CAS CS 440
Lec 33

Policy Learning V: Reinforcement Learning IV

1. Policy Search
    a. Core idea:
        i. Modify the policy until performance stops improving
    b. What kind of policies?
        i. Parameterized:
            1. Only useful if policy uses (significantly) less memory than the # of states
    c. For example:

$$\pi(s) = \underset{a}{\operatorname{argmax}} \hat{Q}_\theta(s, a)$$

    d. When policy relies solely on Q-functions:
        i. Policy Search → learn Q functions
        ii. Different than Q-learning
            1. Q-learning → goal is to learn Q-function that is “close enough” to Q*
            2. policy search → goal is to find Q-functions that are “good enough”
            3. These goals might not produce the same thing!
    e. Problem:
        i. Policy is discontinuous (i.e. not a smooth function)
    f. Why is discontinuity a problem?
        i. Some small change in θ can cause drastic changes is policy
        ii. Bad for gradients!
    g. Idea: combine discontinuous estimates into a continuous Prob:

$$\pi(s, a) = \frac{e^{\hat{Q}_\theta(s, a)}}{\sum_{a'} e^{\hat{Q}_\theta(s, a')}}$$

    h. This is called the softmax function
        i. A form of normalizing a vector → produces probabilities
    i. Why softmax?
        i. Almost deterministic when one action is vastly better than others
        ii. Differentiable!

j. One other nice property:

$$\log\left(\frac{e^{\hat{Q}_\theta(s,a)}}{\sum_{a'} e^{\hat{Q}_\theta(s,a')}}\right)$$
$$= \log\left(\frac{e^{\hat{Q}_\theta(s,a)}}{C}\right)$$
$$= \hat{Q}_\theta(s,a) - \log(C)$$

k. Why is this important?

i. Remember, we want policy to optimize

$$\mathbb{E}\left[R(\tau)\right] = \mathbb{E}\left[\sum_t \gamma^t R(s_t)\right]$$

ii. How can we represent $\mathbb{E}$?

$$\mathbb{E}\left[f(x)\right] = \sum_x \Pr[x] f(x)$$

iii. Looks kinda similar!

$$\mathbb{E}\left[\sum_t \gamma^t R(s_t)\right] = \sum_t \pi_\theta(s_t)\gamma^t R(s_t)$$

l. We need to differentiate

$$\frac{\partial}{\partial\theta_i}\mathbb{E}\left[\sum_t \gamma^t R(s_t)\right] = \frac{\partial}{\partial\theta_i}\sum_t \pi_\theta(s_t)\gamma^t R(s_t)$$
$$= \sum_t \frac{\partial}{\partial\theta_i}\pi_\theta(s_t)\gamma^t R(s_t)$$
$$= \sum_t \gamma^t R(s_t)\frac{\partial}{\partial\theta_i}\pi_\theta(s_t)$$
$$= \sum_t \gamma^t R(s_t)\pi_\theta(s_t)\frac{\frac{\partial}{\partial\theta_i}\pi_\theta(s_t)}{\pi_\theta(s_t)}$$
$$= \sum_t \gamma^t R(s_t)\pi_\theta(s_t)\frac{\partial}{\partial\theta_i}\log\left(\pi_\theta(s_t)\right)$$
$$= \mathbb{E}\left[\sum_t \gamma^t R(s_t)\frac{\partial}{\partial\theta_i}\log\left(\pi_\theta(s_t)\right)\right]$$

m. Woah!

$$\nabla_\theta \mathbb{E}\left[\sum_t \gamma^t R(s_t)\right] = \mathbb{E}\left[\sum_t \gamma^t R(s_t)\, \nabla_\theta \log(\pi_\theta(s_t))\right]$$

n. Our gradient = gradient of policy scaled by how good the choices were
o. Can take sample average to approximate expectation!
p. Turns out there is a little more we can do here!

$$\nabla_\theta \mathbb{E}\left[\sum_t \gamma^t R(s_t)\right] = \mathbb{E}\left[\sum_t G_t\, \nabla_\theta \log(\pi_\theta(s_t))\right]$$

i. G_t = value of trajectory from that point on

2. Policy Search with REINFORCE
   a. Transforms search into a Monte-Carlo procedure
      i. Samples one trajectory (i.e. plays one game)
      ii. Records trajectory
   b. Updates policy w policy gradient (offline i.e. in between games)

function REINFORCE
  Initialise θ arbitrarily
  for each episode $\{s_1, a_1, r_2, \ldots, s_{T-1}, a_{T-1}, r_T\} \sim \pi_\theta$ **do**
    for $t = 1$ to $T - 1$ **do**
      $\theta \leftarrow \theta + \alpha \nabla_\theta \log \pi_\theta(s_t, a_t) v_t$
    end for
  end for
  return θ
end function

   c. Repeat!
3. Can We do Better?
   a. Policy Gradient:

$$\mathbb{E}\left[\sum_t G_t\, \nabla_\theta \log\left(\pi_\theta(s_t)\right)\right]$$

   b. What if G_t was its own function-approx G_θ (could be $Q_{\theta'}(s_t,\ a_t)$)
   c. Already have π_θ
4. Actor-Critic RL
   a. The "critic" = a function approximation for the "value" function

    i. Could be Q-value, utility value

b. The “actor” = a function approximation for the policy
c. Critic still obeys bellman equation, can use TD learning

$$\theta_i' \leftarrow \theta_i' + \alpha\left(R(s) + \gamma\hat{Q}_{\theta'}(s', a') - \hat{Q}_{\theta'}(s, a)\right)\frac{\partial\hat{Q}_{\theta'}(s, a)}{\partial\theta_i'}$$

d. Actor gradients we just derived!

$$\theta_j \leftarrow \theta_j + \eta G_t \nabla_\theta \log\left(\pi_\theta\left(s_t\right)\right)$$