Question 0, Homework 2, CME241

Link to code: https://github.com/nabilah13/RL-book/tree/master/assignment2

We set up the MDP Bellman Optimality Equation and see that we can focus on just the immediate reward:

$$V^*(s) = \max_{a \in A}(R(s,a) + 0.0 \sum_{s' \in S} P(s,a,s')V^*(s')$$

$$V^*(s) = \max_{a \in A}(R(s,a))$$

We know the distribution of the reward (negative of the cost) given a particular action is $-e^{as'}$ where s' has a normal distribution:

$$R(s,a) = -\int_{s'} e^{as'} \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(s'-s)^2}{2\sigma^2}} ds'$$

$$R(s,a) = -\mathrm{E}_{s'}(e^{as'})$$

We can use the the moment generating function for a normal distribution:

$$\mathrm{E}_{s'}(e^{as'}) = e^{sa + \frac{1}{2}\sigma^2 a^2}$$

We want to minimize this quantity with respect to a, and we can just minimize the numerator:

$$\frac{\mathrm{d}}{\mathrm{d}a}(sa + \frac{1}{2}\sigma^2 a^2) = 0$$

$$s + \sigma^2 a = 0$$

$$a = -\frac{s}{\sigma^2}$$

This represents our optimal action for a given state s. The optimal reward is then:

$$R^*(s,a) = -e^{s(-\frac{s}{\sigma^2}) + \frac{1}{2}\sigma^2(-\frac{s}{\sigma^2})^2}$$

$$R^*(s,a) = -e^{\frac{-s^2}{\sigma^2} + \frac{1}{2}\frac{s^2}{\sigma^2}}$$

$$R^*(s,a) = -e^{\frac{-1}{2}\frac{s^2}{\sigma^2}}$$

We calculate $q_1(\cdot, a_1)$:

$$q_1(\cdot, a_1) = \begin{bmatrix} 8.0 \\ 1.0 \\ 0.0 \end{bmatrix} + \begin{bmatrix} 0.2 & 0.6 & 0.2 \\ 0.3 & 0.3 & 0.4 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 10.0 \\ 1.0 \\ 0.0 \end{bmatrix} = \begin{bmatrix} 10.6 \\ 4.3 \\ 0.0 \end{bmatrix}$$

Now for $q_1(\cdot, a_2)$:

$$q_1(\cdot, a_2) = \begin{bmatrix} 10.0 \\ -1.0 \\ 0.0 \end{bmatrix} + \begin{bmatrix} 0.1 & 0.2 & 0.7 \\ 0.5 & 0.3 & 0.2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 10.0 \\ 1.0 \\ 0.0 \end{bmatrix} = \begin{bmatrix} 11.2 \\ 4.3 \\ 0.0 \end{bmatrix}$$

Now we do the policy improvement step. For s_1 it is clear that a_2 is better, while for s_2 we are indifferent. So we'll just set choosing a_2 for both states as our policy and do our next policy evaluation step:

$$q_2(\cdot, a_1) = \begin{bmatrix} 8.0 \\ 1.0 \\ 0.0 \end{bmatrix} + \begin{bmatrix} 0.2 & 0.6 & 0.2 \\ 0.3 & 0.3 & 0.4 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 11.2 \\ 4.3 \\ 0.0 \end{bmatrix} = \begin{bmatrix} 12.82 \\ 5.65 \\ 0.0 \end{bmatrix}$$

Now for $q_2(\cdot, a_2)$:

$$q_2(\cdot, a_2) = \begin{bmatrix} 10.0 \\ -1.0 \\ 0.0 \end{bmatrix} + \begin{bmatrix} 0.1 & 0.2 & 0.7 \\ 0.5 & 0.3 & 0.2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 11.2 \\ 4.3 \\ 0.0 \end{bmatrix} = \begin{bmatrix} 11.98 \\ 5.89 \\ 0.0 \end{bmatrix}$$

Now we can do a policy improvement step. The best policy is to choose a_1 when in state s_1 and a_2 when in state s_2.

We now argue that this policy will never change. Let v_i be the ith component of the value function vector. We can see that v_1 for a subsequent time step will be better or equivalent under policy a_1 as compared to a_2 so long as using the v_i from the previous time step:

$$8.0 + 0.2v_1 + 0.6v_2 \geq 10.0 + 0.1v_1 + 0.2v_2$$

$$0.1v_1 + 0.4v_2 \geq 2.0 \tag{1}$$

We notice that the value vector from time step $t = 2$ where $v_1 = 12.82$ and $v_2 = 5.89$ satisfies the above inequality. We know from slides 10-11 from the Chapter 5 lecture that by combining the Policy Improvement Theorem and monotonicity of the Bellman Policy Operator, the Value Iteration algorithm provides non-decreasing tower of value function vectors:

$$v_{i,t+1} \geq v_{i,t}$$

So then given that inequality (1) is currently satisfied, it will always be satisfied. Thus, we will not change our policy when in state s_1 for any future time step in the Value Iteration algorithm.

We follow a similar argument to show that we will not change our policy for state s_2, as a_2 is superior so long as:

$$-1.0 + 0.5v_1 + 0.3v_2 \geq 1.0 + 0.3v_1 + 0.3v_2$$

$$v_1 \geq 10.0 \tag{2}$$

Inequality (2) is clearly satisfied by the value vector at $t = 2$. We know this inequality will thus always be satisfied by repeating the earlier argument, so the policy when in state s_2 will not change in future time steps.

Code: https://github.com/nabilah13/RL-book/blob/master/assignment2/a2p3.py

The state space is the n jobs you can accept plus the state of being unemployed (state 0). We can map these to integers and define the state space as the following set:

$$S = \{0, 1, 2, \ldots, n\}$$

The action space is whether or not you accept a particular job when offered. So there are two elements in this space for no/yes:

$$A = \{0, 1\}$$

The transition function can be broken into a few cases. Ahead, $1_{a_i=1}$ represents an indicator function which takes value one when the prescribed action for job i is to accept.
When you are unemployed, the probability of remaining unemployed

$$\mathrm{P}(s_0, s_0) = 1 - \sum_{i=1}^{n} p_i * 1_{a_i=1}$$

When you are unemployed, the probability of obtaining job i:

$$\mathrm{P}(s_0, s_i) = p_i * 1_{a_i=1}$$

When you are employed in job i, the probability of staying in your job:

$$\mathrm{P}(s_i, s_i) = 1 - \alpha$$

When you are employed in job i, the probability of switching to job j:

$$\mathrm{P}(s_i, s_j) = \alpha * p_i * 1_{a_j=1}$$

When you are employed in job i, the probability of becoming unemployed:

$$\mathrm{P}(s_i, s_j) = \alpha * (1 - \sum_{i=1}^{n} p_i * 1_{a_i=1})$$

The reward function is the utility of the state that you are in:

$$R(s_i) = \log(w_i)$$

The Bellman Optimality Equation(s):

$$V(s_0) = \log(w_0) + \gamma((1 - \sum_{i=1}^{n}(p_i * 1_{a_i=1})) * V(s_0) + \sum_{i=1}^{n} p_i * 1_{a_i=1} * V(s_i))$$

$$V(s_0) = \log(w_0) + \gamma(V(s_0) + \left(\sum_{i=1}^{n} p_i * 1_{a_i=1} * (V(s_i) - V(s_0))\right))$$

$$V(s_i) = \log(w_i) + \gamma((1-\alpha)V(s_i) + \alpha \sum_{j=1}^{n} p_j * 1_{a_j=1} * V(s_j) + \alpha(1 - \sum_{j=1}^{n} p_j * 1_{a_j=1})V(s_0))$$

$$V(s_i) = \log(w_i) + \gamma((1-\alpha)V(s_i) + \alpha(V(s_0) + \left(\sum_{j=1}^{n} p_j * 1_{a_j=1} * (V(s_j) - V(s_0))\right)))$$