Lecture 7 - *RL Algorithms*

OBJECTIVE: In this lecture, we introduce some of the most popular model free RL algorithms, including the celebrated Q-learning and TD-lambda algorithms.

◇ TD(0)

The first algorithm we consider is TD(0). This algorithm is obtained by approximating the fixed point:

$$V(i) = \sum_{j=1}^{n} p(j|i, d(i)) \left[r(i, d(i), j) + \gamma V(j) \right]$$

with a standard SA:

$$V(i) = V(i) + \alpha \left(r(i, d(i), j^\star) + \gamma V(j^\star) - V(i) \right)$$

where $j^\star$ is a sample from $p(j|i, d(i))$ in model based RL. In model free RL, there is no need to know the transition model as we are simply learning from experience (trials).

The pseudo-code for this algorithm is as follows:

1. Initialize $V(\cdot)$ arbitrarily and let π be the input policy.
2. Repeat for each simulation trial:
   (a) Choose initial state i.
   (b) Repeat for each step in the simulation:
      i. a = action given by $\pi(i)$.
      ii. Take action a and observe the new state j
      iii. Observe reward $r(i, a, j)$
      iv. $V(i) = V(i) + \alpha \left(r(i, a, j) + \gamma V(j) - V(i) \right)$
      v. $i = j$

There is freedom in the above algorithm on how to choose (and improve) the policy π. We could, for example, choose to be greedy.

Before discussing this issue further, it is convenient to introduce Q-functions.

◇ Q-FUNCTIONS

Recall that to make optimal decisions, we need to solve

$$V^\star(i) = \max_a \sum_j p(j|i,a)\left[r(i,a,j) + \gamma V^\star(j)\right]$$

Alternatively, let us introduce the Q-function (matrix in the discrete setting):

$$Q^\star(i,a) \triangleq \sum_j p(j|i,a)\left[r(i,a,j) + \gamma V^\star(j)\right]$$

Then we can choose optimal decisions by finding:

$$V^\star(i) = \max_a Q^\star(i,a) = Q(i,a^\star)$$

Substituting, we get our Q-fixed point:

$$Q^\star(i,a) = \sum_j p(j|i,a)\left[r(i,a,j) + \gamma \max_{a'} Q^\star(i,a')\right]$$

Or, equivalently in operator notation:

$$Q^\star = FQ^\star$$

Again, we can use our knowledge of stochastic approximation to quickly derive an update rule for the Q-function:

$\star$

◇ Q-LEARNING

With our SA update and an ϵ-greedy policy (i.e. a policy that chooses the best action with probability ϵ and any other action randomly with probability $1-\epsilon$), we can easily produce an algorithm:

1. Initialize $Q(\cdot,\cdot)$ arbitrarily.
2. Repeat for each simulation trial:
   (a) Choose initial state i.
   (b) Repeat for each step in the simulation:
      i. Choose a in state i using policy derived from $Q(i,\cdot)$
      ii. Take action a and observe the new state j
      iii. Observe reward $r = r(i,a,j)$
      iv. $Q(i,a) = Q(i,a) + \alpha \left[r + \gamma \max_{a'} Q(j,a') - Q(i,a)\right]$
      v. $i = j$

◇ SARSA

Sarsa is an *on-policy* variant of Q-learning. The algorithm is as follows:

1. Initialize $Q(\cdot,\cdot)$ arbitrarily.
2. Repeat for each simulation trial:
   (a) Choose initial state i.
   (b) Choose a in state i using $Q(i,\cdot)$
   (c) Repeat for each step in the simulation:
      i. Take action a and observe the new state j
      ii. Observe reward $r = r(i,a,j)$
      iii. Choose a' in state j using $Q(j,\cdot)$
      iv. $Q(i,a) = Q(i,a) + \alpha \left[r + \gamma Q(j',a') - Q(i,a)\right]$
      v. $i = j$ and $a = a'$

Q-learning tends to be more "risk taking" than sarsa as illustrated by the cliff walking example in Sutton and Barto's book.

◇ ACTOR-CRITIC METHODS

Actor-critic methods are temporal difference (TD) methods, where an actor chooses a policy and a critic applies stochastic approximation on the value function to criticize the policy. The criticism is usually the following scalar:

$$\delta = r(i, a, j) + \gamma V(j) - V(i)$$

That is, the actor chooses a in state i and the critic computes δ to provide feedback to the author.

The author's policy can be the following *softmax* parametrization:

$$\pi(a|i) = \frac{e^{f(a,i)}}{\sum_b e^{f(b,i)}}$$

where $f(\cdot, \cdot)$ is a preference function.

Then, given a scalar parameter β, the preference function can be updated as follows:

$$f(a, i) = f(a, i) + \beta\delta_t$$

Hence, if δ is positive the preference for taking the action in that state also increases.

◇ TD(λ)

In TD(λ) one considers algorithms that use several steps ahead of the reward recursion. So far, we have used

$$R_t^{(1)} = r(\mathbf{x}_t, \mathbf{a}_t, \mathbf{a}_{t+1}) + \gamma V_t(\mathbf{x}_{t+1})$$

But we could also use:

$$R_t^{(2)} = r(\mathbf{x}_t, \mathbf{a}_t, \mathbf{x}_{t+1}) + \gamma r(\mathbf{x}_{t+1}, \mathbf{a}_{t+1}, \mathbf{x}_{t+2}) + \gamma^2 V_t(\mathbf{x}_{t+2})$$

or higher order expansions. We could even use combinations such as the average of the two-step and four-step returns:

$$R_t = \frac{1}{2}R_t^{(2)} + \frac{1}{2}R_t^{(4)}$$

In general, TD(λ) uses

$$R_t^{\lambda} = (1-\lambda)\sum_k \lambda^{k-1} R_t^{(k)}$$

where $\lambda \in [0, 1]$.

The TD(λ) algorithm is presented in great detail in the book of Sutton and Barto.

◇ RL WITH FUNCTION APPROXIMATION

When the value and Q-functions can be continuous and parameterized. For example we could use neural nets, ridge regression or gaussian processes to represent these functions. The SA updates are standard:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha_t[r_{t+1} + \gamma V(\mathbf{x}_{t+1}, \boldsymbol{\theta}_t) - V(\mathbf{x}_t, \boldsymbol{\theta}_t)]\frac{\partial V(\mathbf{x}_t, \theta)}{\partial \theta}$$

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha_t[r_{t+1} + \gamma \max_{a'} Q(\mathbf{x}_{t+1}, a', \boldsymbol{\theta}_t) - Q(\mathbf{x}_t, a, \boldsymbol{\theta}_t)]\frac{\partial Q(\mathbf{x}_t, a, \theta)}{\partial \theta}$$

It remains to show some examples of this function approximation and present POMDPS, direct policy methods and hierarchical RL.