Reinforcement Learning: Policy Gradients

Forest Agostinelli
University of South Carolina

Topics Covered in This Class

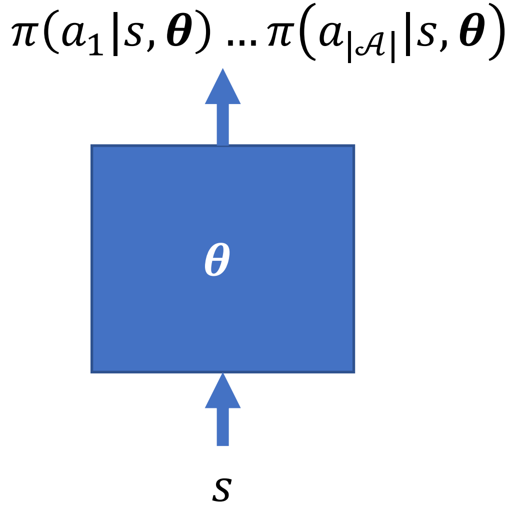
- **Part 1: Search**
  - Pathfinding
    - Uninformed search
    - Informed search
  - Adversarial search
  - Optimization
    - Local search
    - Constraint satisfaction
- **Part 2: Knowledge Representation and Reasoning**
  - Propositional logic
  - First-order logic
  - Prolog
- **Part 3: Knowledge Representation and Reasoning Under Uncertainty**
  - Probability
  - Bayesian networks
- **Part 4: Machine Learning**
  - Supervised learning
    - Inductive logic programming
    - Linear models
    - Deep neural networks
    - PyTorch
  - Reinforcement learning
    - Markov decision processes
    - Dynamic programming
    - Model-free RL
  - Unsupervised learning
    - Clustering
    - Autoencoders

Outline

- Motivation
- Policy gradients (one-step MDP)
- Policy gradient theorem
- Policy gradients with baseline
- Actor-Critic
- Deep deterministic policy gradients

Induced Policies

- The goal of reinforcement learning is to find a policy that maximizes the expected future reward
- So far, we have only learned a policy indirectly by behaving greedily with respect to a value function
  - $\pi(s) = \underset{a}{\operatorname{argmax}}(r(s,a) + \gamma \sum_{s'} p(s'|s,a)\, \hat{v}(s', \boldsymbol{w}))$
  - $\pi(s) = \underset{a}{\operatorname{argmax}}\, \hat{q}(s,a,\boldsymbol{w})$
- Instead, we can learn a policy directly

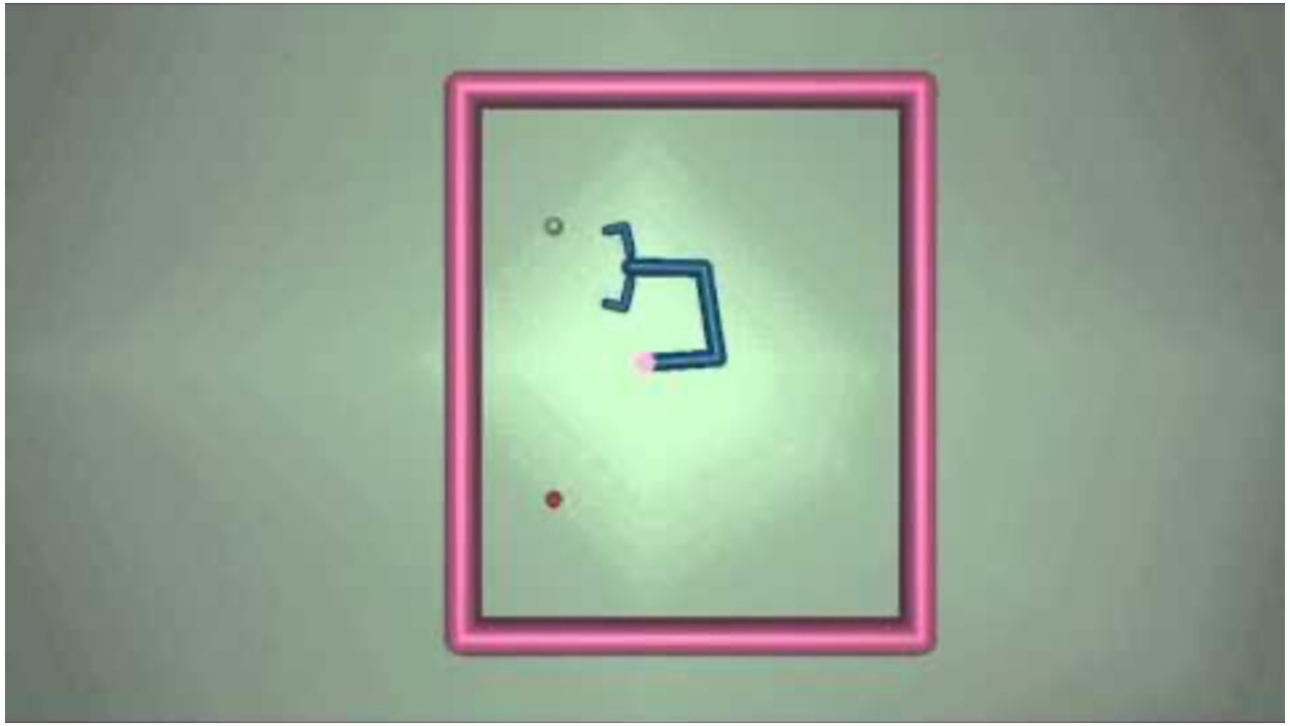


Motivation: Function Complexity

- A policy function may be easier to learn
- Breakout
  - Value function: expected future reward
    - How many blocks are left?
    - Are there any special blocks?
    - How long will it take to clear them all?
  - Policy Function: left, right, or do nothing?
    - Where am I?
    - Where is the ball?
    - Where do I want it to go?

Motivation: Continuous Action Spaces

- Difficult to maximize over all possible actions if there are an infinite number of actions

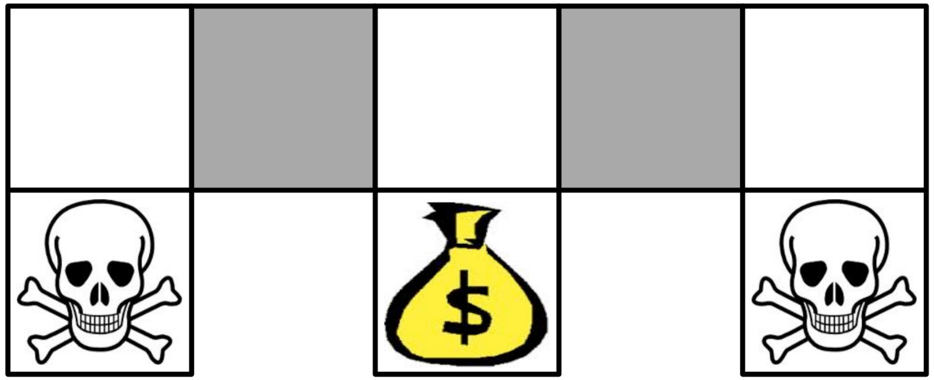


DeepMind Asynchronous Advantage Actor-Critic (A3C)

Motivation: Convergence

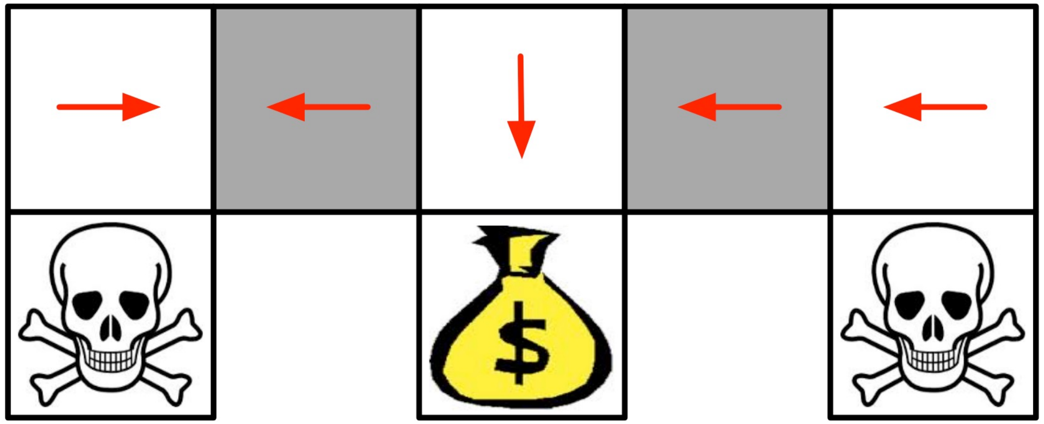
- Induced policies can change drastically based on a small change in the value function
- Parameterized policies can change smoothly
- This can help with convergence in certain cases

Motivation: Aliasing



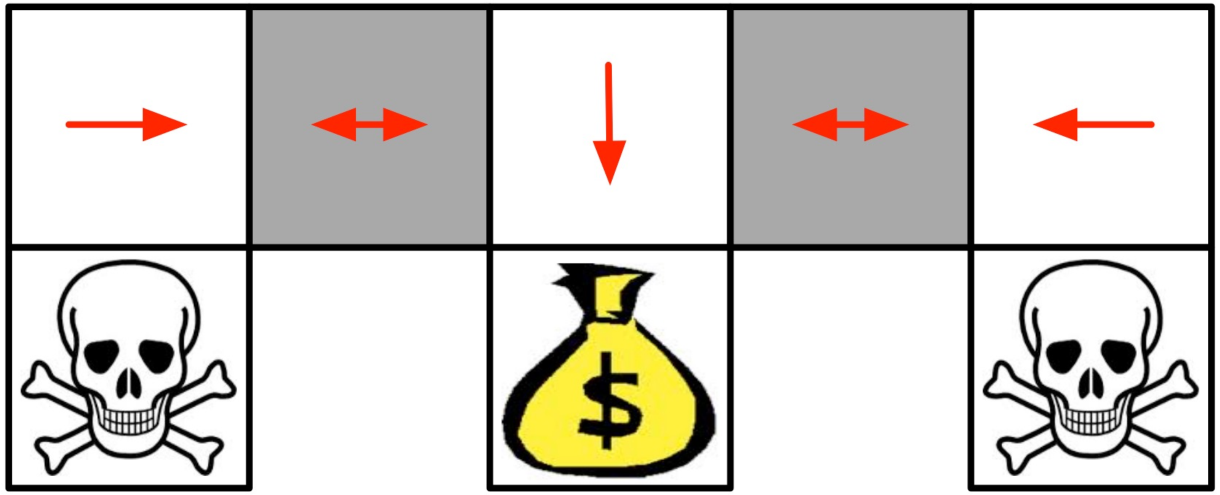
- Aliasing can occur due to
  - Limited sensors
  - Not being able to differentiate based on input features
  - Approximation architecture not being expressive enough

Motivation: Aliasing



- A deterministic policy will not reach the goal from every state
- What about ϵ-greedy?

Motivation: Aliasing



- To maximize our expected future reward, we need to have a stochastic policy such that
  - Move left with probability 0.5 and right with probability 0.5 in only the aliased states
- We cannot accomplish this by behaving greedily or ϵ-greedily with respect to a value function
- We can accomplish this by parameterizing a policy and learning $\pi(a|s)$

Motivation: Adversarial Settings

- In a setting with a non-stationary adversary, it can be good to act randomly
- Rock, paper, scissors

Parameterized Policy: Softmax Function

- Discrete actions
  - We need to make sure that the probability sums to 1
    - $\sum_a \pi(a|s) = 1$
  - $\pi(a|s) = \frac{e^{h(s,a,\boldsymbol{\theta})}}{\sum_{a'} e^{h(s,a',\boldsymbol{\theta})}}$
- Continuous actions
  - $a \sim \mathcal{N}(\mu(s,\boldsymbol{\theta}), \sigma(s,\boldsymbol{\theta}))$
  - $\pi(a|s) = \frac{1}{\sigma(s,\boldsymbol{\theta})\sqrt{2\pi}} e^{-\frac{(a-\mu(s,\boldsymbol{\theta}))^2}{2\sigma(s,\boldsymbol{\theta})^2}}$

One-Step Case

- Episodic (one step)
- Model-free
- Stochastic
- How will you adjust the parameters of the policy function?
  - We want to maximize that reward we receive after one step

Policy Gradients: One-step MDP

- Starting state $s \sim d$
- Objective
  - $J(\boldsymbol{\theta}) = \mathbb{E}_{s \sim d, a \sim \pi_{\boldsymbol{\theta}}}[r(s,a)]$
  - $J(\boldsymbol{\theta}) = \mathbb{E}_{s \sim d}[\sum_a \pi(a|s,\boldsymbol{\theta}) r(s,a)]$
  - $J(\boldsymbol{\theta}) = \sum_s d(s) \sum_a \pi(a|s,\boldsymbol{\theta}) r(s,a)$
- $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \sum_s d(s) \sum_a \nabla_{\boldsymbol{\theta}} \pi(a|s,\boldsymbol{\theta}) r(s,a)$
- $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbb{E}_{s \sim d}[\sum_a \nabla_{\boldsymbol{\theta}} \pi(a|s,\boldsymbol{\theta}) r(s,a)]$
  - Have to try all actions in a state before we can compute the gradient
  - However, we are assuming we are model-free
  - Hard to sample from this!

Policy Gradients: One-step MDP

- $J(\boldsymbol{\theta}) = \sum_s d(s) \sum_a \pi(a|s,\boldsymbol{\theta}) r(s,a)$
- $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbb{E}_{s\sim d}[\sum_a \nabla_{\boldsymbol{\theta}} \pi(a|s,\boldsymbol{\theta}) r(s,a)]$
- Use the likelihood ratio trick: $\nabla_{\boldsymbol{\theta}} \ln \pi(a|s,\boldsymbol{\theta}) = \frac{\nabla_{\boldsymbol{\theta}} \pi(a|s,\boldsymbol{\theta})}{\pi(a|s,\boldsymbol{\theta})}$
- $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbb{E}_{s\sim d}[\sum_a \pi(a|s,\boldsymbol{\theta}) \frac{\nabla_{\boldsymbol{\theta}} \pi(a|s,\boldsymbol{\theta})}{\pi(a|s,\boldsymbol{\theta})} r(s,a)]$
- $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbb{E}_{s\sim d}[\sum_a \pi(a|s,\boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}} \ln \pi(a|s,\boldsymbol{\theta})\, r(s,a)]$
- $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbb{E}_{s\sim d, a\sim \pi_{\boldsymbol{\theta}}}[\nabla_{\boldsymbol{\theta}} \ln \pi(a|s,\boldsymbol{\theta})\, r(s,a)]$

Policy Gradients: One-step MDP

- Now we can do gradient ascent by sampling the gradient
- $J(\boldsymbol{\theta}) = \sum_s d(s) \sum_a \pi(a|s, \boldsymbol{\theta}) r(s, a)$
- $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbb{E}_{s \sim d, a \sim \pi_{\boldsymbol{\theta}}} [\nabla_{\boldsymbol{\theta}} \ln \pi(a|s, \boldsymbol{\theta})\, r(s, a)]$
- $\boldsymbol{\theta} = \boldsymbol{\theta} + \alpha \frac{1}{m} \sum_{i=0}^{m} \nabla_{\boldsymbol{\theta}} \ln \pi\left(a^{(i)} \middle| s^{(i)}, \boldsymbol{\theta}\right) r^{(i)}$
  - Average over m episodes

Policy Gradients: One-step MDP

- $\boldsymbol{\theta} = \boldsymbol{\theta} + \alpha \frac{1}{m} \sum_{i=0}^{m} \nabla_{\boldsymbol{\theta}} \ln \pi\left(a^{(i)} \middle| s^{(i)}, \boldsymbol{\theta}\right) r^{(i)}$
- What is the intuition behind this update?
- $\boldsymbol{\theta} = \boldsymbol{\theta} + \alpha \frac{1}{m} \sum_{i=0}^{m} \frac{\nabla_{\boldsymbol{\theta}} \pi\left(a^{(i)} \middle| s^{(i)}, \boldsymbol{\theta}\right)}{\pi\left(a^{(i)} \middle| s^{(i)}, \boldsymbol{\theta}\right)} r^{(i)}$
- The update is proportional to the reward obtained
- The update is inversely proportional to the probability of taking that action
  - Frequently selected actions get updated more frequently
  - Therefore, we should be more aggressive when updating actions that are selected less frequently

Policy Gradient Theorem

- One-step MDP
  - $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbb{E}[\nabla_{\boldsymbol{\theta}} \ln \pi(a|s, \boldsymbol{\theta})\, r(s, a)]$
- Generalizes to multi-step MDPs
  - $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbb{E}[\nabla_{\boldsymbol{\theta}} \ln \pi(a|s, \boldsymbol{\theta})\, q_{\pi_{\boldsymbol{\theta}}}(s, a)]$
- We do not know $q_{\pi_{\boldsymbol{\theta}}}(s, a)$, so we can sample it with the return G
- After sampling m episodes
- $\boldsymbol{\theta} = \boldsymbol{\theta} + \alpha \frac{1}{m} \sum_{i=0}^{m} \sum_{t=0}^{T-1} \nabla_{\boldsymbol{\theta}} \ln \pi \left(a_t^{(i)} \middle| s_t^{(i)} \right) G_t$

Policy Gradients: REINFORCE

REINFORCE: Monte-Carlo Policy-Gradient Control (episodic) for π_*

Input: a differentiable policy parameterization $\pi(a|s, \boldsymbol{\theta})$
Algorithm parameter: step size $\alpha > 0$
Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ (e.g., to $\mathbf{0}$)

Loop forever (for each episode):
    Generate an episode $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$, following $\pi(\cdot|\cdot, \boldsymbol{\theta})$
    Loop for each step of the episode $t = 0, 1, \ldots, T-1$:
        $G \leftarrow \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$ $\qquad (G_t)$
        $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \gamma^t G \nabla \ln \pi(A_t|S_t, \boldsymbol{\theta})$

- Monte-Carlo algorithm
- On-policy algorithm

Williams, Ronald J. "Simple statistical gradient-following algorithms for connectionist reinforcement learning." Machine learning 8.3-4 (1992): 229-256.

Policy Gradient Derivation

- Trajectory $\tau = (s_0, a_0, r_1, \dots s_{T-1}, a_{T-1}, r_T, s_T)$
- $P(\tau, \boldsymbol{\theta}) = P(s_0) \prod_{t=0}^{T-1} \pi(a_t | s_t, \boldsymbol{\theta}) P(s_{t+1} | s_t, a_t)$
- $J(\boldsymbol{\theta}) = \mathbb{E}_{a \sim \pi_{\boldsymbol{\theta}}}[\sum_{t=0}^{T-1} r(s_t, a_t)]$
  - $R(\tau) = \sum_{t=0}^{T-1} r(s_t, a_t)$
- $J(\boldsymbol{\theta}) = \mathbb{E}_{\tau \sim P(\tau, \boldsymbol{\theta})}[R(\tau)] = \sum_\tau P(\tau, \boldsymbol{\theta}) R(\tau)$
- $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \sum_\tau \nabla_{\boldsymbol{\theta}} P(\tau, \boldsymbol{\theta}) R(\tau)$
- $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \sum_\tau P(\tau, \boldsymbol{\theta}) \frac{\nabla_{\boldsymbol{\theta}} P(\tau, \boldsymbol{\theta})}{P(\tau, \boldsymbol{\theta})} R(\tau)$
  - Likelihood ratio trick
- $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \sum_\tau P(\tau, \boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}} \ln P(\tau, \boldsymbol{\theta})\, R(\tau) = \mathbb{E}_{\tau \sim P(\tau, \boldsymbol{\theta})}[\nabla_{\boldsymbol{\theta}} \ln P(\tau, \boldsymbol{\theta})\, R(\tau)]$

Policy Gradient Derivation

- $J(\boldsymbol{\theta}) = \mathbb{E}_{a\sim\pi_{\boldsymbol{\theta}}}[\sum_{t=0}^{T-1} r(s_t, a_t)] = \sum_{\tau} P(\tau, \boldsymbol{\theta})R(\tau)$
- $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbb{E}_{\tau\sim P(\tau,\boldsymbol{\theta})}[\nabla_{\boldsymbol{\theta}} \ln P(\tau, \boldsymbol{\theta})\, R(\tau)]$
- We can then estimate the gradient through sampling
- $\boldsymbol{\theta} = \boldsymbol{\theta} + \alpha \frac{1}{m} \sum_{i=0}^{m} \nabla_{\boldsymbol{\theta}} \ln P\left(\tau^{(i)}, \boldsymbol{\theta}\right) R(\tau^{(i)})$
  - How do we take the gradient of the probability of a trajectory?
  - $P(\tau, \boldsymbol{\theta}) = P(s_0) \prod_{t=0}^{T-1} \pi(a_t | s_t, \boldsymbol{\theta}) P(s_{t+1} | s_t, a_t)$
  - We probably do not know the state transition probabilities
  - Even if we do, it may not be differentiable

Policy Gradient Derivation

- $\boldsymbol{\theta} = \boldsymbol{\theta} + \alpha \frac{1}{m} \sum_{i=0}^{m} \nabla_{\boldsymbol{\theta}} \ln P\left(\tau^{(i)}, \boldsymbol{\theta}\right) R(\tau^{(i)})$
- $P(\tau, \boldsymbol{\theta}) = P(s_0) \prod_{t=0}^{T-1} \pi(a_t | s_t, \boldsymbol{\theta}) P(s_{t+1} | s_t, a_t)$
- $\nabla_{\boldsymbol{\theta}} \ln P\left(\tau^{(i)}, \boldsymbol{\theta}\right) = \nabla_{\boldsymbol{\theta}} \ln[P(s_0) \prod_{t=0}^{T-1} \pi(a_t | s_t, \boldsymbol{\theta}) P(s_{t+1} | s_t, a_t)]$
- $= \nabla_{\boldsymbol{\theta}} (\ln P(s_0) + \sum_{t=0}^{T-1} \ln \pi(a_t | s_t, \boldsymbol{\theta}) + \sum_{t}^{T} \ln P(s_{t+1} | s_t, a_t))$
- $= \nabla_{\boldsymbol{\theta}} \ln P(s_0) + \sum_{t=0}^{T-1} \nabla_{\boldsymbol{\theta}} \ln \pi(a_t | s_t, \boldsymbol{\theta}) + \sum_{t}^{T} \nabla_{\boldsymbol{\theta}} \ln P(s_{t+1} | s_t, a_t)$
- $= \sum_{t=0}^{T-1} \nabla_{\boldsymbol{\theta}} \ln \pi(a_t | s_t, \boldsymbol{\theta})$

Policy Gradient Derivation

- $\boldsymbol{\theta} = \boldsymbol{\theta} + \alpha \frac{1}{m} \sum_{i=0}^{m} \nabla_{\boldsymbol{\theta}} \ln P\left(\tau^{(i)}, \boldsymbol{\theta}\right) R\left(\tau^{(i)}\right)$
- $\nabla_{\boldsymbol{\theta}} \ln P\left(\tau^{(i)}, \boldsymbol{\theta}\right) = \sum_{t=0}^{T-1} \nabla_{\boldsymbol{\theta}} \ln \pi(a_t | s_t, \boldsymbol{\theta})$
- $\boldsymbol{\theta} = \boldsymbol{\theta} + \alpha \frac{1}{m} \sum_{i=0}^{m} \sum_{t=0}^{T-1} \nabla_{\boldsymbol{\theta}} \ln \pi \left(a_t^{(i)} \middle| s_t^{(i)}, \boldsymbol{\theta}\right) R\left(\tau^{(i)}\right)$
  - Reward of entire trajectory gets applied to each decision made, even past decisions

Policy Gradient Derivation

- $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbb{E}_{\tau \sim P(\tau,\boldsymbol{\theta})}[\nabla_{\boldsymbol{\theta}} \ln P(\tau, \boldsymbol{\theta})\, R(\tau)]$
- $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbb{E}_{\tau \sim P(\tau,\boldsymbol{\theta})}[\sum_{t=0}^{T-1} \nabla_{\boldsymbol{\theta}} \ln \pi(a_t|s_t, \boldsymbol{\theta})\, R(\tau)]$
- $R(\tau) = \sum_{t=0}^{T-1} r(s_t, a_t)$
- $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbb{E}_{\tau \sim P(\tau,\boldsymbol{\theta})}[\sum_{t=0}^{T-1} \nabla_{\boldsymbol{\theta}} \ln \pi(a_t|s_t, \boldsymbol{\theta})\, (\sum_{k=0}^{t-1} r(s_k, a_k) + \sum_{k'=t}^{T-1} r(s_{k'}, a_{k'}))]$
  - $\sum_{k=0}^{t-1} r(s_k, a_k)$ does not depend on the current state and action, therefore, we remove it
- $= \mathbb{E}_{\tau \sim P(\tau,\boldsymbol{\theta})}[\sum_{t=0}^{T-1} \nabla_{\boldsymbol{\theta}} \ln \pi(a_t|s_t, \boldsymbol{\theta})\, \sum_{k'=t}^{T-1} r(s_{k'}, a_{k'})]$
- $= \mathbb{E}_{\tau \sim P(\tau,\boldsymbol{\theta})}[\sum_{t=0}^{T-1} \nabla_{\boldsymbol{\theta}} \ln \pi(a_t|s_t, \boldsymbol{\theta})\, q_{\pi_{\boldsymbol{\theta}}}(s, a)]$

Policy Gradient Derivation

- $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbb{E}_{\tau \sim P(\tau, \boldsymbol{\theta})}[\sum_{t=0}^{T-1} \nabla_{\boldsymbol{\theta}} \ln \pi(a_t | s_t, \boldsymbol{\theta})\, q_{\pi_{\boldsymbol{\theta}}}(s, a)]$
- $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbb{E}_{s \sim \mu, a \sim \pi_{\boldsymbol{\theta}}}[\nabla_{\boldsymbol{\theta}} \ln \pi(a_t | s_t, \boldsymbol{\theta}) q_{\pi_{\boldsymbol{\theta}}}(s, a)]$
  - Where $\mu(s)$ is the probability of seeing state s under policy $\pi_{\boldsymbol{\theta}}$
- $\boldsymbol{\theta} = \boldsymbol{\theta} + \alpha \frac{1}{m} \sum_{i=0}^{m} \sum_{t=0}^{T-1} \nabla_{\boldsymbol{\theta}} \ln \pi \left(a_t^{(i)} \middle| s_t^{(i)}, \boldsymbol{\theta} \right) G_t^{(i)}$

Policy Gradient Derivation

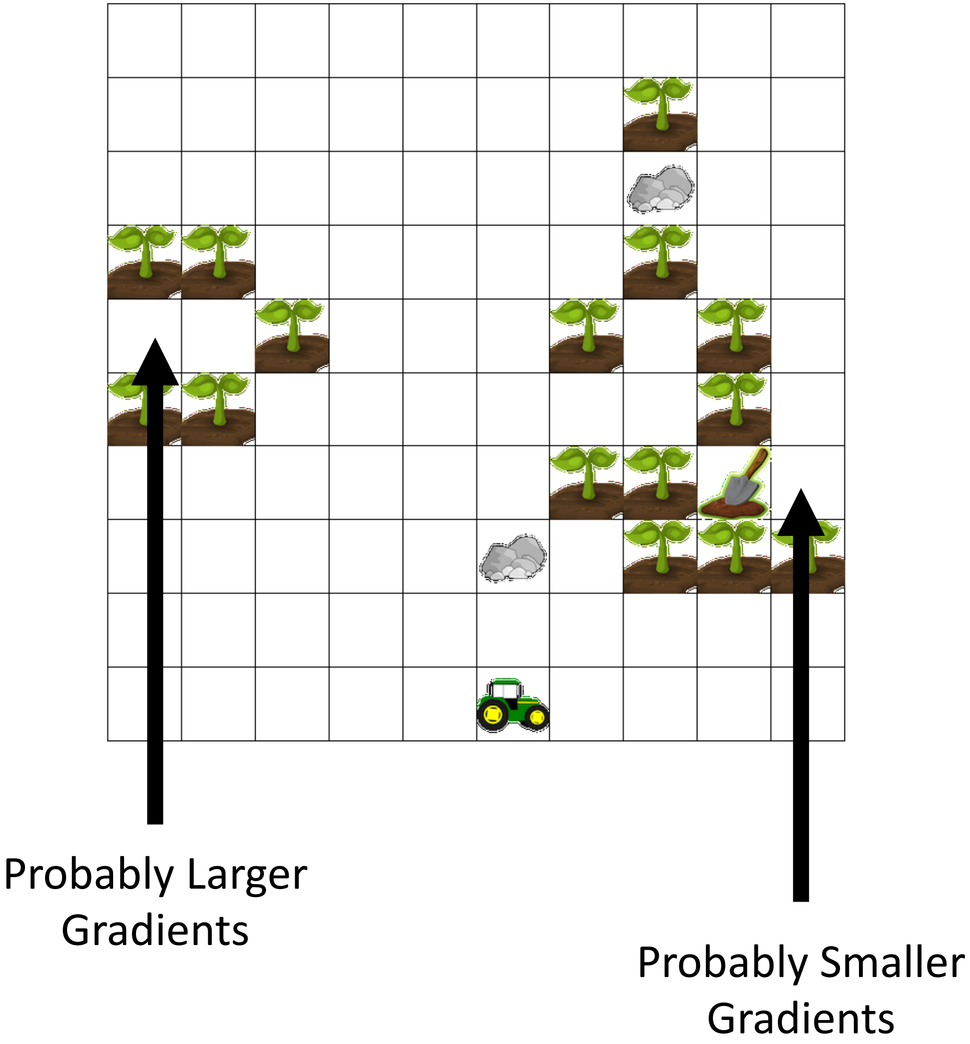
- $J(\boldsymbol{\theta}) = \mathbb{E}_{a\sim\pi_{\boldsymbol{\theta}}}[\sum_{t=0}^{T-1} r(s_t, a_t)] = \sum_{\tau} P(\tau, \boldsymbol{\theta})R(\tau)$
- $P(\tau, \boldsymbol{\theta}) = P(s_0) \prod_{t=0}^{T-1} \pi(a_t | s_t, \boldsymbol{\theta})P(s_{t+1} | s_t, a_t)$
- $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \sum_{\tau} \nabla_{\boldsymbol{\theta}} P(\tau, \boldsymbol{\theta})R(\tau)$
- $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \sum_{\tau} P(\tau, \boldsymbol{\theta}) \frac{\nabla_{\boldsymbol{\theta}} P(\tau,\boldsymbol{\theta})}{P(\tau,\boldsymbol{\theta})} R(\tau)$
  - Likelihood ratio trick
- $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \sum_{\tau} P(\tau, \boldsymbol{\theta})\nabla_{\boldsymbol{\theta}} \ln P(\tau, \boldsymbol{\theta})\, R(\tau) = \mathbb{E}_{\tau\sim P(\tau,\boldsymbol{\theta})}[\nabla_{\boldsymbol{\theta}} \ln P(\tau, \boldsymbol{\theta})\, R(\tau)]$
- $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbb{E}_{\tau\sim P(\tau,\boldsymbol{\theta})}[\sum_{t=0}^{T-1} \nabla_{\boldsymbol{\theta}} \ln \pi(a_t | s_t, \boldsymbol{\theta})\, R(\tau)]$
  - $P(s_0)$ and $P(s_{t+1} | s_t, a_t)$ not a function of $\boldsymbol{\theta}$
- $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbb{E}_{\tau\sim P(\tau,\boldsymbol{\theta})}[\sum_{t=0}^{T-1} \nabla_{\boldsymbol{\theta}} \ln \pi(a_t | s_t, \boldsymbol{\theta})\, q_{\pi_{\boldsymbol{\theta}}}(s, a)]$
  - Remove previous rewards
- $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \approx \frac{1}{m} \sum_{i=0}^{m} \sum_{t=0}^{T-1} \nabla_{\boldsymbol{\theta}} \ln \pi\left(a_t^{(i)} \middle| s_t^{(i)}, \boldsymbol{\theta}\right) G_t^{(i)}$
- $\boldsymbol{\theta} = \boldsymbol{\theta} + \alpha \frac{1}{m} \sum_{i=0}^{m} \sum_{t=0}^{T-1} \nabla_{\boldsymbol{\theta}} \ln \pi\left(a_t^{(i)} \middle| s_t^{(i)}, \boldsymbol{\theta}\right) G_t^{(i)}$

Connection to Policy Iteration

- Policy gradients are still doing a form of policy iteration
- Policy evaluation
  - Calculate the return obtained by your policy over multiple runs
- Policy improvement
  - Take a gradient step to improve your objective

Policy Gradient with a Baseline

- $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \approx \frac{1}{m} \sum_{i=0}^{m} \sum_{t=0}^{T-1} \nabla_{\boldsymbol{\theta}} \ln \pi \left(a_t^{(i)} \middle| s_t^{(i)}, \boldsymbol{\theta} \right) G_t^{(i)}$
- Some gradients will be larger than others
  - That does not mean the update should be larger
  - Relationship to Huber loss for DQNs
- $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \approx \frac{1}{m} \sum_{i=0}^{m} \sum_{t=0}^{T-1} \nabla_{\boldsymbol{\theta}} \ln \pi \left(a_t^{(i)} \middle| s_t^{(i)}, \boldsymbol{\theta} \right) \left(G_t^{(i)} - b(s_t) \right)$
- Reduces variance
- Does not change expectation of gradient if the baseline is independent of the action

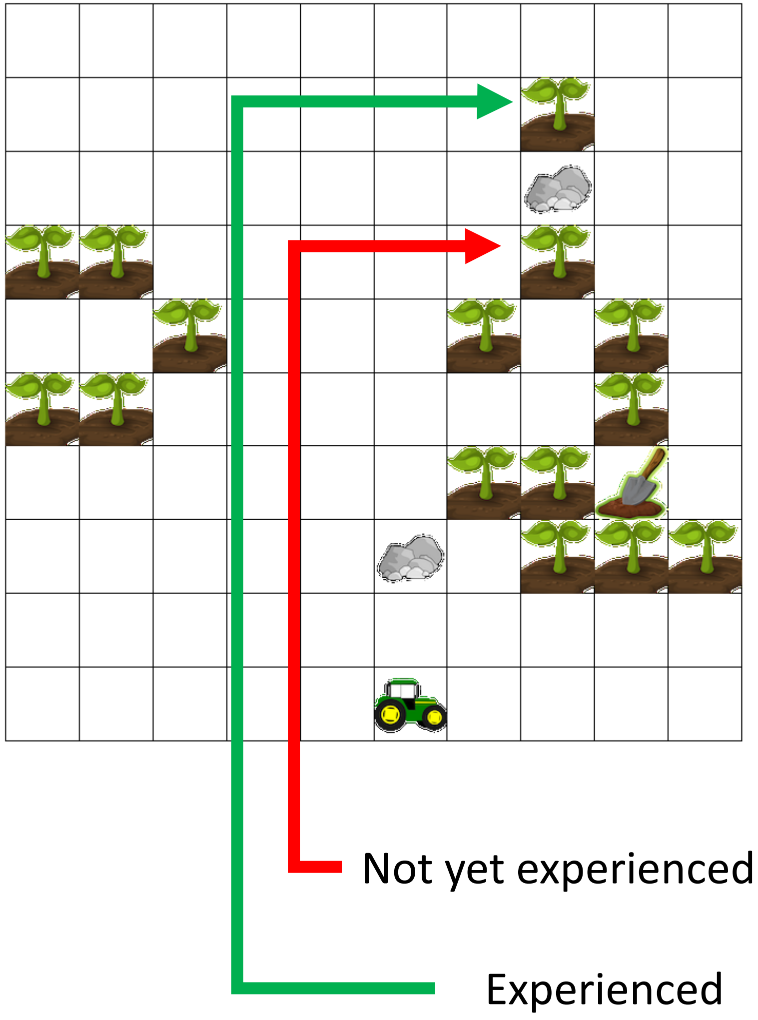


Policy Gradients: Baseline

- $\mathbb{E}_{s\sim\mu,a\sim\pi_{\boldsymbol{\theta}}}\left[\nabla_{\boldsymbol{\theta}} \ln \pi(a|s,\boldsymbol{\theta})\left(q_{\pi_{\boldsymbol{\theta}}}(s,a)-b(s)\right)\right]$
- $= \mathbb{E}_{s\sim\mu,a\sim\pi_{\boldsymbol{\theta}}}\left[\nabla_{\boldsymbol{\theta}} \ln \pi(a|s,\boldsymbol{\theta})\, q_{\pi_{\boldsymbol{\theta}}}(s,a)\right] - \mathbb{E}_{s\sim\mu,a\sim\pi_{\boldsymbol{\theta}}}[\nabla_{\boldsymbol{\theta}} \ln \pi(a|s,\boldsymbol{\theta})\, b(s)]$
- $\mathbb{E}_{s\sim\mu,a\sim\pi_{\boldsymbol{\theta}}}[\nabla_{\boldsymbol{\theta}} \ln \pi(a|s,\boldsymbol{\theta})\, b(s)] = \mathbb{E}_{s\sim\mu}\left[\sum_a \pi(a|s,\boldsymbol{\theta})\frac{\nabla_{\boldsymbol{\theta}}\pi(a|s,\boldsymbol{\theta})}{\pi(a|s,\boldsymbol{\theta})}b(s)\right]$
- $\sum_a \pi(a|s,\boldsymbol{\theta})\frac{\nabla_{\boldsymbol{\theta}}\pi(a|s,\boldsymbol{\theta})}{\pi(a|s,\boldsymbol{\theta})}b(s) = \sum_a \nabla_{\boldsymbol{\theta}}\pi(a|s,\boldsymbol{\theta})b(s)$
- $= b(s)\nabla_{\boldsymbol{\theta}}\sum_a \pi(a|s,\boldsymbol{\theta}) = b(s)\nabla_{\boldsymbol{\theta}}1 = 0$

Policy Gradient: Reducing Variance

- $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbb{E}_{\tau \sim P(\tau,\boldsymbol{\theta})}[\sum_{t=0}^{T-1} \nabla_{\boldsymbol{\theta}} \ln \pi(a_t | s_t, \boldsymbol{\theta})\, q_{\pi_{\boldsymbol{\theta}}}(s,a)]$
- Using sampled returns will have variance due to randomness
  - $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \approx \frac{1}{m} \sum_{i=0}^{m} \sum_{t=0}^{T-1} \nabla_{\boldsymbol{\theta}} \ln \pi\left(a_t^{(i)} \middle| s_t^{(i)}, \boldsymbol{\theta}\right) G_t^{(i)}$
- Instead, we can approximate $q_{\pi_{\boldsymbol{\theta}}}(s,a)$
  - $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \approx \frac{1}{m} \sum_{i=0}^{m} \sum_{t=0}^{T-1} \nabla_{\boldsymbol{\theta}} \ln \pi\left(a_t^{(i)} \middle| s_t^{(i)}, \boldsymbol{\theta}\right) \hat{q}_{\pi_\theta}(s,a,\boldsymbol{w})$
  - Reduces variance
  - Since the function approximator most likely will have errors, this introduces bias
  - Policy is the actor and value function is the critic
    - While a learned value function can be used as a baseline, this is not actor-critic, because it does not introduce bias



Function approximator can generalize

Advantage Actor Critic (A2C)

- Policy gradient with sampled returns
  - $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \approx \frac{1}{m} \sum_{i=0}^{m} \sum_{t=0}^{T-1} \nabla_{\boldsymbol{\theta}} \ln \pi \left(a_t^{(i)} \middle| s_t^{(i)}, \boldsymbol{\theta} \right) G_t^{(i)}$
- Policy gradient with a critic
  - $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \approx \frac{1}{m} \sum_{i=0}^{m} \sum_{t=0}^{T-1} \nabla_{\boldsymbol{\theta}} \ln \pi \left(a_t^{(i)} \middle| s_t^{(i)}, \boldsymbol{\theta} \right) \hat{q}_{\pi_\theta}(s, a, \boldsymbol{w})$
- Advantage actor critic: policy gradient with a critic and baseline
  - $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \approx \frac{1}{m} \sum_{i=0}^{m} \sum_{t=0}^{T-1} \nabla_{\boldsymbol{\theta}} \ln \pi \left(a_t^{(i)} \middle| s_t^{(i)}, \boldsymbol{\theta} \right) \left(\hat{q}_{\pi_\theta}(s, a, \boldsymbol{w}) - \hat{v}(s, \boldsymbol{\phi}) \right)$
  - $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \approx \frac{1}{m} \sum_{i=0}^{m} \sum_{t=0}^{T-1} \nabla_{\boldsymbol{\theta}} \ln \pi \left(a_t^{(i)} \middle| s_t^{(i)}, \boldsymbol{\theta} \right) \hat{A}(s, a)$

Breakout Session: A2C Using Only State Value Function

- $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \approx \frac{1}{m} \sum_{i=0}^{m} \sum_{t=0}^{T-1} \nabla_{\boldsymbol{\theta}} \ln \pi \left(a_t^{(i)} \middle| s_t^{(i)}, \boldsymbol{\theta} \right) \left(\hat{q}_{\pi_\theta}(s, a, \boldsymbol{w}) - \hat{v}_{\pi_\theta}(s, \boldsymbol{\phi}) \right)$
- Advantages of using only a state value function
  - Fewer parameters
  - Depends on fewer variables than an action value function
- Can we approximate the advantage using only a state value function?
  - $\hat{A}(s, a) = \hat{q}_{\pi_\theta}(s, a, \boldsymbol{w}) - \hat{v}_{\pi_\theta}(s, \boldsymbol{\phi})$

A2C Using Only State Value Function

- $v_\pi(s) = \mathbb{E}_\pi[G_t | S_t = s] = \mathbb{E}_\pi[\sum_{k=0}^{\infty} \gamma^k R_{t+1+k} \mid S_t = s]$
- $q_\pi(s,a) = \mathbb{E}_\pi[G_t | S_t = s, A_t = a] = \mathbb{E}_\pi[\sum_{k=0}^{\infty} \gamma^k R_{t+1+k} \mid S_t = s, A_t = a]$
- $q_\pi(s,a) = r(s,a) + \gamma \sum_{s'} p(s' | s,a)\, v_\pi(s')$
- $q_\pi(s,a) \approx R_{t+1} + v_\pi(s')$
  - One step of sampling

A2C Using Only State Value Function

- $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \approx \frac{1}{m}\sum_{i=0}^{m}\sum_{t=0}^{T-1}\nabla_{\boldsymbol{\theta}}\ln\pi\left(a_t^{(i)}\middle|s_t^{(i)},\boldsymbol{\theta}\right)\left(R_{t+1}+\hat{v}_{\pi_\theta}(s',\boldsymbol{\phi})-\hat{v}_{\pi_\theta}(s,\boldsymbol{\phi})\right)$
- Our estimate of the advantage:
  - $R_{t+1}+\hat{v}_{\pi_\theta}(s',\boldsymbol{\phi})-\hat{v}_{\pi_\theta}(s,\boldsymbol{\phi})$
- May be easier to learn (less bias), but, due to the one step of sampling, more variance than:
  - $\hat{q}_{\pi_\theta}(s,a,\boldsymbol{w})-\hat{v}_{\pi_\theta}(s,\boldsymbol{\phi})$
- Less variance, but more biased than:
  - $G_t-\hat{v}_{\pi_\theta}(s,\boldsymbol{\phi})$

Training the Value Function

- $E(\mathbf{w}) = \frac{1}{2}\left(y - \hat{v}(s, \boldsymbol{w})\right)^2$
- $\nabla_{\mathbf{w}} E(\mathbf{w}) = \left(y - \hat{v}(s, \boldsymbol{w})\right) \nabla_{\mathbf{w}} \hat{v}(s, \boldsymbol{w})$
- Monte Carlo
  - $y = G_t$
- TD(0)
  - $y = R_{t+1} + \gamma \hat{v}(S_{t+1}, \boldsymbol{w})$
- n-Step TD
  - $y = \mathrm{R}_{\mathrm{t+1}} + \gamma \mathrm{R}_{\mathrm{t+2}} + \cdots + \gamma^{n-1} R_{t+n} + \gamma^n \hat{v}(S_{t+n}, \boldsymbol{w})$
- TD(λ)
  - Average over n-step returns

Advantage Actor Critic (A2C)

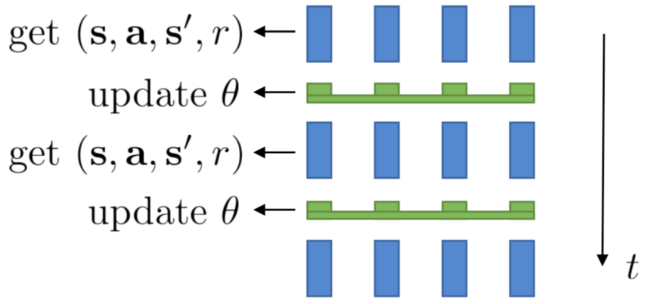
One-step Actor–Critic (episodic), for estimating $\pi_{\boldsymbol\theta} \approx \pi_*$

Input: a differentiable policy parameterization $\pi(a|s, \boldsymbol\theta)$
Input: a differentiable state-value function parameterization $\hat{v}(s,\mathbf{w})$
Parameters: step sizes $\alpha^{\boldsymbol\theta} > 0$, $\alpha^{\mathbf{w}} > 0$
Initialize policy parameter $\boldsymbol\theta \in \mathbb{R}^{d'}$ and state-value weights $\mathbf{w} \in \mathbb{R}^d$ (e.g., to $\mathbf{0}$)
Loop forever (for each episode):
    Initialize S (first state of episode)
    $I \leftarrow 1$
    Loop while S is not terminal (for each time step):
        $A \sim \pi(\cdot|S, \boldsymbol\theta)$
        Take action A, observe S', R
        $\delta \leftarrow R + \gamma\hat{v}(S',\mathbf{w}) - \hat{v}(S,\mathbf{w})$ (if S' is terminal, then $\hat{v}(S',\mathbf{w}) \doteq 0$)
        $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}}\delta\nabla\hat{v}(S,\mathbf{w})$
        $\boldsymbol\theta \leftarrow \boldsymbol\theta + \alpha^{\boldsymbol\theta} I\delta\nabla\ln\pi(A|S, \boldsymbol\theta)$
        $I \leftarrow \gamma I$
        $S \leftarrow S'$

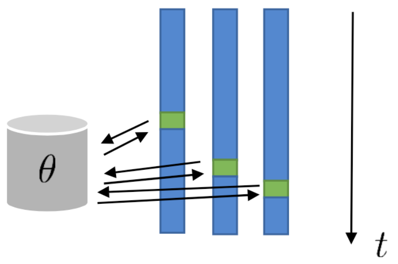
Online vs Batch Methods

- Online methods
  - Update value and policy function after every step
- Batch methods
  - Sample multiple steps or trajectories and then update value and policy function from batch
  - Can be more stable due to more samples

synchronized parallel actor-critic



asynchronous parallel actor-critic



Comparison of Methods

- Policy gradient with fixed baseline
  - $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \approx \frac{1}{m}\sum_{i=0}^{m}\sum_{t=0}^{T-1}\nabla_{\boldsymbol{\theta}}\ln\pi\left(a_t^{(i)}\middle|s_t^{(i)},\boldsymbol{\theta}\right)\left(G_t^{(i)}-b\right)$
  - Unbiased
  - Higher variance
- Policy gradient with state value function baseline
  - $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \approx \frac{1}{m}\sum_{i=0}^{m}\sum_{t=0}^{T-1}\nabla_{\boldsymbol{\theta}}\ln\pi\left(a_t^{(i)}\middle|s_t^{(i)},\boldsymbol{\theta}\right)\left(G_t^{(i)}-\hat{v}(s,\boldsymbol{w})\right)$
  - Unbiased
  - Lower variance
- Advantage actor critic
  - $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \approx \frac{1}{m}\sum_{i=0}^{m}\sum_{t=0}^{T-1}\nabla_{\boldsymbol{\theta}}\ln\pi\left(a_t^{(i)}\middle|s_t^{(i)},\boldsymbol{\theta}\right)\left(R_{t+1}+\hat{v}_{\pi_\theta}(s',\boldsymbol{\phi})-\hat{v}_{\pi_\theta}(s,\boldsymbol{\phi})\right)$
  - Biased
  - Lower variance

On-Policy vs Off-Policy

- $J(\boldsymbol{\theta}) = \mathbb{E}_{a\sim\pi_{\boldsymbol{\theta}}}[\sum_{t=0}^{T-1} r(s_t, a_t)]$
- Policy gradients are on-policy because we are assuming the actions are drawn according to the current policy $a \sim \pi_{\boldsymbol{\theta}}$
- Off-policy variants can be obtained from importance sampling or violating this on-policy assumption in a constrained manner
- Q-learning was a very convenient off-policy algorithm that could make use of data obtained from other policies
  - Is difficult in continuous action spaces

Q-learning in Continuous Action Spaces

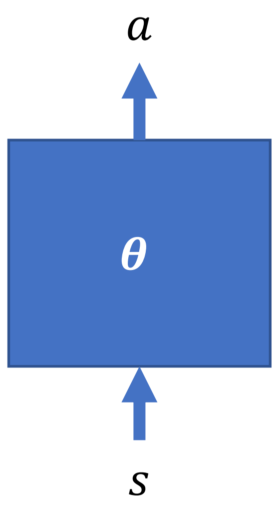
- Q-learning requires us to select actions and to bootstrap using a max over all actions
  - $\max_a Q(s, a)$
  - In general, this is not possible for continuous action spaces
- Analytical Methods
  - Restrict your Q-function to functions where it is easy to obtain the maximum (i.e. using simple calculus)
- Numerical Methods
  - Estimate the maximum by randomly sampling actions or using evolutionary methods

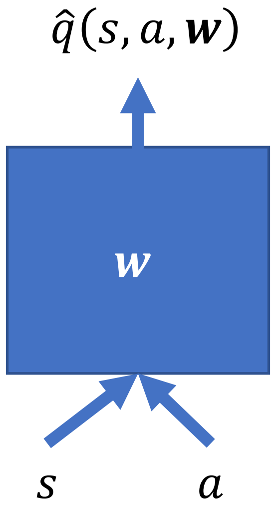
Deep Deterministic Policy Gradients

- Previously policy gradient methods
  - Need a policy that defines probabilities over actions
  - On-policy
- Q-learning
  - $\mathrm{Q}(s, a) = Q(s, a) + \alpha[r + \gamma \max_{a\prime} Q(s', a') - Q(s, a)]$
- DQN
  - Off-policy but, generally, does not work for continuous action spaces
  - $y = r + \gamma \max_{a\prime} \hat{q}(s', a', \boldsymbol{w})$
  - $E(\boldsymbol{w}) = \frac{1}{2}\big(y - \hat{q}(s, a, \boldsymbol{w})\big)^2$
- Deep Deterministic Policy Gradients (DDPG)
  - $y = r + \gamma \hat{q}(s', \pi(s, \boldsymbol{\theta}), \boldsymbol{w})$
  - $E(\boldsymbol{w}) = \frac{1}{2}\big(y - \hat{q}(s, a, \boldsymbol{w})\big)^2$
  - $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \approx \mathbb{E}\big[\nabla_{\boldsymbol{\theta}} \hat{q}(s, a, \boldsymbol{w})|_{a=\pi(s,\boldsymbol{\theta})}\big]$
  - $= \mathbb{E}\big[\nabla_a \hat{q}(s, a, \boldsymbol{w}) \nabla_{\boldsymbol{\theta}} \pi(s, \boldsymbol{\theta})|_{a=\pi(s,\boldsymbol{\theta})}\big]$

Deep Deterministic Policy Gradients

- Action can be a vector
  - Multiple joints on a robot





Deep Deterministic Policy Gradients

- Similar to DQN, utilize
  - Replay buffer
  - Target networks
- Exploration
  - When acting, add Gaussian noise to the policy
  - DQN used ϵ-greedy

Deep Deterministic Policy Gradients

Algorithm 1 DDPG algorithm

Randomly initialize critic network $Q(s, a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ with weights θ^Q and θ^μ.
Initialize target network Q' and μ' with weights $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$
Initialize replay buffer R
for episode = 1, M **do**
  Initialize a random process $\mathcal{N}$ for action exploration
  Receive initial observation state s_1
  for t = 1, T **do**
    Select action $a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$ according to the current policy and exploration noise
    Execute action a_t and observe reward r_t and observe new state s_{t+1}
    Store transition (s_t, a_t, r_t, s_{t+1}) in R
    Sample a random minibatch of N transitions (s_i, a_i, r_i, s_{i+1}) from R
    Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$
    Update critic by minimizing the loss: $L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$
    Update the actor policy using the sampled policy gradient:

$$\nabla_{\theta^\mu} J \approx \frac{1}{N} \sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu} \mu(s|\theta^\mu)|_{s_i}$$

    Update the target networks:

$$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$$

$$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$$

  end for
end for

Summary

- Policy gradient with sampled returns
  - $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \approx \frac{1}{m} \sum_{i=0}^{m} \sum_{t=0}^{T-1} \nabla_{\boldsymbol{\theta}} \ln \pi \left(a_t^{(i)} \middle| s_t^{(i)}, \boldsymbol{\theta} \right) G_t^{(i)}$
- Policy gradient with a baseline
  - $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \approx \frac{1}{m} \sum_{i=0}^{m} \sum_{t=0}^{T-1} \nabla_{\boldsymbol{\theta}} \ln \pi \left(a_t^{(i)} \middle| s_t^{(i)}, \boldsymbol{\theta} \right) (G_t^{(i)} - b)$
  - $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \approx \frac{1}{m} \sum_{i=0}^{m} \sum_{t=0}^{T-1} \nabla_{\boldsymbol{\theta}} \ln \pi \left(a_t^{(i)} \middle| s_t^{(i)}, \boldsymbol{\theta} \right) (G_t^{(i)} - \hat{v}(s, \boldsymbol{w}))$
- Actor-critic
  - $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \approx \frac{1}{m} \sum_{i=0}^{m} \sum_{t=0}^{T-1} \nabla_{\boldsymbol{\theta}} \ln \pi \left(a_t^{(i)} \middle| s_t^{(i)}, \boldsymbol{\theta} \right) \hat{q}_{\pi_\theta}(s, a, \boldsymbol{w})$
- Advantage actor critic
  - $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \approx \frac{1}{m} \sum_{i=0}^{m} \sum_{t=0}^{T-1} \nabla_{\boldsymbol{\theta}} \ln \pi \left(a_t^{(i)} \middle| s_t^{(i)}, \boldsymbol{\theta} \right) (\hat{q}_{\pi_\theta}(s, a, \boldsymbol{w}) - \hat{v}_{\pi_\theta}(s, \boldsymbol{\phi}))$
  - $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \approx \frac{1}{m} \sum_{i=0}^{m} \sum_{t=0}^{T-1} \nabla_{\boldsymbol{\theta}} \ln \pi \left(a_t^{(i)} \middle| s_t^{(i)}, \boldsymbol{\theta} \right) (R_{t+1} + \hat{v}_{\pi_\theta}(s', \boldsymbol{\phi}) - \hat{v}_{\pi_\theta}(s, \boldsymbol{\phi}))$
- Deterministic policy gradients
  - $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \approx \mathbb{E}\left[\nabla_{\boldsymbol{\theta}} \hat{q}(s, a, \boldsymbol{w}) |_{a=\pi(s,\boldsymbol{\theta})} \right] = \mathbb{E}\left[\nabla_a \hat{q}(s, a, \boldsymbol{w}) \nabla_{\boldsymbol{\theta}} \pi(s, \boldsymbol{\theta}) |_{a=\pi(s,\boldsymbol{\theta})} \right]$