Week 9 Exercises: State Machines and Markov Decision Processes

Bookmark this page

Week 9 Exercises: State Machines and Markov Decision Processes

(1.0 points possible)

For these exercises, you should read the notes on State Machines and Markov Decision Processes.

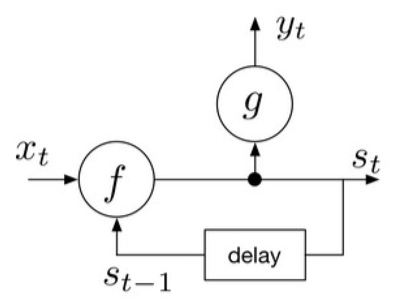
1) State Machines

State machines are a powerful and pervasive concept in information processing, playing important roles in computer science, signal processing, and control. Fundamentally, you can think of a state machine as a module that maps a sequence of input values $x_1, \ldots, x_T$ (they could be numbers, words, graphs, or anything) into a sequence of output values (also of any type) $y_1, \ldots, y_T$.

In the simplest case, we might have a simple functional dependence, so that $y_t = h(x_t)$, but more generally, we are interested in the case where y_t might potentially depend on all previous input values, so $y_t = h(x_1, \ldots, x_t)$. Generally, describing a functional dependency like this is very difficult, because the function h seems to need a different number of arguments on each time step. However, we can describe a very large class of h functions compactly, as a recursive combination of two functions, f and g, with an intermediate *state* value capturing the information that flows between applications of these functions. Mathematically, we have

$$y_t = g(s_t)$$
$$s_t = f(s_{t-1}, x_t)$$
$$s_0 = 0$$

Here is a figure illustrating the process: you can think of it as a circuit with the state fed back into f, but with a one-step delay.



If f and g are linear functions, then this is an LTI (linear time-invariant) system, a popular model in control systems analysis. In general, an LTI system can depend on a finite number of previous input values and a finite number of previous output values; we can model that dependence by allowing s_t to be a vector, storing as many of these values as we'd like.

If x_t, y_t, and s_t are all elements of finite sets, then this describes an FSM (finite-state machine), a popular model in computer science and some kinds of discrete control.

For each of the following state machines, provide the output sequence $[y_1, y_2, \ldots, y_T]$ given the input sequence $[x_1, x_2, \ldots, x_T]$:

1A)

```
s_0 = 0
f(s, x_i) = max(s, x_i)
g(s) = s * 2
Input: [0, 1, 2, 1]
```

Enter a Python list of four numbers: [0, 2, 4, 4]

Submit View Answer 100.00%

You have infinitely many submissions remaining.

1B)

```
s_0 = (0, 0)
f(s, x_i) = (s[0] + x_i, s[1] + 1)
g(s) = s[0] / s[1]
Input: [0, 1, 2, 1]
```

Enter a Python list of four numbers: [0, 0.5, 1, 1]

Submit View Answer 100.00%

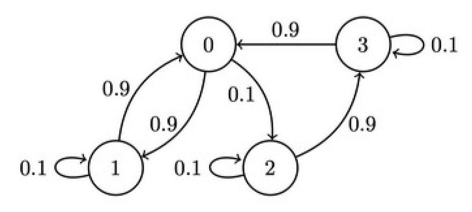
You have infinitely many submissions remaining.

2) Tiny Policy Evaluation

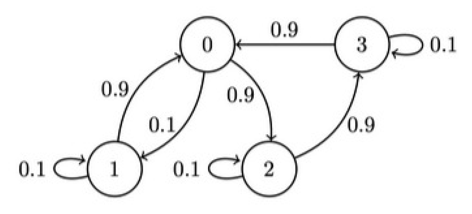
Consider an MDP with states (0, 1, 2, 3) and actions ('b', 'c'). The reward function is:

$$R(s,a) = \begin{cases} 1 & \text{if } s = 1 \\ 2 & \text{if } s = 3 \\ 0 & \text{otherwise} \end{cases}$$

You get the reward associated with a state on the step when you exit that state. The transition function for each action is below, where $T[i, x, j]$ is the conditional probability $P(s_{t+1} = j | a = x, s_t = i)$. The state transition diagrams are given as well, but you may find it easier to calculate the state values using the transition matrices.



$$T(s_t, \text{'b'}, s_{t+1}) = \begin{bmatrix} 0.0 & 0.9 & 0.1 & 0.0 \\ 0.9 & 0.1 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.1 & 0.9 \\ 0.9 & 0.0 & 0.0 & 0.1 \end{bmatrix}$$



$$T(s_t, \text{'c'}, s_{t+1}) = \begin{bmatrix} 0.0 & 0.1 & 0.9 & 0.0 \\ 0.9 & 0.1 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.1 & 0.9 \\ 0.9 & 0.0 & 0.0 & 0.1 \end{bmatrix}$$

Note that the **only** effect of the action is to change the transition probability from state 0 (the first row of the transition matrix: rows correspond to the input states, and columns correspond to the output states).

We would like to find the value function associated with the policy that **always** chooses action 'c'.

2A) What are the horizon 0 undiscounted values of the states under this policy?

Enter a Python list of four numbers: [0, 0, 0, 0]

Submit View Answer 100.00%

You have infinitely many submissions remaining.

2B) For horizon 1?

Enter a Python list of four numbers: [0, 1, 0, 2]

Submit View Answer 100.00%

You have infinitely many submissions remaining.

2C) For horizon 2?

Enter a Python list of four numbers: [0.1, 1.1, 1.8, 2.2]

Submit View Answer 100.00%

You have infinitely many submissions remaining.