Reinforcement Learning David Silver - Lecture 7 Notes: Policy Gradient

Name: Eli Andrew

- **Policy-based RL**
  - Previously, we were parameterizing V and Q and generating a policy directly from them (using ϵ-greedy for example)
  - Now, we directly parameterize the policy

$$\pi_\theta(s, a) = P[a|s, \theta]$$

  - The motivation here is, once again, to be able to scale efficiently to many states
  - Advantages:
    * Better convergence properties
    * (best reason) Effective in high-dimensional or continuous action spaces
    * Can learn stochastic policies
  - Disadvantages:
    * Typically converge to a local rather than global optimum
    * Evaluating a policy is typically inefficient and high variance
  - Why would you want a stochastic policy?
    * Games like Rock-paper-scissors
    * State aliasing problems where the agent can't differentiate between certain states
    * The state aliasing problem can occur because of a partially observed enviornment (which is equivalent to not having the correct features to represent the enviornment)
    * If we have state aliasing with a determinisic policy, then all aliased states (different states that appear the same) will have to have the same action
- **Policy Objective Functions**
  - Goal: given policy $\pi_\theta(s, a)$ with parameters θ, find best θ
  - How to measure quality of policy $\pi_\theta(s, a)$
    * Episodic environments: use **start value**

$$J_1(\theta) = V^{\pi_\theta}(s_1) = E_{\pi_\theta}[v_1]$$

    * In continuing environments: use **average reward per time-step**

$$J_{avR}(\theta) = \sum_s d^{\pi_\theta}(s) \sum_a \pi_\theta(s, a) R_s^a$$

        * Or **average value**

$$J_{avV}(\theta) = \sum_s d^{\pi_\theta}(s) V^{\pi_\theta}(s)$$

        * Where $d^{\pi_\theta}(s)$ is stationary distribution of Markov chain for π_θ

- **Policy Optimization**
    - Find θ that minimises $J(\theta)$
    - Some approaches don't use gradient
        * Hill climbing
        * Simplex / amoeba / Nelder Mead
        * Genetic algorithms
    - Greater efficiency often possible using gradient
        * Gradient descent
        * Conjugate gradient
        * Quasi-newton
- **Policy Gradient**
    - Let $J(\theta)$ be any policy objective function
    - Policy gradient algorithms search for *local* maximum in $J(\theta)$ by ascending the gradient of the policy, w.r.t parameters θ

$$\Delta\theta = \alpha \nabla_\theta J(\theta)$$

    - Where $\nabla_\theta J(\theta)$ is the **policy gradient** and α is step size parameter
- **Score Function**
    - Assume policy π_θ is differentiable everywhere we can take actions
    - Assume we know gradient $\nabla_\theta \pi_\theta(s, a)$
    - Likelihood ratios exploit the following identity:

$$\begin{aligned} \nabla_\theta \pi_\theta(s, a) &= \pi_\theta(s, a) \frac{\nabla_\theta \pi_\theta(s, a)}{\pi_\theta(s, a)} \\ &= \pi_\theta(s, a) \nabla_\theta \log \pi_\theta(s, a) \end{aligned}$$

    - Score Function is: $\nabla_\theta \log \pi_\theta(s, a)$
    - Using this form allows us to take expectations much easier
- **Softmax Policy**

– Weight actions using linear combination of features $\phi(s,a)^\top\theta$

– Probability of action is proportional to exponentiated weight:

$$\pi_\theta(s,a) \propto \exp(\phi(s,a)^\top\theta)$$

– Score function is then:

$$\nabla_\theta \log \pi_\theta(s,a) = \phi(s,a) - E_{\pi_\theta}[\phi(s,.)]$$

– In other words the score function is the difference between the feature of the action we actually took and the average feature: this says how much more of this function do I have than usual

– Since adjustments are then made based on this difference, it means that if I feature contributed more than usual then it will be updated more in the direction of the reward we received

- **Gaussian Policy**
  - Most common policy for continuous action spaces
  - Mean is linear combination of state features

$$\mu(s) = \phi(s)^\top\theta$$

  - Variance may be fixed σ^2 or can be parameterized
  - Policy is Gaussian:

$$a \sim \mathcal{N}(\mu(s), \sigma^2)$$

  - Score function is:

$$\nabla_\theta \log \pi_\theta(s,a) = \frac{(a-\mu(s))\phi(s)}{\sigma^2}$$

- **One step MDP**
  - Use example of a simple one-step MDP where you start in state $s \sim d(s)$ and terminate after one step with reward $r = R_{s,a}$
  - Likelihood ratios help with calculating Policy gradient:

$$\begin{aligned} J(\theta) &= E[r] \\ &= \sum_{s\in S} d(s) \sum_{a\in A} \pi_\theta(s,a) R_{s,a} \\ \nabla_\theta J(\theta) &= \sum_{s\in S} d(s) \sum_{a\in A} \pi_\theta(s,a) \nabla_\theta \log(\pi_\theta(s,a)) R_{s,a} \\ &= E[\nabla_\theta \log(\pi_\theta(s,a)) r] \end{aligned}$$

- **Policy Gradient Theorem**
  - Generalizes liklihood ratio to multi-step MDPs
  - Replaces instantaneous reward r with long term value $Q^{\pi}(s,a)$
  - Applies to start state objective, average reward, and average value objective
  - Theorem: for any differentiable policy $\pi_\theta(s,a)$ for any of the applicable objective functions (stated above) the policy gradient is:

$$\nabla_\theta J(\theta) = E_{\pi_\theta}[\nabla_\theta \log \pi_\theta(s,a) Q^{\pi_\theta}(s,a)]$$

- **Monte Carlo Policy Gradient**
  - Use v_t as an unbiased sample of $Q^{\pi_\theta}(s_t, a_t)$
  - For each episode: select $s_1, a_1, r_2, \ldots, s_{T-1}, a_{T-1}, r_T \sim \pi_\theta$
  - For each time step of the epsiode make update: $\theta \leftarrow \theta + \alpha \nabla_\theta \log \pi_\theta(s_t, a_t) v_t$
  - Where v_t is cummulated rewards from that time step onwards

- **Reducing variance with a Critic**
  - Monte-Carlo policy gradient has high variance
  - Use a **critic** to estimate the state-action value function:

$$Q_w(s,a) \approx Q^{\pi_\theta}(s,a)$$

  - Actor-critic algorithms maintain two sets of parameters
    * Critic: Updates action-value function parameters w
    * Actor: Updates policy parameters θ, in direction suggested by critic
  - Actor-critic algorithms follow an *approximate* policy gradient

$$\nabla_\theta J(\theta) \approx E_{\pi_\theta}[\nabla_\theta \log \pi_\theta(s,a) Q_w(s,a)]$$
$$\Delta\theta = \alpha \nabla_\theta \log \pi_\theta(s,a) Q_w(s,a)$$

- **Action-Value Actor-Critic**
  - Using linear value function approx. $Q_w(s,a) = \phi(s,a)^\top w$
    * Critic: Updates w by linear TD(0)
    * Actor: Updates θ by policy gradient
    * Intialize s, θ
    * Sample $a \sim \pi_\theta$

* For each step:

$$\text{Sample reward } r = R_s^a;\ \text{Sample transition } s' \sim P_s^a$$
$$\text{Sample action } a' \sim \pi_\theta(s', a')$$
$$\delta = r + \gamma Q_w(s', a') - Q_w(s, a)$$
$$\theta = \theta + \alpha \nabla_\theta \log \pi_\theta(s, a) Q_w(s, a)$$
$$w \leftarrow w + \beta \delta \phi(s, a)$$
$$a \leftarrow a', s \leftarrow s'$$

- In summary:
  * Actor picks the actions using some policy
  * Critic evaluates and says whether the actions are good or bad
  * Actor moves its policy in the direction suggested by the critic

- **Reducing Variance Using a Baseline**
  - Subtract a baseline function $B(s)$ from the policy gradient
  - This reduces variance without changing expectation:

$$\begin{aligned} E_{\pi_\theta}[\nabla_\theta \log \pi_\theta(s, a) B(s)] &= \sum_{s \in S} d^{\pi_\theta}(s) \sum_a \nabla_\theta \pi_\theta(s, a) B(s) \\ &= \sum_{s \in S} d^{\pi_\theta} B(s) \nabla_\theta \sum_{a \in A} \pi_\theta(s, a) \\ &= 0 \end{aligned}$$

  - State value function is good baseline: $B(s) = V^{\pi_\theta}(s)$
  - Re-write policy gradient using advantage function: $A^{\pi_\theta}(s, a)$:

$$A^{\pi_\theta}(s, a) = Q^{\pi_\theta}(s, a) - V^{\pi_\theta}(s)$$
$$\nabla_\theta J(\theta) = E_{\pi_\theta}[\nabla_\theta \log \pi_\theta(s, a) A^{\pi_\theta}]$$

  - Advantage function tells us how much better it is than usual to take action a
  - $\nabla_\theta \log \pi_\theta(s, a)$ tells us how to adjust θ to acheive that action a with our policy π_θ

- **Estimating the Advantage Function**
  - TD-error is a sample of the advantage function
  - For true value function $V^{\pi_\theta}(s)$, the TD-error δ^{π_θ}

$$\delta^{\pi_\theta} = r + \gamma V^{\pi_\theta}(s') - V^{\pi_\theta}(s)$$

- is an unbiased estimate of the advantage function:

$$E[\delta^{\pi_\theta}|s,a] = E_{\pi_\theta}[r + \gamma V^{\pi_\theta}(s')|s,a] - V^{\pi_\theta}$$
$$= Q^{\pi_\theta}(s,a) - V^{\pi_\theta}(s) = A^{\pi_\theta}(s,a)$$

- So, we can use TD error to compute policy gradient:

$$\nabla_\theta J(\theta) = E_{\pi_\theta}[\nabla_\theta \log \pi_\theta(s,a)\delta^{\pi_\theta}]$$

- And in practice we can just estimate the TD error:

$$\delta_v = r + \gamma V_v(s') - V_v(s)$$

- This leads to only one set of parameters: v

- **Natural Policy Gradient**
  - Policies we've considered have all been stochastic
  - We've estimated our policy gradients by sampling our own noise
  - Taking expectation of gradient of our own noise can run into issues, especially as Gaussian gets narrower as your policy improves (noise hurts you more the better the policy gets)
  - Instead, start with a determinisitc policy
  - With the deterministic case where our noise is narrowed down to 0 we get an update:

$$\nabla_\theta^{nat}\pi_\theta(s,a) = G_\theta^{-1}\nabla_\theta\pi_\theta(s,a)$$

  - Where G_θ is Fischer Information matrix: $E_{\pi_\theta}[\nabla_\theta \log \pi_\theta(s,a)\nabla_\theta \log \pi_\theta(s,a)^\top]$
  - In practice, this performs much better than stochastic gradient policies especially in cases with continuous actions

- **Natural Actor-Critic**
  - Using compatible function approximation

$$\nabla_w A_w(s,a) = \nabla_\theta \log \pi_\theta(s,a)$$

  - Natural policy gradient becomes:

$$\nabla_\theta J(\theta) = E_{\pi_\theta}[\nabla_\theta \log \pi_\theta(s,a)A^{\pi_\theta}(s,a)]$$
$$= E_{\pi_\theta}[\nabla_\theta \log \pi_\theta(s,a)\nabla_\theta \log \pi_\theta(s,a)^\top w]$$
$$= G_\theta w \nabla_\theta^{nat} J(\theta) = w$$

- **Summary of Policy Gradient Algorithms**
  - The **policy gradient** has many equivalent forms
  - REINFORCE: v_t

$$\nabla_\theta J(\theta) = E_{\pi_\theta}[\nabla_\theta \log \pi_\theta(s,a) v_t]$$

  - Q Actor-Critic: $Q^w(s,a)$

$$\nabla_\theta J(\theta) = E_{\pi_\theta}[\nabla_\theta \log \pi_\theta(s,a) Q^w(s,a)]$$

  - Advantage Actor-Critic: $A^w(s,a)$

$$\nabla_\theta J(\theta) = E_{\pi_\theta}[\nabla_\theta \log \pi_\theta(s,a) A^w(s,a)]$$

  - TD Actor-Critic: δ

$$\nabla_\theta J(\theta) = E_{\pi_\theta}[\nabla_\theta \log \pi_\theta(s,a) \delta]$$

  - TD(λ) Actor-Critic: δe

$$\nabla_\theta J(\theta) = E_{\pi_\theta}[\nabla_\theta \log \pi_\theta(s,a) \delta e]$$

  - Natural Actor-Critic:

$$G_\theta^{-1} \nabla_\theta J(\theta) = w$$

  - Each form leads a stochastic gradient ascent algorithm
  - Critic uses **policy evaluation** (e.g. MC, TD, or other algorithms from before) to estimate $Q^\pi(s,a), A^\pi(s,a)$, or $V^\pi(s)$