EQ2341 Pattern Recognition and Machine Learning

Assignment 1 HMM

Jiaojiao Wu
Antoine Camus

KTH Royal Institute of Technology

15-04-2020

Contents

1. Description of the Assignment

This assignment includes two parts. The first is to finish the codes in the PattRecClasses code package, and the second part is to verify the code and do some calculations and analysis.

2. Main tasks

2.1 Task 1

The first step is to finish codes, copies of @DiscreteD/rand, @MarkovChain/rand, and@HMM/rand Matlab functions files which are attached in a zip archive.

2.2 Task 2

After finishing the code package, the next step is to verify the code. This can be done by the following steps:

2.2.1 Calculate $\mathbf{P(S_t = j)}$

For the following infinite-duration HMM = $\{\mathrm{q}, A, b\}$ with parameters as follows:

$$q = \begin{pmatrix} 0.75 \\ 0.25 \end{pmatrix}; \qquad A = \begin{pmatrix} 0.99 & 0.01 \\ 0.03 & 0.97 \end{pmatrix}; \qquad b = \begin{pmatrix} \mathrm{b}_1(\mathrm{x}) \\ \mathrm{b}_2(\mathrm{x}) \end{pmatrix}$$

Now let's calculate the possibility of S_t = j. The initial state (t=1) probability distribution is shown as q, which means

$$P(S_1 = 1) = 0.75$$
$$P(S_1 = 2) = 0.25$$

For t=2, the current S_2 depends on the S_1.

$$\begin{aligned} P(S_2 = 1) &= P(S_2 = 1|S_1 = 1) * P(S_1 = 1) + P(S_2 = 1|S_1 = 2) * P(S_1 = 2) \\ &= A(1,1) * 0.75 + A(2,1) * 0.25 = 0.75 \end{aligned}$$

$$\begin{aligned} P(S_2 = 2) &= P(S_2 = 2|S_1 = 1) * P(S_1 = 1) + P(S_2 = 2|S_1 = 2) * P(S_1 = 2) \\ &= A(1,2) * 0.75 + A(2,2) * 0.25 = 0.25 \end{aligned}$$

And if t=3, the probability distribution depends on S_2 but has no relationship with S_1, then use the same method to do the calculation:

$$\begin{aligned} P(S_3 = 1) &= P(S_3 = 1|S_2 = 1) * P(S_2 = 1) + P(S_3 = 1|S_2 = 2) * P(S_2 = 2) \\ &= A(1, 1) * 0.75 + A(2, 1) * 0.25 = 0.75 \end{aligned}$$

$$\begin{aligned} P(S_3 = 2) &= P(S_3 = 2|S_2 = 1) * P(S_2 = 1) + P(S_3 = 2|S_2 = 2) * P(S_2 = 2) \\ &= A(1, 2) * 0.75 + A(2, 2) * 0.25 = 0.25 \end{aligned}$$

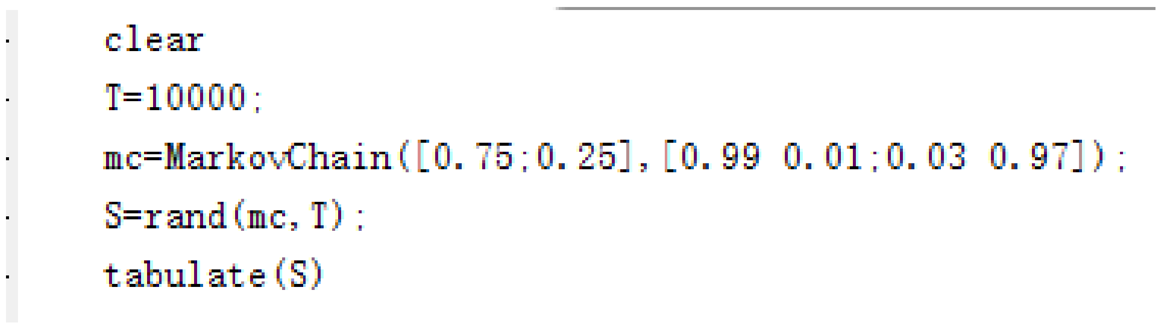
From t=2, each state's probability distribution only depends on the former states, which means only the distribution of S_{t-1} influences the distribution of S_t. To calculate the distribution of S_t, we always use the transition probability matrix A, and the probability distribution of S_t, at $t = 2$, equals to S_t, $t = 1$. In sum it shows that calculating P(S_t=j) is to repeat the calculation done before with the same value, thus P(S_t=j) are always be the same no matter the value of t is t=1,2,3,. . . . Then we draw the conclusion for P(S_t=j) as below.

$$\begin{aligned} P(S_t = 1) &= P(S_t = 1|S_{t-1} = 1) * P(S_{t-1} = 1) + P(S_t = 1|S_{t-1} = 2) * P(S_{t-1} = 2) \\ &= A(1, 1) * 0.75 + A(2, 1) * 0.25 = 0.75 \end{aligned}$$

$$\begin{aligned} P(S_t = 2) &= P(S_t = 2|S_{t-1} = 1) * P(S_{t-1} = 1) + P(S_t = 2|S_{t-1} = 2) * P(S_{t-1} = 2) \\ &= A(1, 2) * 0.75 + A(2, 2) * 0.25 = 0.25 \end{aligned}$$

2.2.2 Verify Markov chain rand function

The next task is to test the rand function of Markov chain, using MatLab, to check if the results correspond to the calculation above. The test code, used to do this, is shown in Fig. 2.1.

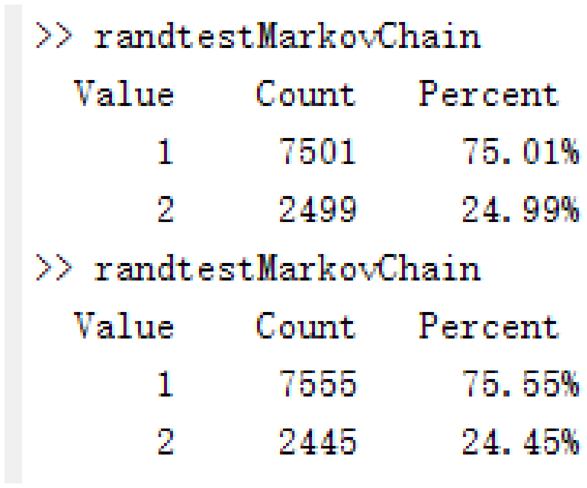


```
clear
T=10000;
mc=MarkovChain([0.75;0.25],[0.99 0.01;0.03 0.97]);
S=rand(mc,T);
tabulate(S)
```

Figure 2.1: Test code for Markov chain rand function

The test code is divided in several parts. First, we create the Markov Chain (MC) using the constructor method of a MC object 'mc = MarkovChain (pInit, pTrans)' in Markovchain class. Then, we get a random state sequence S of length T=10000. The next step is to find if the distribution of the state sequence has the same probabilities compared to the calculation results in 2.2.1, which are P(S_t=1)=0.75, P(S_t=2)=0.25.

Since values in state sequence are all positive integers, the test process uses 'tabulate' function of MatLab to count the values that appear in the random state sequence, with also the frequency of appearances for each value. The results can be seen in Fig. 2.2. They are not totally the same as calculated. It is because the sample size is not large enough. They are close to theoretical values, so they are valid.



```
>> randtestMarkovChain
  Value    Count   Percent
      1     7501     75.01%
      2     2499     24.99%
>> randtestMarkovChain
  Value    Count   Percent
      1     7555     75.55%
      2     2445     24.45%
```

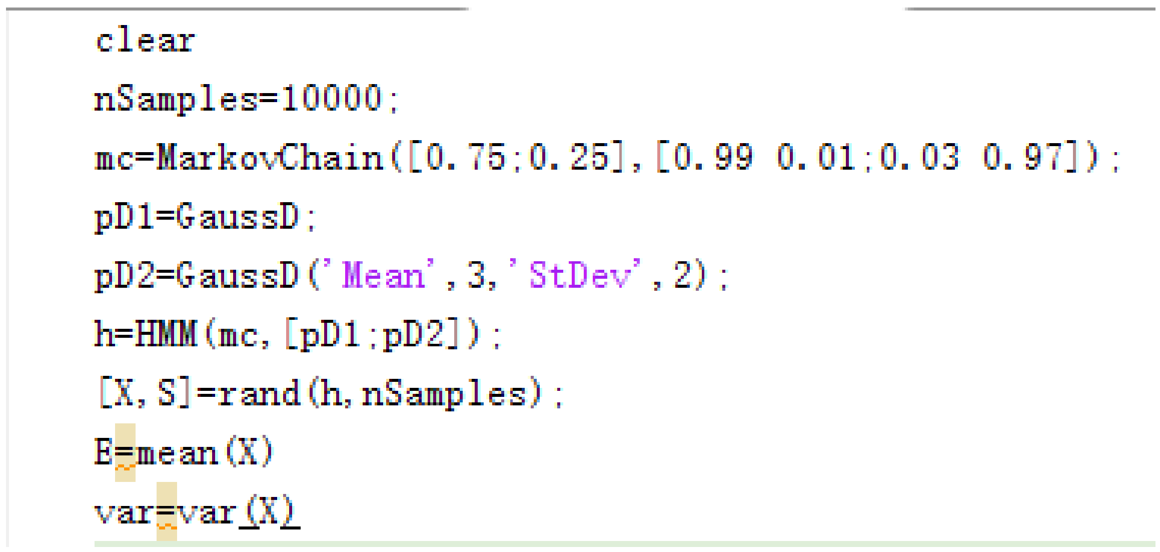
Figure 2.2: Results for test twice

2.2.3 Calculate E[X] and var[X]

We give the conditional expectation formulas μ_X = E [X] = E$_Z$ [E$_X$ [X|Z]], and var [X] = E$_Z$ [var$_X$ [X|Z]] + var$_Z$ [E$_X$ [X|Z]]. According to the expansion in the textbook, the E[X_t] and var[X_t] can be calculated as follows:

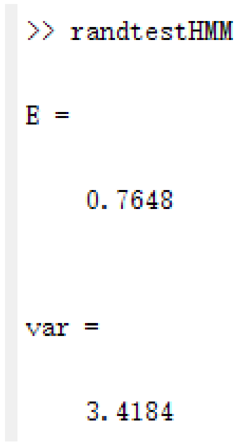
$$
\begin{aligned}
E[X] = E_Z[E_X[X|Z]] &= w_1 E_X[X|Z=1] + w_2 E_X[X|Z=2] = w_1\mu_1 + w_2\mu_2 \\
&= 0.75 * 0 + 0.25 * 3 = 0.75 \\
\text{var}[X] = E_Z[\text{var}_X[X|Z]] + \text{var}_Z[E_X[X|Z]] &= w_1 \text{var}_X[X|Z=1] + w_2 \text{var}_X[X|Z=2] \\
&\quad + w_1(E_X[X|Z=1]-\mu)^2 + w_2(E_X[X|Z=2]-\mu)^2 \\
&= w_1\sigma_1^2 + w_2\sigma_2^2 + w_1(\mu_1-\mu)^2 + w_2(\mu_2-\mu)^2 \\
&= 0.75 * 1 + 0.25 * 4 + 0.75 * (0-0.75)^2 + 0.25 * (3-0.75)^2 = 3.4375
\end{aligned}
$$

The test code (shown in Fig. 2.3 is constructed as the same way as the verification of the Markov chain. First, we create Markov chain using the method 'mc = MarkovChain (pInit, pTrans)' in Markovchain class. Then, we get the Gaussian random distribution vector B. And we use the Gaussian random vector and the Markov chain to build the HMM object h. Then we use the random method 'rand' of the HMM class to generate a random state sequence S of length nSamples=10000. Finally, we use the MatLab functions mean and var to calculate the mean and variance of the generated sequence X. The answers are shown as Fig. 2.4. Again because of the size of samples, it is close to theoretical values but are usually not the same.



```
clear
nSamples=10000;
mc=MarkovChain([0.75;0.25],[0.99 0.01;0.03 0.97]);
pD1=GaussD;
pD2=GaussD('Mean',3,'StDev',2);
h=HMM(mc,[pD1;pD2]);
[X,S]=rand(h,nSamples);
E=mean(X)
var=var(X)
```

Figure 2.3: Test code for HMM rand function



```
>> randtestHMM

E =

    0.7648


var =

    3.4184
```

Figure 2.4: Result for test

2.2.4 Plot of HMM behaviour with μ_1=0,μ_2=3

For $t = 500$ samples, we plot the figure 2.5 and study the variations of the variable X_t, on y-axis, given the time t, on x-axis. The output of this HMM should correspond to the Gaussian distribution with the output distributions $b_1(x)$ ∼N(0,1), $b_2(x)$ ∼N(3,4). After plotting it many times, the curve has two approximated levels, one for the state 1 with low variations, a big presence and a low level around 0, and one for the state 2 with big variations, a lower presence and a higher level around 3. The plot is shown in fig. 2.5.

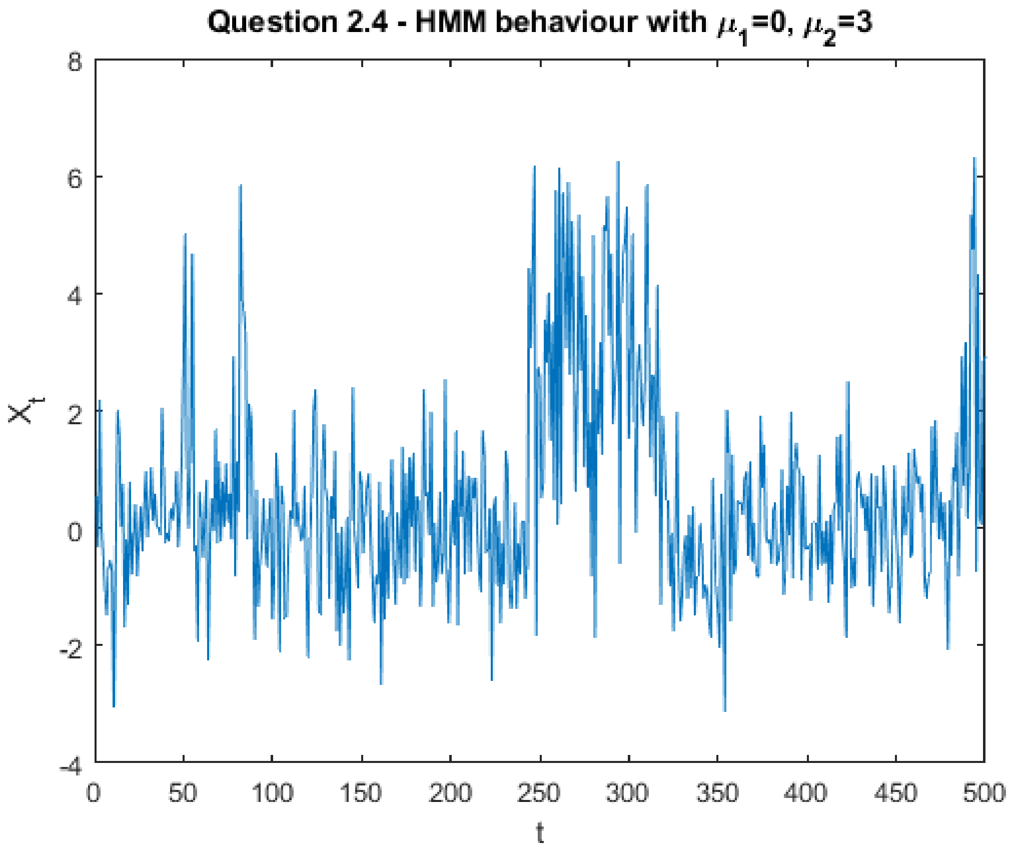


Figure 2.5: HMM behaviour with μ_1=0,μ_2=3

2.2.5 Plot of HMM behaviour with μ_1=μ_2=0

For μ_1=μ_2=0, the plot can be seen in fig. 2.6. Compared with the former HMM they both fluctuate along the same mean. But last case has two means and this case only has the same mean 0. So it is more difficult to distinguish the 2 states because the mean is the same. The only criterion to see the difference is the standard deviation, with be low variations for the state 1 and big variations for the state 2.

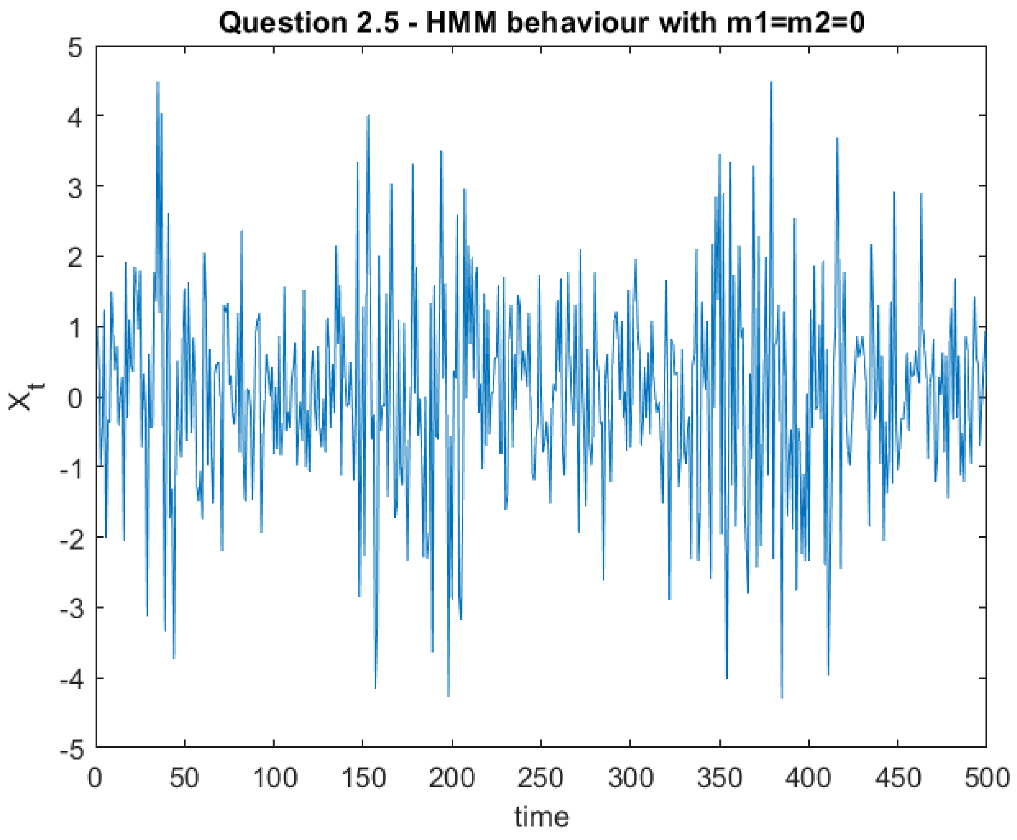


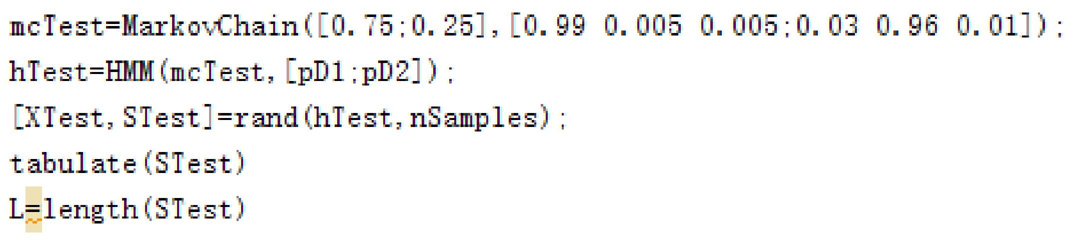
Figure 2.6: HMM behaviour with μ_1=μ_2=0

2.2.6 Check for finite-duration HMMs

In order to check if the rand function works for finite duration, we modify the transition matrix from 2*2 to 2*3 dimensions, and check if the process will reach the exit state and stop without producing any output. One way to verify is to compare the length of S with the number of random samples nSamples.

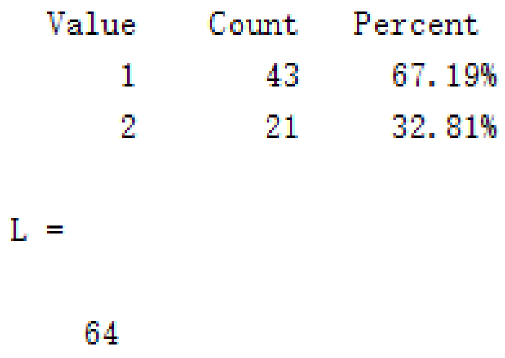
$$A = \begin{pmatrix} 0.99 & 0.005 & 0.005 \\ 0.03 & 0.96 & 0.01 \end{pmatrix};$$

For example, if we use the transition matrix of A as above, we can check the length of the sequence after generating a random sequence in the output. So we obtain the output sequence length $L = 64 \leqslant nSamples$, which means this is a finite duration sequence with its exit state.



```
mcTest=MarkovChain([0.75;0.25],[0.99 0.005 0.005;0.03 0.96 0.01]);
hTest=HMM(mcTest,[pD1;pD2]);
[XTest,STest]=rand(hTest,nSamples);
tabulate(STest)
L=length(STest)
```

Figure 2.7: Test code for finite-duration HMMs



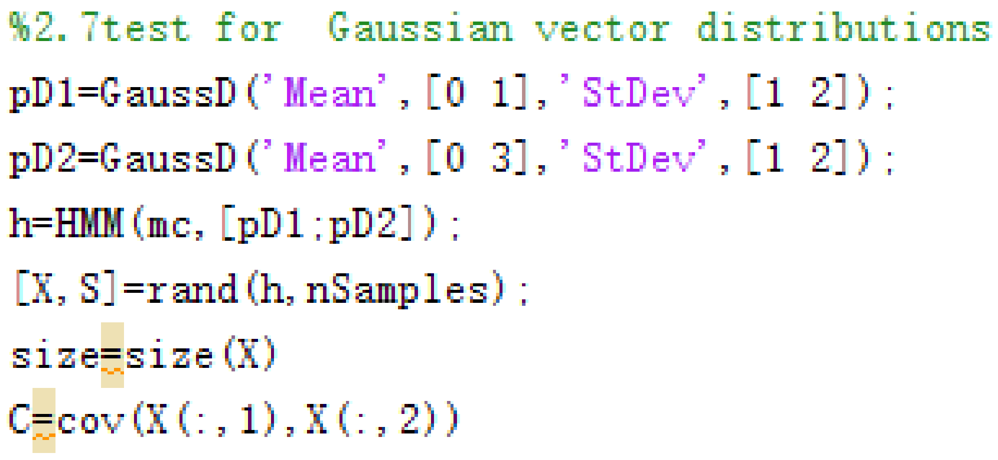
```
  Value    Count   Percent
      1       43    67.19%
      2       21    32.81%


L =


    64
```

Figure 2.8: Output the length of sequence

2.2.7 Work with Gaussian vector distributions

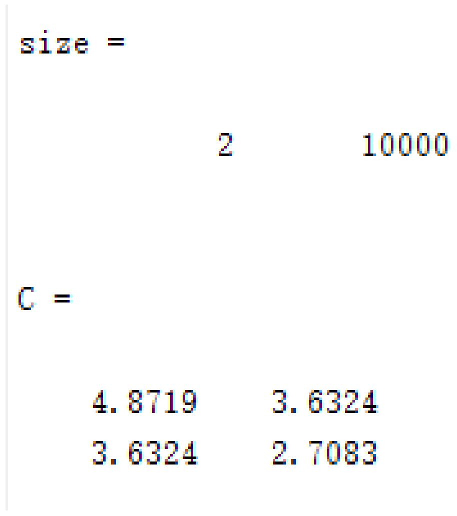
In order to check if the HMMs function works with Gaussian vector distributions output, we use the GaussD method to generate Gaussian vectors and generate HMMs model. Then we check if the output distributions have a non-diagonal covariance matrix. The test code is shown in fig. 2.9.

By calculating the covariance of the output matrix, in the above Gaussian vector, the size of C shown in the output is 2∗2, which prove the effectiveness of the function.



```
%2.7test for  Gaussian vector distributions
pD1=GaussD('Mean',[0 1],'StDev',[1 2]);
pD2=GaussD('Mean',[0 3],'StDev',[1 2]);
h=HMM(mc,[pD1;pD2]);
[X,S]=rand(h,nSamples);
size=size(X)
C=cov(X(:,1),X(:,2))
```

Figure 2.9: Test code for Gaussian vector distribution



```
size =

           2       10000



C =

    4.8719    3.6324
    3.6324    2.7083
```

Figure 2.10: Covariance from output