Lecture 8: Policy Gradient

Chonggang Wang

Outline

- Policy Gradient RL
- Actor-Critic Methods
- Policy Gradient w/ Advantage Function

*materials are modified from David Silver's RL lecture notes

Outline

- Policy Gradient RL
- Actor-Critic Methods
- Policy Gradient w/ Advantage Function

Lecture 7 – Value Approximation

- Parameterized value function $\hat{v}(s, w)$ and action-value function $\hat{q}(s, a, w)$ using a parameter column vector w, to approximate true value functions $V_\pi(s)$ and $Q_\pi(s)$

$$\hat{v}(s, w) \approx V_\pi(s)$$
$$\hat{q}(s, a, w) \approx Q_\pi(s, a)$$

  - How to obtain $\hat{v}(s, w)$ and $\hat{q}(s, a, w)$
    - Stochastic Gradient Descent: $\Delta w = \alpha * (V_\pi(s) - \hat{v}(s, w)) * \nabla_w \hat{v}(s, w)$
    - Calculate the gradient $\nabla_w \hat{v}(s, w)$: Represent a state by a feature vector
      - Linear Approach: $\hat{v}(s, w) = x(s)^T w$
      - Non-Linear Approach: Neural Network (e.g., DQN)
- Use the action-value function to determine/select next action based on a pre-configured policy (e.g., greedy, ε-greedy)
- Question: Can we determine next action directly from a parameterized probability function which only depends on the states: Policy-based RL

$$\pi_\theta(s, a) = p(a|s, \theta)$$

Why Policy-based RL

- Advantages
  - Action probabilities change more smoothly → better convergence
  - Easy to learn stochastic policies -> better exploration
  - Avoid "argmax(q(s, a))" → Lower complexity and more effective in high-dimensional or continuous action spaces

- Disadvantages
  - Value-based approaches could be more efficient for a small number of states and actions
  - Hard to get unbiased estimates of policy gradient through sampling
  - Basic policy gradient approaches (e.g., REINFOCE as a Monte Carlo Method) introduce high variance and lead to a lower convergence speed

Policy Objective Functions

- Goal: given policy $\pi_\theta(s, a)$ with parameters θ, find best θ
- But how do we measure the quality of a policy π_θ?
- In episodic environments we can use the start value

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

- In continuing environments we can use the average value

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

- Or the average reward per time-step

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

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

Policy Optimization

- Policy based reinforcement learning is an optimisation problem
- Find θ that maximises $J(\theta)$

- Similar to the value based function approximation, we focus on gradient method
  - Gradient is a key to connect neural network with RL algorithms
- Other approaches are possible

Gradient Descent (recap)

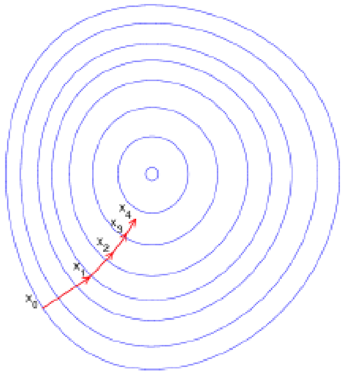
- Let $J(\mathbf{w})$ be a differentiable function of parameter vector $\mathbf{w}$, a column vector
- Define the *gradient* of $J(\mathbf{w})$ to be

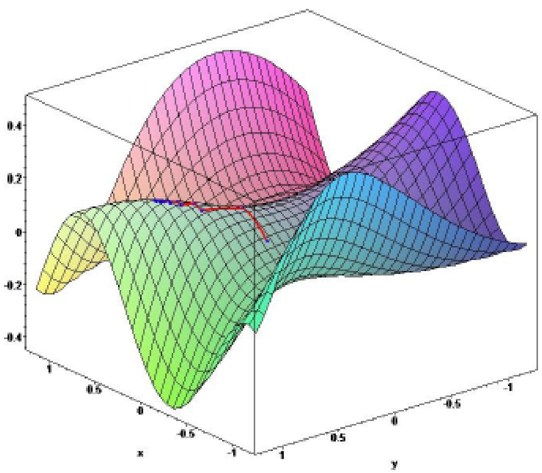
$$\nabla_{\mathbf{w}} J(\mathbf{w}) = \begin{pmatrix} \frac{\partial J(\mathbf{w})}{\partial \mathbf{w}_1} \\ \vdots \\ \frac{\partial J(\mathbf{w})}{\partial \mathbf{w}_n} \end{pmatrix}$$

Partial Derivatives with respect to **w**

- To find a local minimum of $J(\mathbf{w})$
- Adjust $\mathbf{w}$ in direction of -ve gradient to reduce *J*(w) (i.e., the Value Error (VE))

$$\Delta \mathbf{w} = -\frac{1}{2}\alpha \nabla_{\mathbf{w}} J(\mathbf{w})$$





Gradient Ascent

- $J(\theta)$ is the policy-related performance function to be maximized
- $J(\theta)$ is differentiable with respect to θ, which is a column parameter vector

$$\theta = (\theta_1, \dots, \theta_n)^T$$

- $\nabla_\theta J(\theta)$: The Gradient of $J(\theta)$ can be calculated

$$\nabla_\theta J(\theta) = \begin{pmatrix} \frac{\partial J(\theta)}{\theta_1} \\ \cdot \\ \cdot \\ \cdot \\ \frac{\partial J(\theta)}{\theta_n} \end{pmatrix}$$

Partial Derivatives with respect to θ

- **Gradient Ascent**
  - To find the optimal parameter vector which maximizes $J(\theta)$

$$\boldsymbol{\Delta\theta = \alpha * \nabla_\theta J(\theta)}$$

Score Function

- We now compute the policy gradient *analytically*
- Assume policy π_θ is differentiable whenever it is non-zero and we know the gradient $\underline{\nabla_\theta \pi_\theta(s, a)}$
- Likelihood ratios exploit the following identity

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

- The score function is $\nabla_\theta \ ln \ \pi_\theta(s, a)$

Example: Softmax Policy

- We will use a softmax policy as a running example
- Weight actions using linear combination of features $\phi(s, a)^\top \theta$
- Probability of action is proportional to exponentiated weight

$$\pi_\theta(s,a) = p(a|s,\theta) = \frac{e^{\phi(s,a)^T\theta}}{\sum_a e^{\phi(s,a)^T\theta}}$$

- The score function is

$$\nabla_\theta \; ln \; \pi_\theta(s, a) = \phi(s, a) - \mathbb{E}_{\pi_\theta}\left[\phi(s, \cdot)\right]$$

Example: Softmax Policy

- Proof:

$$\nabla\, ln\, \pi_\theta(s,a) = \nabla\, ln \frac{e^{\phi(s,a)^T\theta}}{\sum_a e^{\phi(s,a)^T\theta}} \quad \text{--Eq. (1)}$$

$$\nabla \ln\left(\frac{\boldsymbol{x}}{\boldsymbol{y}}\right) = \nabla \ln(\boldsymbol{x}) - \nabla \ln(\boldsymbol{y})$$

$$= \nabla\phi(s,a)^T\theta - \nabla\, ln\left(\sum_a e^{\phi(s,a)^T\theta}\right) \quad \text{--Eq. (2)}$$

$$\nabla \ln(\boldsymbol{g}(\boldsymbol{x})) = \frac{\nabla \boldsymbol{g}(\boldsymbol{x})}{\boldsymbol{g}(\boldsymbol{x})}$$

$$= \phi(s,a) - \frac{\sum_a \left(e^{\phi(s,a)^T\theta}\phi(s,a)\right)}{\sum_a e^{\phi(s,a)^T\theta}} \quad \text{--Eq. (3)}$$

$$= \phi(s,a) - \sum_a \frac{e^{\phi(s,a)^T\theta}}{\underline{\sum_a e^{\phi(s,a)^T\theta}}}\phi(s,a) \quad \text{--Eq. (4)}$$

$$= \phi(s,a) - \sum_a \underline{\pi_\theta(s,a)}\phi(s,a) \quad \text{--Eq. (5)}$$

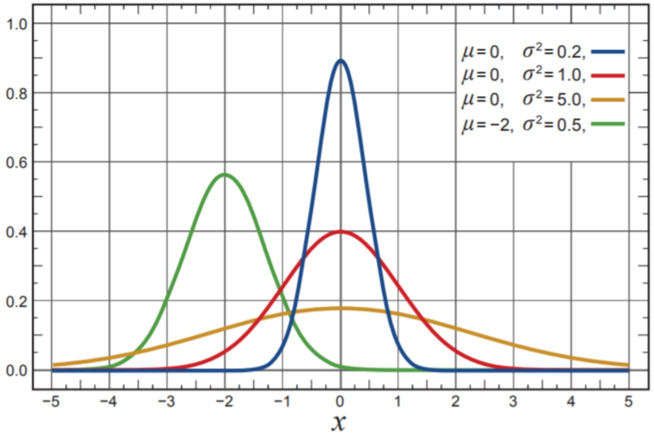
$$= \phi(s,a) - \mathbb{E}_{\pi_\theta}[\phi(s,\cdot)] \quad \text{--Eq. (6)}$$

Example: Gaussian Policy

- In continuous action spaces, a Gaussian policy is natural
- Mean is a linear combination of state features $\mu(s) = \phi(s)^{\top}\theta$
- Variance may be fixed σ^2, or can also parametrised
- Policy is Gaussian, $a \sim \mathcal{N}(\mu(s), \sigma^2)$
- The score function is

$$\pi_\theta(a|s,\theta) = \frac{1}{\sigma\sqrt{2\pi}} e^{\left(-\frac{(a-\mu(s,\theta))^2}{2\sigma^2}\right)}$$

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



One-Step MDP

- Consider a simple class of one-step MDPs
  - Starting in state $s \sim d(s)$
  - Terminating after one time-step with reward $r = \mathcal{R}_{s,a}$
- Use likelihood ratios to compute the policy gradient

$$\begin{aligned}
J(\theta) &= \mathbb{E}_{\pi_\theta}\left[r\right] \\
&= \sum_{s \in \mathcal{S}} d(s) \sum_{a \in \mathcal{A}} \pi_\theta(s,a) \mathcal{R}_{s,a} \\
\nabla_\theta J(\theta) &= \sum_{s \in \mathcal{S}} d(s) \sum_{a \in \mathcal{A}} \underline{\pi_\theta(s,a)} \underline{\nabla_\theta \; ln \, \pi_\theta(s,a) \mathcal{R}_{s,a}} \\
&= \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \; ln \, \pi_\theta(s,a) r\right]
\end{aligned}$$

Generalized MDP

- The policy gradient theorem generalises the likelihood ratio approach to multi-step MDPs
- Replaces instantaneous reward r with long-term value $Q^{\pi}(s, a)$
- Policy gradient theorem applies to start state objective, average reward and average value objective

For any differentiable policy $\pi_\theta(s, a)$,
for any of the policy objective functions $J = J_1, J_{avR},$ or $\frac{1}{1-\gamma} J_{avV}$,
the policy gradient is

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta} \left[\nabla_\theta \ln \pi_\theta(s, a) \; Q^{\pi_\theta}(s, a) \right]$$

Policy Gradient Theorem

Monte-Carlo Policy Gradient (REINFORCE)

- Update parameters by stochastic gradient ascent
- Using policy gradient theorem
- Using return G_t as an unbiased sample of $Q^{\pi_\theta}(s_t, a_t)$

High Variance

$$\Delta\theta_t = \alpha \underbrace{\nabla_\theta \ ln\ \pi_\theta(s_t, a_t)}_{\text{Policy Gradient}} G_t$$

```
function REINFORCE
    Initialise θ arbitrarily
    for each episode {s_1, a_1, r_2, ..., s_{T-1}, a_{T-1}, r_T} ~ π_θ do
        for t = 1 to T − 1 do
            θ ← θ + α ∇_θ ln π_θ(s_t, a_t) G_t
        end for
    end for
    return θ
end function
```

Outline

- Policy Gradient RL
- Actor-Critic Methods
- Policy Gradient w/ Advantage Function

*materials are modified from David Silver's RL lecture notes

Reducing Variance Using a Critic

- Monte-Carlo policy gradient still has high variance
- We use a critic to estimate the 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

$$\nabla_\theta J(\theta) \approx \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \ln \pi_\theta(s, a)\ Q_w(s, a)\right]$$
$$\Delta\theta = \alpha \nabla_\theta \ln \pi_\theta(s, a)\ Q_w(s, a)$$

Example: TD(0)-based Actor-Critic Algorithm

- Simple actor-critic algorithm based on action-value critic
- Using linear value fn approx. $Q_w(s,a) = \phi(s,a)^\top w$

  Critic Updates w by linear TD(0)

  Actor Updates θ by policy gradient

function QAC
    Initialise s, θ
    Sample $a \sim \pi_\theta$
    for each step **do**
        Sample reward $r = \mathcal{R}_s^a$; sample transition $s' \sim \mathcal{P}_{s,\cdot}^a$
        Sample action $a' \sim \pi_\theta(s', a')$
        $\delta = r + \gamma Q_w(s',a') - Q_w(s,a)$
        $\theta = \theta + \alpha \nabla_\theta \, ln \, \pi_\theta(s,a) Q_w(s,a)$
        $w \leftarrow w + \beta \delta \phi(s,a)$ ← $w = w + \beta * \delta * (\nabla_w Q_w(s,a))$
        $a \leftarrow a', s \leftarrow s'$
    end for
end function

Problem: Bias in Actor-Critic Algorithms

- Approximating the policy gradient introduces bias
- A biased policy gradient may not find the right solution

- Luckily, if we choose value function approximation carefully
- Then we can avoid introducing any bias

  i.e. We can still follow the *exact* policy gradient

Compatible Function Approximation Theorem*

If the following two conditions are satisfied:

Value function approximator is compatible to the policy

$$\nabla_w Q_w(s,a) = \nabla_\theta \ln \pi_\theta(s,a)$$

Value function parameters w minimise the mean-squared error

$$\varepsilon = \mathbb{E}_{\pi_\theta}\left[\left(Q^{\pi_\theta}(s,a) - Q_w(s,a)\right)^2\right]$$

$$\frac{\partial \varepsilon}{\partial w} = 0$$

Then the policy gradient is exact,

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \ln \pi_\theta(s,a)\ Q_w(s,a)\right]$$

*R. Sutton, et al. "Policy gradient methods for reinforcement learning with function approximation", 2000

Outline

- Policy Gradient RL
- Actor-Critic Methods
- Policy Gradient w/ Advantage Function

Reducing Variance Using a Baseline

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta\, ln\, \pi_\theta(s,a)\; Q^{\pi_\theta}(s,a)\right]$$ (Without a Baseline)

- We subtract a baseline function $B(s)$ from the policy gradient
- This can reduce variance, without changing expectation

$$\begin{aligned}\mathbb{E}_{\pi_\theta}\left[\nabla_\theta\, ln\, \pi_\theta(s,a)B(s)\right] &= \sum_{s\in\mathcal{S}} d^{\pi_\theta}(s) \sum_a \nabla_\theta \pi_\theta(s,a)B(s)\\ &= \sum_{s\in\mathcal{S}} d^{\pi_\theta} B(s)\nabla_\theta \sum_{a\in\mathcal{A}} \pi_\theta(s,a)\\ &= 0\end{aligned}$$

B(s) is action-independent

- A good baseline is the state value function $B(s) = V^{\pi_\theta}(s)$
- So we can rewrite the policy gradient using the advantage function $A^{\pi_\theta}(s,a)$

$$A^{\pi_\theta}(s,a) = Q^{\pi_\theta}(s,a) - V^{\pi_\theta}(s)$$

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta\, ln\, \pi_\theta(s,a)\; A^{\pi_\theta}(s,a)\right]$$ (Using a Baseline)

"Variance Reduction for Policy Gradient with Action-Dependent Factorized Baselines " - ICLR 2018

Estimating the Advantage Function

- The advantage function can significantly reduce variance of policy gradient
- So the critic should really estimate the advantage function

  For example, by estimating *both* $V^{\pi_\theta}(s)$ *and* $Q^{\pi_\theta}(s,a)$
- Using two function approximators and two parameter vectors,

$$
\begin{aligned}
V_v(s) &\approx V^{\pi_\theta}(s) \\
Q_w(s,a) &\approx Q^{\pi_\theta}(s,a) \\
A(s,a) &= Q_w(s,a) - V_v(s)
\end{aligned}
$$

- And updating *both* value functions by e.g. TD learning

Estimating the Advantage Function (cont.)

- For the 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

$$\begin{aligned}\mathbb{E}_{\pi_\theta}\left[\delta^{\pi_\theta}|s,a\right] &= \mathbb{E}_{\pi_\theta}\left[r + \gamma V^{\pi_\theta}(s')|s,a\right] - V^{\pi_\theta}(s) \\ &= Q^{\pi_\theta}(s,a) - V^{\pi_\theta}(s) \\ &= A^{\pi_\theta}(s,a)\end{aligned}$$

- So we can use the TD error to compute the policy gradient

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \, ln \, \pi_\theta(s,a) \, \delta^{\pi_\theta}\right]$$

- In practice we can use an approximate TD error

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

- This approach only requires one set of critic parameters v

Critic at Different Time-Scales

- Critic can estimate value function $V_v(s)$ from many targets at different time-scales

For MC, the target is the return v_t

$$\Delta V = \alpha(G_t - V_v(s))\boxed{\phi(s)}$$

linear approximation

For TD(0), the target is the TD target $r + \gamma V(s')$

$$\Delta V = \alpha(r + \gamma V(s') - V_v(s))\phi(s)$$

For forward-view TD(λ), the target is the λ-return v_t^λ

$$\Delta V = \alpha(G_t^\lambda - V_v(s))\phi(s)$$

For backward-view TD(λ), we use eligibility traces

$$\delta_t = r_{t+1} + \gamma V(s_{t+1}) - V(s_t)$$
$$e_t = \gamma \lambda e_{t-1} + \phi(s_t)$$
$$\Delta V = \alpha \delta_t e_t$$

Actor at Different Time-Scales

- The policy gradient can also be estimated at many time-scales

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \, ln \, \pi_\theta(s, a) \; A^{\pi_\theta}(s, a)\right]$$

- Monte-Carlo policy gradient uses error from complete return

$$\Delta\theta = \alpha(G_t - V_v(s_t))\nabla_\theta \; ln \, \pi_\theta(s_t, a_t)$$

- Actor-critic policy gradient uses the one-step TD error

$$\Delta\theta = \alpha(r + \gamma V_v(s_{t+1}) - V_v(s_t))\nabla_\theta \; ln \, \pi_\theta(s_t, a_t)$$

Policy Gradient with Eligibility Traces

- Just like forward-view TD(λ), we can mix over time-scales

$$\Delta\theta = \alpha(G_t^{\lambda} - V_v(s_t))\nabla_\theta \; ln \; \pi_\theta(s_t, a_t)$$

where $G_t^{\lambda} - V_v(s_t)$ is a biased estimate of advantage fn

- Like backward-view TD(λ), we can also use eligibility traces
  By equivalence with TD(λ), substituting $\phi(s) = \nabla_\theta \; ln \, \pi_\theta(s, a)$

$$\begin{aligned} \delta &= r_{t+1} + \gamma V_v(s_{t+1}) - V_v(s_t) \\ e_{t+1} &= \lambda e_t + \nabla_\theta \; ln \; \pi_\theta(s, a) \\ \Delta\theta &= \alpha\delta e_t \end{aligned}$$

- This update can be applied online, to incomplete sequences

Summary of Policy Gradient Algorithms

- The policy gradient has many equivalent forms

$$
\begin{aligned}
\nabla_\theta J(\theta) &= \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \; ln \; \pi_\theta(s,a) \; G_t\right] && \text{REINFORCE} \\
&= \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \; ln \; \pi_\theta(s,a) \; Q^w(s,a)\right] && \text{Q Actor-Critic} \\
&= \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \; ln \; \pi_\theta(s,a) \; A^w(s,a)\right] && \text{Advantage Actor-Critic} \\
&= \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \; ln \; \pi_\theta(s,a) \; \delta\right] && \text{TD Actor-Critic} \\
&= \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \; ln \; \pi_\theta(s,a) \; \delta e\right] && \text{TD}(\lambda)\text{ Actor-Critic}
\end{aligned}
$$

- Each leads to a stochastic gradient ascent algorithm.