UC San Diego
JACOBS SCHOOL OF ENGINEERING

ECE/SIOC 228 Machine Learning for Physical Applications
Lecture 14: Deep Reinforcement Learning I

Yuanyuan Shi

Assistant Professor, ECE

University of California, San Diego

Reference & Acknowledgement: David Silver's Course on RL https://www.davidsilver.uk/teaching/
Richard Sutton and Andrew Barto, Reinforcement Learning: An Introduction http://incompleteideas.net/book/the-book-2nd.html

Course Structure

UC San Diego

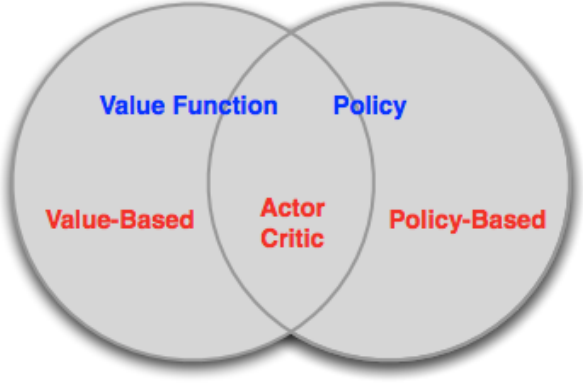
- **Planning** in Markov Decision Process (offline)
  - Have the environment model (reward, transition)
  - Find the optimal policy via value iteration and policy iteration

- **Reinforcement Learning** in Markov Decision Process (online)
  - Don't know how the environment works
  - Find the optimal policy by interacting with the environment (taking actions and collect reward)

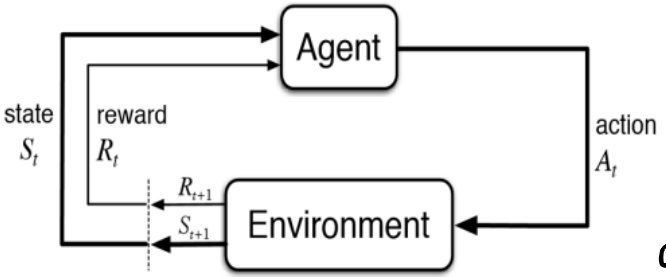
Reinforcement Learning Algorithms

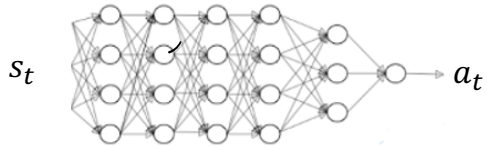
UC San Diego

- **Policy Based Reinforcement Learning**
  - Learn a policy
  - No value function / not necessarily have a value function
- Value Based Reinforcement Learning
  - Learn value function
  - Policy is implicit (e.g., greedy)
- Actor-Critic
  - Learn value function
  - Learn policy



UC San Diego



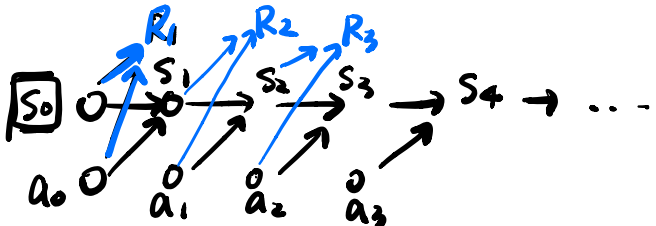


$\pi_\theta(a_t|s_t)$

Goal: given policy $\pi_\theta(s,a)$, with parameters θ, find best θ

- How do we measure the Quality of policy? — value function:
- Episodic task, initial state $s_0 \sim p_0$

$$J(\theta) = E_{s_0 \sim p_0}[V^{\pi_\theta}(s_0)]$$



- Continuing task: $J(\theta) = \sum_s d^{\pi_\theta}(s) V^{\pi_\theta}(s)$

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

☆ Policy based RL is an optimization problem

☆ Find θ^* that maximizes $J(\theta)$

☆ Use Gradient ascent.

$$\theta_{t+1} \leftarrow \theta_t + \alpha \nabla_\theta J(\theta)$$

Policy Gradient

UC San Diego

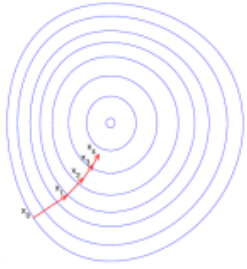
- Let $J(\theta)$ be any policy objective function
- Policy gradient algorithms search for a *local* maximum in $J(\theta)$ by ascending the gradient of the policy, w.r.t. parameters θ

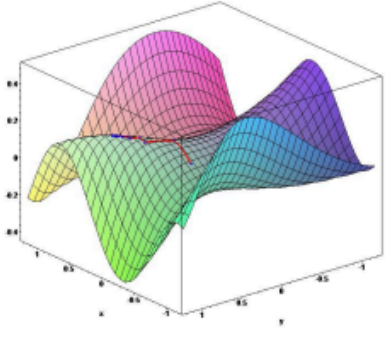
$$\Delta\theta = \alpha \nabla_\theta J(\theta)$$

- Where $\nabla_\theta J(\theta)$ is the policy gradient

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

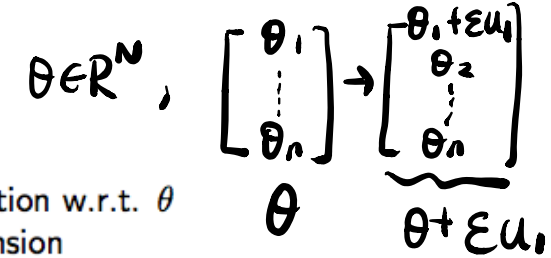
- and α is a step-size parameter





Computing Gradients By Finite Differences

UC San Diego



- To evaluate policy gradient of $\pi_\theta(s, a)$
- For each dimension $k \in [1, n]$
  - Estimate kth partial derivative of objective function w.r.t. θ
  - By perturbing θ by small amount ϵ in kth dimension

$$\frac{\partial J(\theta)}{\partial \theta_k} \approx \frac{J(\theta + \epsilon u_k) - J(\theta)}{\epsilon}$$

where u_k is unit vector with 1 in kth component, 0 elsewhere

- Uses n evaluations to compute policy gradient in n dimensions
- Simple, noisy, inefficient - but sometimes effective
- Works for arbitrary policies, even if policy is not differentiable

- We now compute the policy gradient analytically
- Assume policy $\pi_\theta(s,a)$ is differentiable

Episodic task: $J(\theta) = E_{s_0 \sim p_0}[V^{\pi_\theta}(s_0)]$

$= E_{s_0 \sim p_0}[E_{\pi_\theta}[G_0 | s_0]] = E_{z \sim p_\theta(z)}[\underbrace{\sum_{t=0}^{T-1} \gamma^t R(s_t, a_t)}_{r(z)}]$

$r(z)$ is the return (total discounted reward) of an episode

$z = s_0, a_0, s_1, a_1, s_2, a_2, \cdots, s_{T-1}, a_{T-1}, s_T$

$$P_\theta(z) = P_0(s_0) \prod_{t=0}^{T-1} \pi_\theta(a_t|s_t)\, p(s_{t+1}|s_t, a_t)$$

Find the best θ that optimize $J(\theta)$:

$$\theta^* = \arg\max_\theta J(\theta) = \arg\max_\theta E_{z\sim p_\theta(z)}[r(z)]$$

Leibniz integral rule

Policy gradient

$$\nabla_\theta J(\theta) = \nabla_\theta E_z[r(z)] = \nabla_\theta \int p_\theta(z)\, r(z)\, dz = \int \nabla_\theta p_\theta(z)\, r(z)\, dz$$

$$= \int p_\theta(z) \frac{\nabla_\theta p_\theta(z)}{p_\theta(z)} r(z)\, dz = \int p_\theta(z) \nabla_\theta \log p_\theta(z)\, r(z)\, dz$$

$$= E_{z\sim p_\theta(z)}[\nabla_\theta \log p_\theta(z)\, r(z)]$$

(From previous page) $\nabla_\theta J(\theta) = E_{z\sim p_\theta(z)}\left[\nabla_\theta \log p_\theta(z)\, r(z)\right]$

$\nabla_\theta \log p_\theta(z)$: ?

$r(z)$: return of episode z

$$p_\theta(z) = p_0(s_0)\prod_{t=0}^{T-1}\pi_\theta(a_t|s_t)\,p(s_{t+1}|s_t,a_t)$$

$$\log p_\theta(z) = \log p_0(s_0) + \sum_{t=0}^{T-1}\log\pi_\theta(a_t|s_t) + \sum_{t=0}^{T-1}\log p(s_{t+1}|s_t,a_t)$$

$$\nabla_\theta \log p_\theta(z) = \sum_{t=0}^{T-1}\nabla_\theta \log\pi_\theta(a_t|s_t)$$

$$\Rightarrow \nabla_\theta J(\theta) = E_{z\sim p_\theta(z)}\left[\sum_{t=0}^{T-1}\nabla_\theta \log\pi_\theta(a_t|s_t)\, r(z)\right]$$

- This result is elegant, since to compute $\nabla_\theta J(\theta)$, we don't need to know the transition matrix $P(s_{t+1}|s_t, a_t)$. ⇒ "model-free"
- Evaluating the expectation $E_{z\sim p_\theta(z)}[\cdots]$ still remains a challenge ⇒ use sampled episodes

$$\nabla_\theta J(\theta) = E_{\tau \sim p_\theta(\tau)}\left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t)\, r(\tau)\right]$$

$$\approx \frac{1}{N}\sum_{i=1}^{N} \underbrace{\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t^i|s_t^i)}_{\nabla_\theta \log p_\theta(\tau)} \underbrace{\left(\sum_{t=0}^{T-1} \gamma^t r(s_t^i, a_t^i)\right)}_{\text{maximum "return"-weighted likelihood}}$$

- high return episode is given a larger weight ⇒ become more likely
- low return episode is give a smaller weight ⇒ become less likely