same EM algorithm and the only difference is the emission probability B. In discrete model, B is L by K matrix where value in each row sum up to 1. In continuous case, I firstly assign random values for K' distribution of mean and variance.

(In the experiment of discrete case, K is the hidden state numbers: 2, L is number of variables for a single state: 6 while in continuous case, I generated K Gaussian distributions for initialization.)

1.1 Results

N = 10; % number of sequences

T = 100; % length of the sequence

- ***Discrete Model***

Case 1:

pi = [0.4; 0.6];

A = [0.4 0.6 ; 0.4 0.6];

B = [1/6 1/6 1/6 1/6 1/6 1/6;

1/10 1/10 1/10 1/10 1/10 1/2];

good result 1 :

pi: [0.4885;0.5115]

A:[0.4875 0.5125; 0.4890 0.5110]

B:[0.1513 0.1240 0.1397 0.1621 0.0425 0.3805;

0.1057 0.1338 0.1384 0.0974 0.2292 0.2955]

bad result

We could sometimes get the result which is not so good:

For example:

Pi: 0.7001; 0.2999

A: [0.6071 0.3929; 0.9173 0.0827]

B: [0.1349 0.1167 0.1012 0.1314 0.1282 0.3876;

0.1119 0.1576 0.2272 0.1235 0.1607 0.2190]

case 2:

pi = [0.5; 0.5]

A = [0.9 0.1 ; 0.1 0.9]

B = [1/6 1/6 1/6 1/6 1/6 1/6;

1/10 1/10 1/10 1/10 1/10 1/2];

good result :

pi: [0.5181 ; 0.4819]

A: [0.8566   0.1434; 0.1529   0.8471]

B:

[0.1549   0.1886   0.1858   0.1729   0.1775   0.1204;

0.1029   0.0870   0.0587   0.1229   0.0838   0.5446;]

- ***Continuous Model***

Input:

pi = [0.5; 0.5];

A   = [0.4 0.6 ; 0.4 0.6];

E.mu   =[.9 10]; %%the means of each of the Gaussians

E.sigma2=[.4 .2]; %%the variances

Output:

Pi: [0.5923 ;0.4077]

A: [0.4242   0.5758;

0.3994   0.6006]

mu: [0.9001   10.0005]

sigma2: [0.4186   0.2232]

Viterbi

In order to find the most likely sequence of hidden states of one observation sample, I used Viterbi algorithm, a dynamic programming method. By iteratively calculate each state probability at each time frame and applying backtracking, we could find the most likely decode sequence. (see Viterbi.m)

*In order to decrease the complexity, I used log2 to preprocess parameters of A, π, **B.** The performance of Viterbi is shown as follows. By comparing the decode path with the original hidden state sequence (generated by the HmmGenerateData function) with 10 samples, I found that the Viterbi algorithm's correctness fluctuates between 50~97% (figure 2)*

Results:

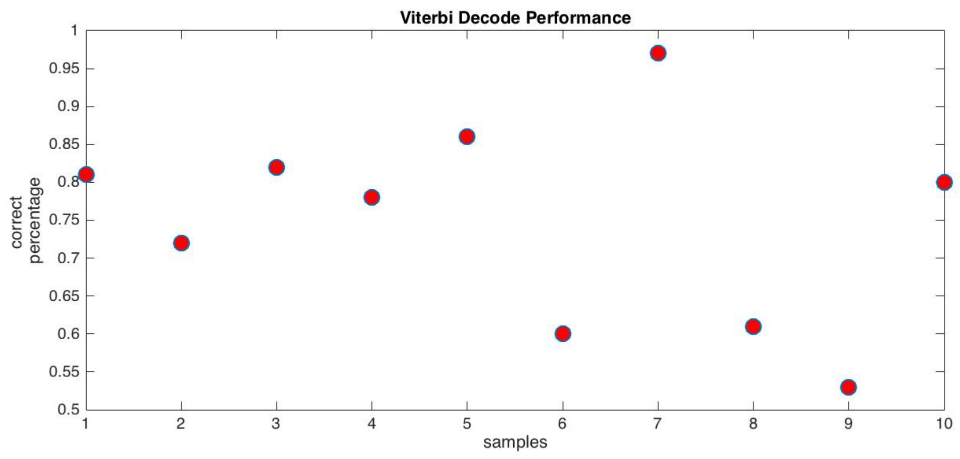

figure 2

Example input: (see run_viterbi.m)

```
N   = 10;          % number of sequences
T   = 100;         % length of the sequence
pi = [0.5; 0.5]; % inital probability pi_1 = 0.5 and pi_2 =0.5
A   = [0.1 0.9 ; 0.9 0.1 ];          %p(y_tly_{t-1})
%%alphabet of 6 letters (e.g., a die with 6 sides) B(i,j) is the
```

```
B = [1/6 1/6 1/6 1/6 1/6 1/6;          %p(x_t|y_{t})

     1/10 1/10 1/10 1/10 1/10 1/2];
```