EQ2340 - Pattern Recognition
Assignment 1: HMM Signal Source

Tianxiao Zhao tzh@kth.se
Zheyu Zhang zheyuz@kth.se

November 9, 2017

Problem 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.

Solution

$$P(S_{t+1} = j) = \sum_{i=1}^{N} P(S_{t+1} = j, S_t = i) = \sum_{i=1}^{N} \underbrace{P(S_{t+1} = j \mid S_t = i)}_{a_{ij}} \; P(S_t = i) \tag{0.1}$$

The relation above connects the state probabilities at consecutive time steps. To make it more concise, we can rewrite it into a matrix notation in this specific case

$$\begin{bmatrix} P(S_{t+1} = 1) \\ P(S_{t+1} = 2) \end{bmatrix} = A^T \begin{bmatrix} P(S_t = 1) \\ P(S_t = 2) \end{bmatrix} \;\Leftrightarrow\; \boldsymbol{p}_{t+1} = A^T \; \boldsymbol{p}_t \tag{0.2}$$

We start from $t = 1$, then $\boldsymbol{p}_2$ can be calculated based on equation 0.2

$$\boldsymbol{p}_2 = A^T \; \boldsymbol{q} = \begin{pmatrix} 0.99 & 0.03 \\ 0.01 & 0.97 \end{pmatrix} \cdot \begin{pmatrix} 0.75 \\ 0.25 \end{pmatrix} = \begin{pmatrix} 0.75 \\ 0.25 \end{pmatrix} \;\Rightarrow\; \boldsymbol{p}_2 = A^T \; \boldsymbol{q} \tag{0.3}$$

Since the transition probability matrix is time-invariant, by keeping iterating the calculation above, we can conclude that $\boldsymbol{p}_t = \begin{bmatrix} P(S_t = 1) & P(S_t = 2) \end{bmatrix}^T = \begin{bmatrix} 0.75 & 0.25 \end{bmatrix}^T$ for all t.

To verify that $P(S_t = j)$ is constant for all t is equivalent to prove that this distribution is stationary, which means the distribution should satisfy following equation

$$\boldsymbol{p} = A^T \; \boldsymbol{p} \tag{0.4}$$

If equation 0.4 holds, then any eigenvector of the transposed transition probability matrix A^T , with eigenvalue 1, is a possible stationary probability vector. After calculation, the eigenvector with eigenvalue 1 in this case is $\begin{bmatrix} 0.9487 & 0.3162 \end{bmatrix}^T$, which turns out to be a scale of our initial state probability vector $\boldsymbol{q} = \begin{bmatrix} 0.75 & 0.25 \end{bmatrix}^T$. Therefore, we can prove that our case is a stationary probability distribution, and $\boldsymbol{p}_t = \begin{bmatrix} 0.75 & 0.25 \end{bmatrix}^T$ is actually constant for all t.

Problem 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 $P(S_t)$.

Solution

According to the instruction, we run our function `@MarkovChain/rand` to generate sequences of $T = 10000$ states and use them to compute the relative frequency of occurrences of $S_t = 1$ and $S_t = 2$. Since the states are randomly picked and linked in order, we think it makes more sense to first generate 500 sequences of 10000 states, count the frequency of occurrence for each state of each sequence, and then average them. The results are shown below, and they are approximately equal to the analytical results $P(S_t)$ in Problem 1. This means that our function `@MarkovChain/rand` is correctly implemented.

```
> ------- verify MarkovChain rand method with 500 trials -------
> Average freq of St = 1: 0.75092
> Average freq of St = 2: 0.24908
```

Problem 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 $\underline{x} = (x_1...x_t...x_T)$ from the given HMM test example. Use the standard MatLab functions `mean` and `var` to calculate the mean and variance of your generated sequence. The result should agree approximately with your theoretical values.

Solution

Theoretically, we can calculate $E[X_t]$ and $var[X_t]$ by expanding the conditional expectation formulas mentioned above:

$$\begin{aligned} E[X_t] &= E_{S_t}[E_{X_t}[X_t \mid S_t]] = P(S_t = 1) \cdot \mu_1 + P(S_t = 2) \cdot \mu_2 \\ &= 0.75 \cdot 0 + 0.25 \cdot 3 = 0.75 \\ var[X_t] &= E_{S_t}[var_{X_t}[X_t \mid S_t]] + var_{Z_t}[E_{X_t}[X_t \mid S_t]] \\ &= \sum_{j=1}^{2} P(S_t = j) \cdot \sigma_j^2 + \sum_{j=1}^{2} P(S_t = j) \cdot (\mu_j - E[X_t])^2 \\ &= 0.75 \cdot 1 + 0.25 \cdot 4 + 0.75 \cdot (0.75 - 0)^2 + 0.25 \cdot (0.75 - 3)^2 = 3.4375 \end{aligned} \tag{0.5}$$

We implement this experiment by first generating 500 sequences of 10000 states, and then averaging the means and variances. The averaged results are shown below. It is clear to see that this results are quite close to those calculated before. Therefore, we can make sure that our HMM `rand` method is correct.

```
> ------- verify HMM rand method with 500 trials -------
> Average mean of Xt: 0.75985
> Average variance of Xt: 3.4178
```

Problem 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. Do this many times until you have a good idea of what characterizes typical output of this HMM, and what structure there is to the randomness. Describe the behaviour in one or two sentences in your report. Also include one such plot in the report, labelled using `title`, `xlabel`, and `ylabel` to clearly show which variable is plotted along which axis. You should do this for every plot in the course project.

Solution

Figure 1 is an example generated by our HMM signal source. To understand the output of this HMM, we also plot the state transition as time goes. From this figure, we can observe following behaviors:

- Samples X_t mainly oscillates between two values - 0 and 3, due to the fact that the two sub-sources are Gaussian distributions with a mean of 0 and 3 respectively. Also a difference in variance leads to a smaller jitter around value 0 and a larger one around value 3;
- X_t changes its value as S_t transits to a different state. And this HMM tends to be trapped within a state for a long time since $P(S_{t+1} = 1 \mid S_t = 1) = 0.99$ and $P(S_{t+1} = 2 \mid S_t = 2) = 0.97$ for all t;
- The transition of state S_t (or transition probability matrix A) and output probability distribution of sub-sources $b_j(\mathbf{x})$ jointly characterizes the output of this HMM.

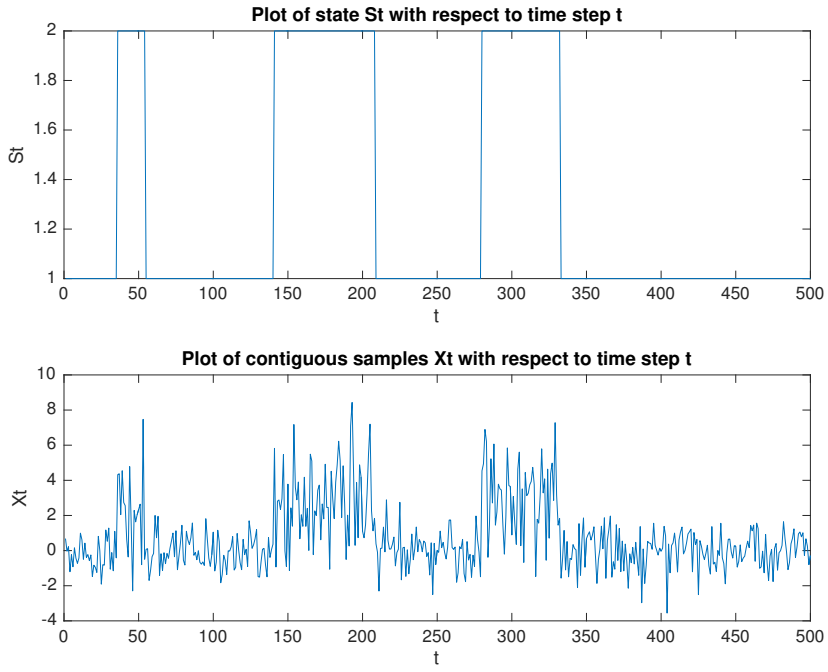

Figure 1: A sequence of 500 contiguous samples from the HMM source.

Problem 5

Create a new HMM, identical to the previous one except that it has $\mu_2 = \mu_1 = 0$. 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 $\underline{S}$ of the underlying Markov chain from the observed output variables $\underline{x}$ in this case?

Solution

It is similar that both HMMs in these two cases tend to stay within one state for a long time since the transition probability matrix has dominant entries in diagonal and it is time-invariant. And we can also observe that a higher spread appears when state transits from 1 to 2, and a lower spread comes back when it is the other way around. However, it is harder to distinguish between source 1 and 2 because of the same means of Gaussian distributions $b_1(x)$ and $b_2(x)$. From Figure 1, it is obvious that the high values correspond to state $S_t = 2$ and low values correspond to state $S_t = 1$. Figure 2 is an example generated with the new HMM which has the same means for $b_1(x)$ and $b_2(x)$. Clearly, it is almost impossible to find the relation between outputs and states by pure observation. This difference can also be indicated by an average variance of two sequences. The average variance of the new sequence is around 1.7395 after 500 experiments, which is much less that from previous section. This means that the new sequence has a lower spread around the mean which makes it more like a white noise, so it is harder to estimate the state sequence.

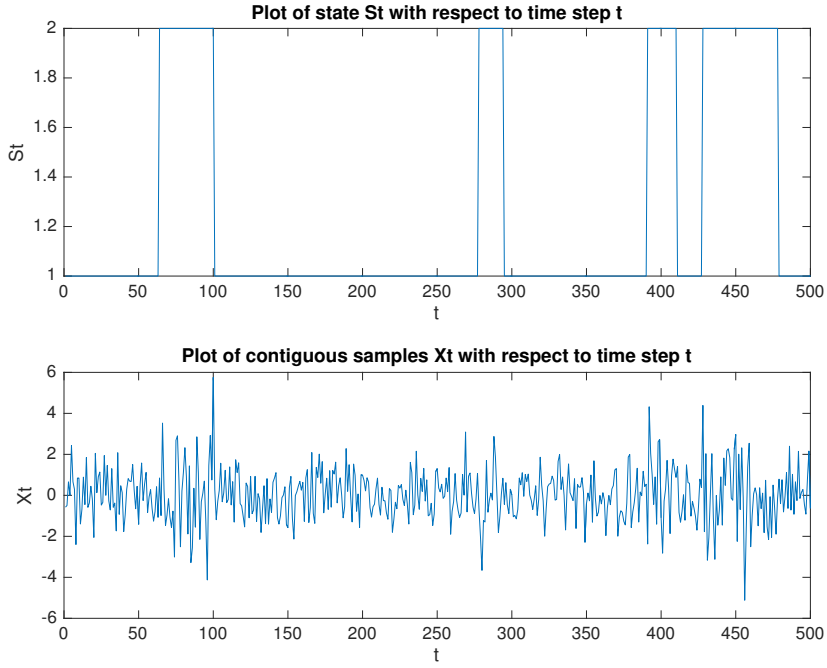

Figure 2: A sequence of 500 contiguous samples from the same-mean HMM source.

Problem 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.

Solution

To test our `rand`-function for *finite-duration* HMMs, we re-define our HMM source with the following parameters:

$$q = \begin{pmatrix} 0.75 \\ 0.25 \end{pmatrix};\ A = \begin{pmatrix} 0.2 & 0.5 & 0.3 \\ 0.4 & 0.4 & 0.2 \end{pmatrix};\ B = \begin{pmatrix} b_1(x) \\ b_2(x) \end{pmatrix} \tag{0.6}$$

where $b_1(x)$ and $b_2(x)$ are kept the same with previous case. According to the definition of finite-duration HMM, we resize the transition probability matrix to a 2×3 matrix, with the third column of entries representing the probabilities of exiting.

Same as before, we run our program to generate a sequence of output with an expected length of 500 samples, but after 7 steps the process goes to the special exit state, see Figure 3.

To further verify that our function returns reasonable results, we were thinking about running this HMM repeatedly, e.g. for 2000 times, and recording counts of different lengths of sequence when the process reaches the exit state, and then calculating the probabilities of exiting HMM process after different steps. The statistical results are listed below:

```
> ------- verify finite HMM (sequence length from 1 to 5) -------
> Prob of sequence length = 1: 0.2735
> Prob of sequence length = 2: 0.1725
> Prob of sequence length = 3: 0.1395
> Prob of sequence length = 4: 0.086
> Prob of sequence length = 5: 0.0865
```

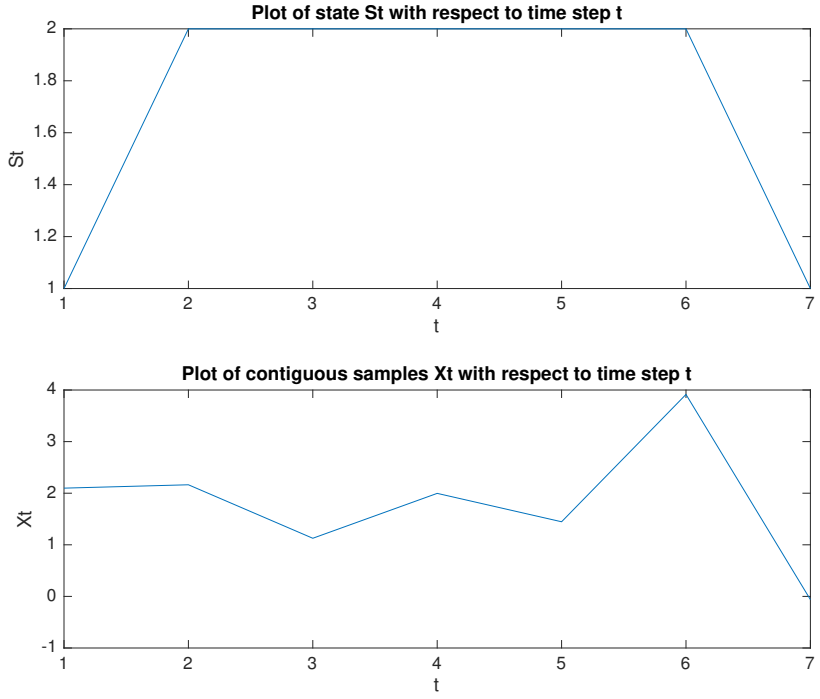

Figure 3: An example of output sequence sampled from the finite-duration HMM source.

The probability of exiting HMM process after 1 step can be computed by hand as following.

$$\begin{aligned} P(S_2 = 3) &= \sum_{i=1}^{2} P(S_2 = 3, S_1 = i) = \sum_{i=1}^{2} P(S_2 = 3 \mid S_1 = i) \cdot P(S_1 = i) \\ &= a_{1,3} \cdot q_1 + a_{2,3} \cdot q_2 = 0.3 \cdot 0.75 + 0.2 \cdot 0.25 = 0.275 \end{aligned} \tag{0.7}$$

Similarly, we could also obtain the probability of exiting HMM process after 2 steps, see equations below. To make it concise, we skip the calculations of probabilities of more-than-three-step HMMs, but it won't affect the final conclusion when it continues.

$$\begin{aligned} P(S_2 = 1) &= \sum_{i=1}^{2} P(S_2 = 1, S_1 = i) = \sum_{i=1}^{2} P(S_2 = 1 \mid S_1 = i) \cdot P(S_1 = i) \\ &= a_{1,1} \cdot q_1 + a_{2,1} \cdot q_2 = 0.2 \cdot 0.75 + 0.4 \cdot 0.25 = 0.25 \\ P(S_2 = 2) &= \sum_{i=1}^{2} P(S_2 = 2, S_1 = i) = \sum_{i=1}^{2} P(S_2 = 2 \mid S_1 = i) \cdot P(S_1 = i) \\ &= a_{1,2} \cdot q_1 + a_{2,2} \cdot q_2 = 0.5 \cdot 0.75 + 0.4 \cdot 0.25 = 0.475 \\ P(S_3 = 3) &= \sum_{i=1}^{2} P(S_3 = 3, S_2 = i) = \sum_{i=1}^{2} P(S_3 = 3 \mid S_2 = i) \cdot P(S_2 = i) \\ &= a_{1,3} \cdot P(S_2 = 1) + a_{2,3} \cdot P(S_2 = 2) = 0.3 \cdot 0.25 + 0.2 \cdot 0.475 = 0.17 \end{aligned} \tag{0.8}$$

Since these analytical probabilities are approximate to the empirical results above, we can guarantee that our function could give reasonable results.

The code that generates above experimental results is also attached here for reference:

```
%% ============== main part (from verify.m) ==============
clear;

%% build up HMM sources
```

```
test_name = 'finite-duration HMM';
para = testCasePara(test_name);
mc = MarkovChain(para.q, para.A);
h = HMM(mc, [para.b1; para.b2]);

%% verify finite HMM
if strcmp(test_name, 'finite-duration HMM')
    nSamples = 500;
    n_trial = 2000;
    max_steps = 10;
    S_len_prob = zeros(1, nSamples);

    % calculate exiting probabilities after different steps
    for n = 1 : n_trial
        [X, S] = rand(h, nSamples);
        S_len_prob(length(S)) = S_len_prob(length(S)) + 1;
    end
    S_len_prob = S_len_prob/n_trial;

    % output results
    disp('-------- verify finite HMM --------');
    for i = 1 : max_steps
        disp(['Prob of sequence length = ' num2str(i) ': ' num2str(S_len_prob(i))]);
    end
end


%% ========== parameters for different test case (from testCasePara.m) ==========
function para = testCasePara(test_name)

switch test_name
    case 'regular HMM'
        f1 = 'q';   q = [0.75; 0.25];
        f2 = 'A';   A = [0.99 0.01; 0.03 0.97];
        f3 = 'b1';  b1 = GaussD('Mean', 0, 'StDev', 1);
        f4 = 'b2';  b2 = GaussD('Mean', 3, 'StDev', 2);
    case 'same-mean HMM'
        f1 = 'q';   q = [0.75; 0.25];
        f2 = 'A';   A = [0.99 0.01; 0.03 0.97];
        f3 = 'b1';  b1 = GaussD('Mean', 0, 'StDev', 1);
        f4 = 'b2';  b2 = GaussD('Mean', 0, 'StDev', 2);
    case 'finite-duration HMM'
        f1 = 'q';   q = [0.75; 0.25];
        f2 = 'A';   A = [0.2 0.5 0.3; 0.4 0.4 0.2];
        f3 = 'b1';  b1 = GaussD('Mean', 0, 'StDev', 1);
        f4 = 'b2';  b2 = GaussD('Mean', 3, 'StDev', 2);
    case 'vector-output HMM'
        f1 = 'q';   q = [0.75; 0.25];
        f2 = 'A';   A = [0.99 0.01; 0.03 0.97];
        f3 = 'b1';  b1 = GaussD('Mean', [0; 20], 'Covariance', [2 1; 1 4]);
        f4 = 'b2';  b2 = GaussD('Mean', [10; 40], 'Covariance', [1 0; 0 2]);
    otherwise
        warning('Wrong test case name!')
end

para = struct(f1, q, f2, A, f3, b1, f4, b2);
```

Problem 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.) At least one of the output distributions should have a non-diagonal covariance matrix such as

$$\Sigma = \begin{pmatrix} 2 & 1 \\ 1 & 4 \end{pmatrix}$$

.

Solution

To verify that our `rand`-function also works when the state-conditional output distributions generate random vectors, we change the output probability distributions into the following form:

$$B = \begin{pmatrix} b_1(x) \\ b_2(x) \end{pmatrix},\ b_1(x) \sim \mathcal{N}\left(\mu_1 = \begin{bmatrix} 0 \\ 20 \end{bmatrix}, C_1 = \begin{bmatrix} 2 & 1 \\ 1 & 4 \end{bmatrix}\right),\ b_2(x) \sim \mathcal{N}\left(\mu_2 = \begin{bmatrix} 10 \\ 40 \end{bmatrix}, C_2 = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}\right) \quad (0.9)$$

We generate a sequence after 500 time steps, and obtain a 2×500 matrix as output since each sub-source is a two-dimensional Gaussian distribution. The result is shown below, see Figure 4 and 5. In the first graph, we can easily recognize state transitions based on value changes of both features. And the two Gaussian distributions are also well separated and distinguished. In figure 5, output sequence is divided into two clusters, and both of the clusters are Gaussian-shape like. For state $S = 1$, the cluster centers at (0, 20) and has a standard deviation of 1.4 in roughly the lower left-upper right direction and of 2 in the orthogonal direction (note that units of the two axes are not the same), which corresponds to the parameter settings (b_1) of Gaussian distribution of the first sub-source. And this also works for state $S = 2$. Thus, we can say that our implementation of `rand`-function works well in this case.

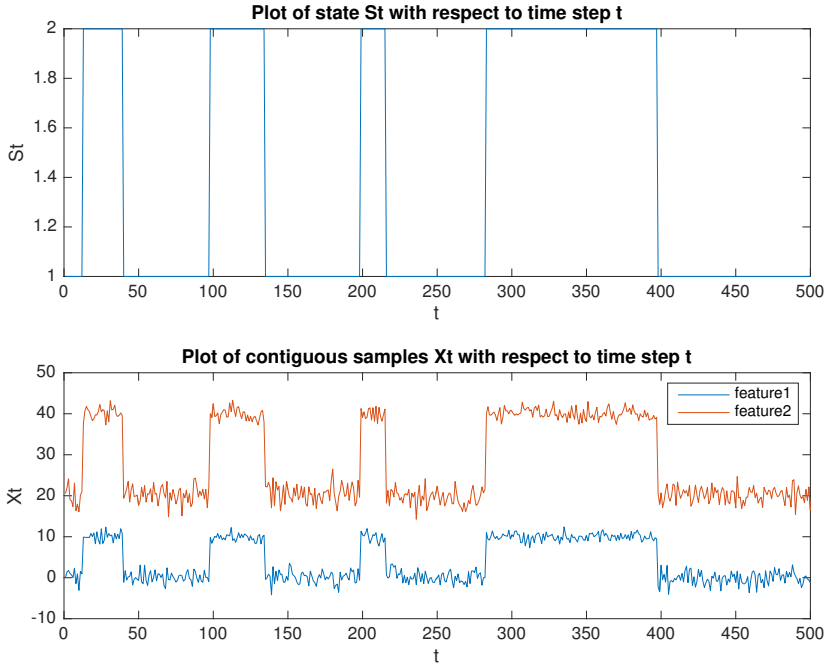

Figure 4: A sequence of 500 contiguous samples from the vector-output HMM source.

The code below is used to verify that vector output distributions work with our implementation:

```
%% ============== main part (from verify.m) ==============
clear;

%% build up HMM sources
test_name = 'vector-output HMM';
para = testCasePara(test_name);
mc = MarkovChain(para.q, para.A);
h = HMM(mc, [para.b1; para.b2]);

%% HMM impression
nSamples = 500;
[X, S] = rand(h, nSamples);

% view results
figure;
subplot(2, 1, 1);
plot(1:length(S), S);
```

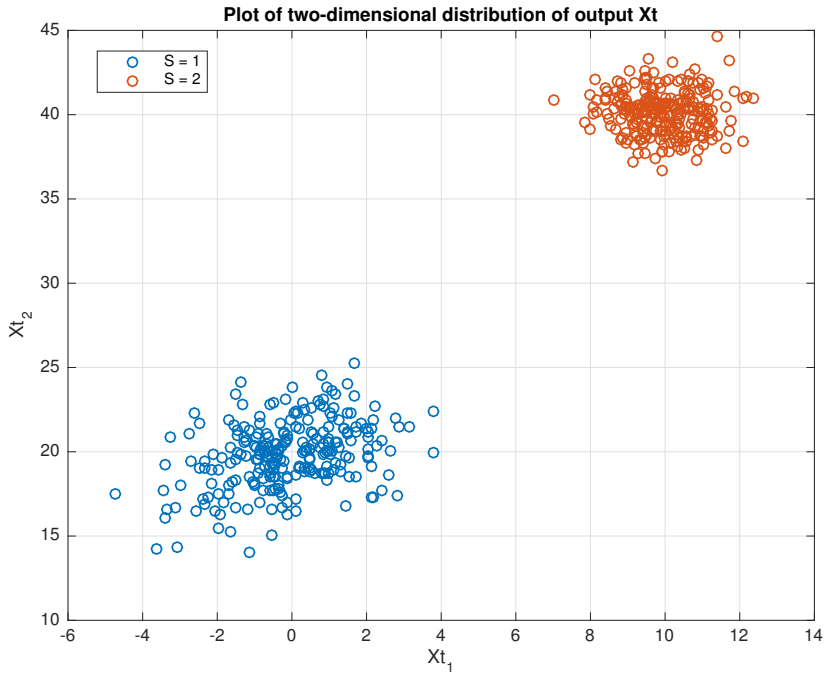

Figure 5: Two dimensional distribution of the output sequence. Xt_1 and Xt_2 denote features from two dimensions respectively.

```
18  title('Plot of state St with respect to time step t')
19  xlabel('t'); ylabel('St');
20  subplot(2, 1, 2);
21  plot(1:length(X), X);
22  title('Plot of contiguous samples Xt with respect to time step t')
23  xlabel('t'); ylabel('Xt');
24  legend('feature 1', 'feature 2');
25
26  %% verify vector-output HMM
27  if strcmp(test_name, 'vector-output HMM')
28      figure;
29      plot(X(1, S == 1), X(2, S == 1), 'o');
30      hold on; grid on;
31      plot(X(1, S == 2), X(2, S == 2), 'o');
32      title('Plot of two-dimensional distribution of output Xt');
33      xlabel('Xt_1'); ylabel('Xt_2');
34      legend('S = 1', 'S = 2');
35  end
36
37  %% ========== parameters for different test case (from testCasePara.m) ==========
38  function para = testCasePara(test_name)
39
40  switch test_name
41      case 'regular HMM'
42          f1 = 'q';   q = [0.75; 0.25];
43          f2 = 'A';   A = [0.99 0.01; 0.03 0.97];
44          f3 = 'b1';  b1 = GaussD('Mean', 0, 'StDev', 1);
45          f4 = 'b2';  b2 = GaussD('Mean', 3, 'StDev', 2);
46      case 'same-mean HMM'
47          f1 = 'q';   q = [0.75; 0.25];
48          f2 = 'A';   A = [0.99 0.01; 0.03 0.97];
49          f3 = 'b1';  b1 = GaussD('Mean', 0, 'StDev', 1);
50          f4 = 'b2';  b2 = GaussD('Mean', 0, 'StDev', 2);
51      case 'finite-duration HMM'
52          f1 = 'q';   q = [0.75; 0.25];
53          f2 = 'A';   A = [0.2 0.5 0.3; 0.4 0.4 0.2];
54          f3 = 'b1';  b1 = GaussD('Mean', 0, 'StDev', 1);
55          f4 = 'b2';  b2 = GaussD('Mean', 3, 'StDev', 2);
56      case 'vector-output HMM'
```

```
57          f1 = 'q';   q = [0.75; 0.25];
58          f2 = 'A';   A = [0.99 0.01; 0.03 0.97];
59          f3 = 'b1';  b1 = GaussD('Mean', [0; 20], 'Covariance', [2 1; 1 4]);
60          f4 = 'b2';  b2 = GaussD('Mean', [10; 40], 'Covariance', [1 0; 0 2]);
61      otherwise
62          warning('Wrong test case name!')
63  end
64
65  para = struct(f1, q, f2, A, f3, b1, f4, b2);
```