Assignment 1
EQ2341 Pattern Recognition and Machine Learning

Names
Oriol Closa and Clara Escorihuela Altaba

Date
January 23, 2022

1 Questions

1.1 To verify your Markov chain code, calculate $P(S_t = j)$, $j \in 1, 2$ for $t = 1, 2, 3, ...$ theoretically, by hand, to verify that $P(S_t = j)$ is actually constant for all t.

In order to prove that $P(S_t = j)$ is constant for an arbitrary j we need to verify that it is time invariant and therefore it represents a stationary distribution. Such a distribution fulfils equation 1.

$$p = A^T p \tag{1}$$

In order to exemplify this with the previously provided data, equation 2 needs to hold for 1.

$$p_{t+1} = A^T p_t \tag{2}$$

$$p_2 = A^T p_1 = A^T q = \begin{pmatrix} 0,99 & 0,03 \\ 0,01 & 0,97 \end{pmatrix} \cdot \begin{pmatrix} 0,75 \\ 0,25 \end{pmatrix} = \begin{pmatrix} 0,75 \\ 0,25 \end{pmatrix} \implies p = A^T p \tag{3}$$

Equation $p = A^T p$ is equivalent to $pA^T = \lambda p$ when $\lambda = 1$. The solution to this equation is $\begin{pmatrix} 0,949 & 0,316 \end{pmatrix}$, therefore all the scalar vectors originated from this will make this distribution stationary, like in the case presented before.

1.2 Use your Markov chain rand function to generate a sequence of $t = 10000$ state integer numbers from the test Markov chain. Calculate the relative frequency of occurrences of $S_t = 1$ and $S_t = 2$ The relative frequencies should of course be approximately equal to $P(S_t)$.

To empirically prove that the generation behaves according to the expected distribution, we create a sequence of 10.000 states and count the number of occurrences for each state. This means that approximately 75% of the states will correspond to $S_t = 1$ and 25% of them to $S_t = 2$. The following table shows the empirical results where we can see the distribution with a single sequence and also the mean of 100 different generated sequences.

State	1 run	100 runs
$S_t = 1$	$0,745$	$0,756$
$S_t = 2$	$0,255$	$0,243$

Table 1: Empirical results with $t = 10.000$

1.3 To verify your HMM `rand` method, first calculate $E[X_t]$ and $var[X_t]$ theoretically. The conditional expectation formulas $\mu_x = E[X] = E_z[E_x[X|Z]]$ and $var[X] = E_z[var_x[X|Z]] + var_z[E_x[X|Z]]$ apply generally whenever some variable X depends on another variable Z and may be useful for the calculations. Then use your HMM rand function to generate a sequence of $T = 10000$ output scalar random numbers $x = (x_1 ... x_t ... x_T)$ from the given HMM test example. Use the standard Numpy functions $np.mean()$ and $np.var()$ to calculate the `mean` and `variance` of your generated sequence. The result should agree approximately with your theoretical values.

$$\begin{aligned} E[X_t] &= E_{S_t}[E_{X_t}[X_t|S_t]] \\ &= P(S_t = 1) \cdot \mu_1 + P(S_t = 2) \\ &= 0.75 \cdot 0 + 0.25 \cdot 3 = 0.75 \end{aligned} \tag{4}$$

$$var[X_t] = E_{S_t}[var_{X_t}[X|Z]] + var_z[E_{x_t}[X_t|S_t]]$$
$$= \sum_{j=1}^{2} P(S_t = j) \cdot \theta_j^2 + \sum_{j=1}^{2} P(S_t = j) \cdot (\mu_j - E[X_t]^2) \tag{5}$$
$$= 0,71 \cdot 1 + 0,25 \cdot 4 + 0,75 \cdot (0,75 - 0)^2 - 0,25 \cdot (0,75 - 3)^2 = 3,428$$

Metric	1 run	100 runs
Mean	$0,746$	$0,748$
Variance	$3,361$	$3,422$

Table 2: Results with $t = 10.000$

1.4 To get an impression of how the HMM behaves, use `@HMM/rand`to generate a series of 500 contiguous samples X_t from the HMM, and plot them as a function of t. What characterizes the typical output of this HMM?

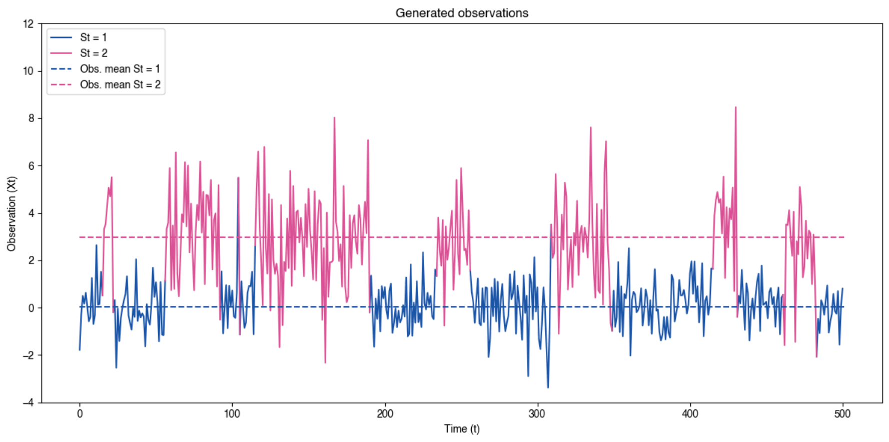


Figure 1: Generated observed sequence with the different states shown

As we can see in figure 1 the amount of times we change states is much lower than the number of times we transition to the same. In particular, for this specific plot we can see this happens 16 times but this is an arbitrary number and can be lower or higher depending on the run. This is due to the fact that the transition probability of repeating the same state for both $P(S_t = 1)$ and $P(S_t = 2)$ is much higher than the one that makes it change. If that was not the case, for example exactly the opposite, we would see a change almost for every t and essentially no state repetitions.

Moreover, figure 1 also presents the means for both classes, which closely match the initial values for both gaussians, 0 for $P(S_t = 1)$ and 3 for $P(S_t = 2)$.

1.5 Create a new HMM, identical to the previous one except that it has $\mu_1 = \mu_2 = 0$. Generate and plot 500 contiguous values several times using `@HMM/rand` for this HMM. What is similar about how the two HMMs behave? What is different with this new HMM? Is it possible to estimate the state sequence S of the underlying Markov chain from the observed output variables x in this case?

In figure 2 we included the new version of the HMM while keeping the previous one as a reference in a softer colour in order to facilitate comparisons. As the figure presents, the observations for the second state are now located closer to 0 corresponding as well to the mean for the first state. If we did not have the information about what observation belongs to which state, we would not be able to visually

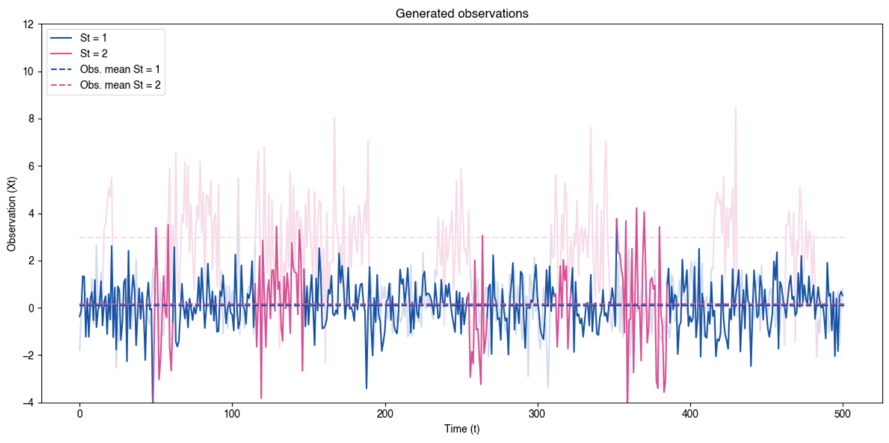


Figure 2: Generated observed sequence for the new means

identify the one in which they belong. With the original distributions however, it was much easier to differentiate them. On the other hand, as it also happens for the previous model, the deviation stays changed for both distributions and creates a more spread pattern for the second state.

Regarding the possibility of identifying the hidden state sequence we must consider the transition A and the observation matrix B. The former does not depend on the distribution of the observations, for that reason it can keep the same values in both HMMs. However, the observation matrix does, consequently, if the distributions vary they will also differ.

Because of the fact both gaussians have the same mean $(\mu_{S_1} = \mu_{S_2} = 0)$ it would be really hard to determine if an observation closer to that point (μ) belongs to the first or the second state. However, because they have different deviations where the second distribution has a higher value than the first, the further an observation is from the mean, the more probable it will belong to the second. That is, the areas where the second gaussian do not overlap with the first.

To sum up, in the second case the observation matrix will present very similar probabilities which will creates a difficult way to determine the state sequence of the Hidden Markov Model.

1.6 Another aspect you must check is that your `rand` function works for finite-duration HMMs. Define a new test HMM of your own and verify that your function returns reasonable results.

In order to test if our implementation works for finite-duration HMMs we modify the initial transition matrix adding a third column which represents the $S_t = n + 1$ state while making sure the matrix is still row stochastic.

$$q = \begin{pmatrix} 0,75 \\ 0,25 \end{pmatrix} \qquad A = \begin{pmatrix} 0,54 & 0,36 & 0,1 \\ 0,15 & 0,75 & 0,1 \end{pmatrix} \tag{6}$$

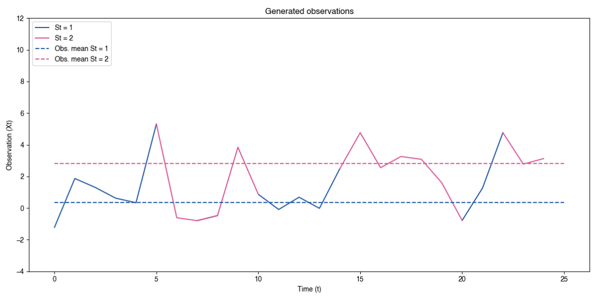
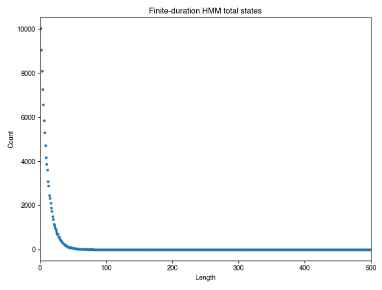
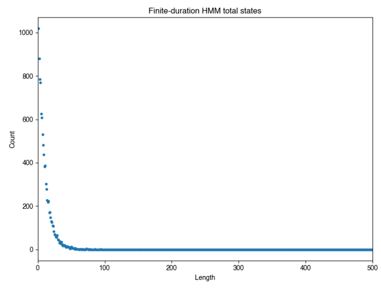


Figure 3: Random generated sequence terminating in $t = 24$

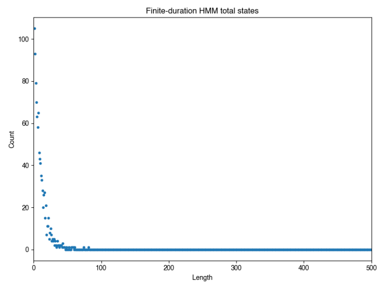
Figure 3 represents a given generated observation sequence when the final state has been reached in $t = 24$. On the other hand, in figure 4 we can see what the effect of a finite-duration HMM has on the length of the generated sequence. In particular, the probability of transitioning to the final state from either $S_t = 1$ or $S_t = 2$ is 10%. Therefore, we empirically expect approximately a 10% of the total generated sequences to reach the final state in $t = 1$. From this point onward, there should also be a 10% of sequences from $j = i$ that terminate in $j = i + 1$.



(a) 100.000 iterations



(b) 10.000 iterations



(c) 1.000 iterations

Figure 4: Distribution of the number of total states for the previous finite-duration HMM

1.7 Finally, your `rand` function should work also when the state-conditional output distributions generate random vectors. Define a new test HMM of your own where the outputs are Gaussian vector distributions, and verify that this also works with your code. (Note that a single instance of the GaussD class is capable of generating vector output; stacking several GaussD-objects is not correct).

In order to prove that our code also generates sequences of observations that contain random vectors, we have changed the output probability distributions for 2D gaussians. Equation 7 presents the final parameters that define these distributions.

$$\begin{aligned} B &= \begin{pmatrix} b_1(x) \\ b_2(x) \end{pmatrix} \\ b_1(x) &\sim \mathcal{N}\left(\mu_1 = \begin{bmatrix} \mu_{11} = 0 \\ \mu_{12} = 10 \end{bmatrix}, C_1 = \begin{bmatrix} 2 & 1 \\ 1 & 4 \end{bmatrix}\right) \\ b_2(x) &\sim \mathcal{N}\left(\mu_2 = \begin{bmatrix} \mu_{21} = 5 \\ \mu_{22} = 12 \end{bmatrix}, C_2 = \begin{bmatrix} 1 & 0 \\ 0 & 3 \end{bmatrix}\right) \end{aligned} \tag{7}$$

Figure 5 presents a 3D plot of the observation sequence when they are generated from two 2D gaussians. The observations in blue belong to the $b_1(x)$ distribution, that is why they expose values around 0 for feature 1 and around 5 for feature 2, which correspond to μ_{11} and μ_{21}. On the other hand, pink observations are part of the $b_2(x)$ distribution, consequently they present values around 10 and 12 for features 1 and 2 respectively, which also correspond to μ_{12} and μ_{22}.

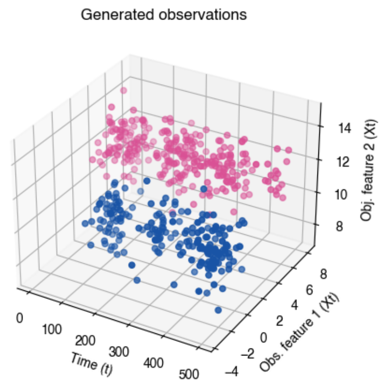


Figure 5: 3D plot for the new HMMs