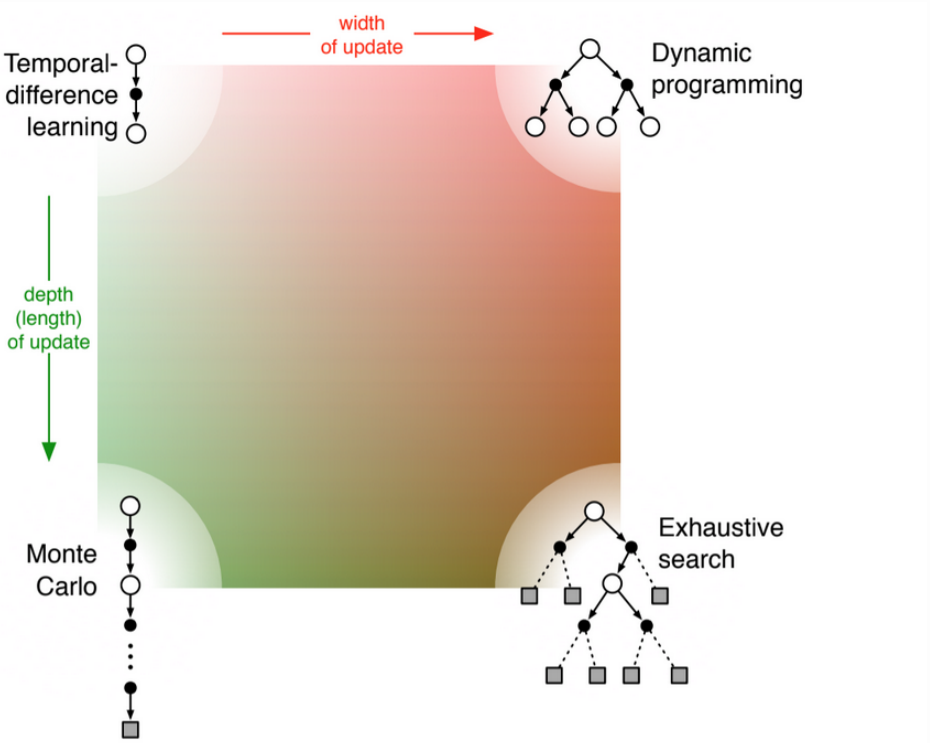
REINFORCEMENT LEARNING: THEORY AND PRACTICE

Ch. 13: Policy Gradient Methods

Profs. Amy Zhang and Peter Stone

Previously

Ch 3-12: State and state-action value function learning methods



Policy Gradient Methods

Why do we want to parameterize a policy directly?

Policy Gradient Methods

Why do we want to parameterize a policy directly?

- To handle continuous actions

Why is this hard for Q learning?

Policy Gradient Methods

Why do we want to parameterize a policy directly?

- To handle continuous actions

Why is this hard for Q learning? Max operator

Policy Gradient Methods

Why do we want to parameterize a policy directly?

- Continuous actions
- Policy simpler than value function
-

Policy Gradient Methods

Why do we want to parameterize a policy directly?

- Continuous actions
- Policy simpler than value function
- Scales with policy complexity, not size of state space
-

Policy Gradient Methods

Why do we want to parameterize a policy directly?

- Continuous actions
- Policy simpler than value function
- Scales with policy complexity, not size of state space
- More robust to poor features (not trying to get exact values right, just ordering of actions -- allocates its power to optimize performance directly, not value representation)

Policy Gradient Methods

Why do we want to parameterize a policy directly?

- Continuous actions
- Policy simpler than value function
- Scales with policy complexity, not size of state space
- More robust to poor features (not trying to get exact values right, just ordering of actions -- allocates its power to optimize performance directly, not value representation)
- Allows for stochastic policies: great for function approximation and partial observability (e.g. bluffing in poker or not knowing which way is exit).

Reading Responses

Neil Roy
Section 13.1 mentions that one advantage of soft-max in action preferences is that it can approach a deterministic policy, which epsilon greedy action selection or soft-max distribution based action-values cannot do. I am a little confused on why action preferences are able to do this while action values are not. It mentions that action preferences do not approach specific values, they are instead driven to produce the optimal stochastic policy. I guess I was a little confused on how the action preferences are driven towards the optimal policy.

Cody Rushing
- Example 13.1 highlights how soft-max action preferences are better because "it can learn a specific probability with which to select right." Could we still approximate this with e-greedy policies by fine-tuning the hyperparameter for e to maximize performance?

Policy as softmax over preferences

$$\pi(a|s,\boldsymbol{\theta}) = \frac{\exp(h(s,a,\boldsymbol{\theta}))}{\sum_b \exp(h(s,b,\boldsymbol{\theta}))}$$

No constraint over what h can be. If h goes to infinity for (s, a), policy becomes deterministic.

If values don't go to infinity, softmax over values can't lead to deterministic policies.

Eps-greedy always takes a random action with eps probability. You could anneal eps, but becomes another design choice.

Policy Gradient Methods

Downsides to policy gradient methods?

Policy Gradient Methods

Downsides to policy gradient methods?

- higher variance without value function to bootstrap
- gradient methods usually locally optimal (except special cases)

Policy Gradient Methods

Consider methods that learn a parameterized policy that can select actions without consulting a value function.

$$\pi(a|s,\boldsymbol{\theta}) = \Pr\{A_t = a \mid S_t = s, \boldsymbol{\theta}_t = \boldsymbol{\theta}\}$$

Want to perform gradient ascent to maximize a scalar performance measure J:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha\widehat{\nabla J(\boldsymbol{\theta}_t)}$$

Policy Gradient Theorem

Theoretical benefits to policy gradient methods: We will show we can perform gradient ascent on the following objective:

$$J(\boldsymbol{\theta}) \doteq v_{\pi_{\boldsymbol{\theta}}}(s_0)$$

Intuition: With continuous policy parameterization the action probability change smoothly as a function of the learned parameter.

Why does this not apply to value functions?

Policy Gradient Theorem

Theoretical benefits to policy gradient methods:

Intuition: With continuous policy parameterization the action probability change smoothly as a function of the learned parameter.

Why does this not apply to value functions?
An arbitrarily small change in the estimated action values can result in a different action having the maximal value.

From lecture 2!

Gradient Bandits: Arm Preferences

$$\Pr\{A_t = a\} \doteq \frac{e^{H_t(a)}}{\sum_{b=1}^{k} e^{H_t(b)}} \doteq \pi_t(a)$$

From lecture 2!

Gradient Bandits: Arm Preferences

$$\Pr\{A_t = a\} \doteq \frac{e^{H_t(a)}}{\sum_{b=1}^{k} e^{H_t(b)}} \doteq \pi_t(a)$$

Differentiable

From lecture 2!

Gradient Bandits: Arm Preferences

$$\Pr\{A_t = a\} \doteq \frac{e^{H_t(a)}}{\sum_{b=1}^{k} e^{H_t(b)}} \doteq \pi_t(a)$$

Differentiable

$$H_{t+1}(A_t) \doteq H_t(A_t) + \alpha\big(R_t - \bar{R}_t\big)\big(1 - \pi_t(A_t)\big), \qquad \text{and}$$
$$H_{t+1}(a) \doteq H_t(a) - \alpha\big(R_t - \bar{R}_t\big)\pi_t(a), \qquad \text{for all } a \neq A_t$$

Updates can be high variance

From lecture 2!

Exact Gradient Ascent

$$H_{t+1}(a) \doteq H_t(a) + \alpha \frac{\partial \mathbb{E}[R_t]}{\partial H_t(a)}$$

Where expected reward is $\mathbb{E}[R_t] = \sum_x \pi_t(x) q_*(x)$

Just a scalar per arm. No states!

In full RL setting, policy influences future states.

Policy Gradient Theorem

$$\nabla J(\boldsymbol{\theta}) \propto \sum_s \mu(s) \sum_a q_\pi(s,a) \nabla_{\boldsymbol{\theta}} \pi(a|s,\boldsymbol{\theta})$$

Exact formula for how performance is affected by the policy parameter.
Does not involve derivatives of the state distribution

Proof of policy gradient theorem

$$
\begin{aligned}
\nabla v_\pi(s) &= \nabla \left[\sum_a \pi(a|s) q_\pi(s,a) \right], \quad \text{for all } s \in \mathcal{S} && \text{(Exercise 3.18)} \\
&= \sum_a \Big[\nabla\pi(a|s) q_\pi(s,a) + \pi(a|s) \nabla q_\pi(s,a) \Big] && \text{(product rule of calculus)} \\
&= \sum_a \Big[\nabla\pi(a|s) q_\pi(s,a) + \pi(a|s) \nabla \sum_{s',r} p(s',r|s,a) \big(r + v_\pi(s')\big) \Big] \\
& && \text{(Exercise 3.19 and Equation 3.2)} \\
&= \sum_a \Big[\nabla\pi(a|s) q_\pi(s,a) + \pi(a|s) \sum_{s'} p(s'|s,a) \nabla v_\pi(s') \Big] && \text{(Eq. 3.4)} \\
&= \sum_a \Big[\nabla\pi(a|s) q_\pi(s,a) + \pi(a|s) \sum_{s'} p(s'|s,a) && \text{(unrolling)} \\
&\qquad \sum_{a'} \big[\nabla\pi(a'|s') q_\pi(s',a') + \pi(a'|s') \sum_{s''} p(s''|s',a') \nabla v_\pi(s'') \big] \Big] \\
&= \sum_{x \in \mathcal{S}} \sum_{k=0}^{\infty} \Pr(s \to x, k, \pi) \sum_a \nabla\pi(a|x) q_\pi(x,a),
\end{aligned}
$$

after repeated unrolling, where $\Pr(s \to x, k, \pi)$ is the probability of transitioning from state s to state x in k steps under policy π. It is then immediate that

$$
\begin{aligned}
\nabla J(\boldsymbol{\theta}) &= \nabla v_\pi(s_0) \\
&= \sum_s \left(\sum_{k=0}^{\infty} \Pr(s_0 \to s, k, \pi) \right) \sum_a \nabla\pi(a|s) q_\pi(s,a) \\
&= \sum_s \eta(s) \sum_a \nabla\pi(a|s) q_\pi(s,a) && \text{(box page 199)} \\
&= \sum_{s'} \eta(s') \sum_s \frac{\eta(s)}{\sum_{s'} \eta(s')} \sum_a \nabla\pi(a|s) q_\pi(s,a) \\
&= \sum_{s'} \eta(s') \sum_s \mu(s) \sum_a \nabla\pi(a|s) q_\pi(s,a) && \text{(Eq. 9.3)} \\
&\propto \sum_s \mu(s) \sum_a \nabla\pi(a|s) q_\pi(s,a) && \text{(Q.E.D.)}
\end{aligned}
$$

Proof of policy gradient theorem

$$
\begin{aligned}
\nabla v_\pi(s) &= \nabla \left[\sum_a \pi(a|s) q_\pi(s,a) \right], \quad \text{for all } s \in \mathcal{S} && \text{(Exercise 3.18)} \\
&= \sum_a \Big[\nabla\pi(a|s) q_\pi(s,a) + \pi(a|s) \nabla q_\pi(s,a) \Big] && \text{(product rule of calculus)} \\
&= \sum_a \Big[\nabla\pi(a|s) q_\pi(s,a) + \pi(a|s) \nabla \sum_{s',r} p(s',r|s,a)\big(r + v_\pi(s')\big) \Big] \\
& && \text{(Exercise 3.19 and Equation 3.2)} \\
&= \sum_a \Big[\nabla\pi(a|s) q_\pi(s,a) + \pi(a|s) \sum_{s'} p(s'|s,a) \nabla v_\pi(s') \Big] && \text{(Eq. 3.4)} \\
&= \sum_a \Big[\nabla\pi(a|s) q_\pi(s,a) + \pi(a|s) \sum_{s'} p(s'|s,a) && \text{(unrolling)} \\
&\qquad \sum_{a'} \big[\nabla\pi(a'|s') q_\pi(s',a') + \pi(a'|s') \sum_{s''} p(s''|s',a') \nabla v_\pi(s'') \big] \Big] \\
&= \sum_{x \in \mathcal{S}} \sum_{k=0}^{\infty} \Pr(s \to x, k, \pi) \sum_a \nabla\pi(a|x) q_\pi(x,a),
\end{aligned}
$$

Marginalize R, push in gradient

Dynamics + Reward constant w.r.t. θ

after repeated unrolling, where $\Pr(s \to x, k, \pi)$ is the probability of transitioning from state s to state x in k steps under policy π. It is then immediate that

$$
\begin{aligned}
\nabla J(\boldsymbol{\theta}) &= \nabla v_\pi(s_0) \\
&= \sum_s \left(\sum_{k=0}^{\infty} \Pr(s_0 \to s, k, \pi) \right) \sum_a \nabla\pi(a|s) q_\pi(s,a) \\
&= \sum_s \eta(s) \sum_a \nabla\pi(a|s) q_\pi(s,a) && \text{(box page 199)} \\
&= \sum_{s'} \eta(s') \sum_s \frac{\eta(s)}{\sum_{s'} \eta(s')} \sum_a \nabla\pi(a|s) q_\pi(s,a) \\
&= \sum_{s'} \eta(s') \sum_s \mu(s) \sum_a \nabla\pi(a|s) q_\pi(s,a) && \text{(Eq. 9.3)} \\
&\propto \sum_s \mu(s) \sum_a \nabla\pi(a|s) q_\pi(s,a) && \text{(Q.E.D.)}
\end{aligned}
$$

Proof of policy gradient theorem

$$
\begin{aligned}
\nabla v_\pi(s) &= \nabla \left[\sum_a \pi(a|s) q_\pi(s,a) \right], \quad \text{for all } s \in \mathcal{S} && \text{(Exercise 3.18)} \\
&= \sum_a \Big[\nabla\pi(a|s) q_\pi(s,a) + \pi(a|s) \nabla q_\pi(s,a) \Big] && \text{(product rule of calculus)} \\
&= \sum_a \Big[\nabla\pi(a|s) q_\pi(s,a) + \pi(a|s) \nabla \sum_{s',r} p(s',r|s,a)\big(r + v_\pi(s')\big) \Big] \\
& && \text{(Exercise 3.19 and Equation 3.2)} \\
&= \sum_a \Big[\nabla\pi(a|s) q_\pi(s,a) + \pi(a|s) \sum_{s'} p(s'|s,a) \nabla v_\pi(s') \Big] && \text{(Eq. 3.4)} \\
&= \sum_a \Big[\nabla\pi(a|s) q_\pi(s,a) + \pi(a|s) \sum_{s'} p(s'|s,a) && \text{(unrolling)} \\
&\qquad \sum_{a'} \big[\nabla\pi(a'|s') q_\pi(s',a') + \pi(a'|s') \sum_{s''} p(s''|s',a') \nabla v_\pi(s'') \big] \Big] \\
&= \sum_{x \in \mathcal{S}} \sum_{k=0}^{\infty} \Pr(s \to x, k, \pi) \sum_a \nabla\pi(a|x) q_\pi(x,a),
\end{aligned}
$$

after repeated unrolling, where $\Pr(s \to x, k, \pi)$ is the probability of transitioning from state s to state x in k steps under policy π. It is then immediate that

$$
\begin{aligned}
\nabla J(\boldsymbol{\theta}) &= \nabla v_\pi(s_0) \\
&= \sum_s \left(\sum_{k=0}^{\infty} \Pr(s_0 \to s, k, \pi) \right) \sum_a \nabla\pi(a|s) q_\pi(s,a) \\
&= \sum_s \eta(s) \sum_a \nabla\pi(a|s) q_\pi(s,a) && \text{(box page 199)} \\
&= \sum_{s'} \eta(s') \sum_s \frac{\eta(s)}{\sum_{s'} \eta(s')} \sum_a \nabla\pi(a|s) q_\pi(s,a) \\
&= \sum_{s'} \eta(s') \sum_s \mu(s) \sum_a \nabla\pi(a|s) q_\pi(s,a) && \text{(Eq. 9.3)} \\
&\propto \sum_s \mu(s) \sum_a \nabla\pi(a|s) q_\pi(s,a) && \text{(Q.E.D.)}
\end{aligned}
$$

Marginalize R, push in gradient

Dynamics + Reward constant w.r.t. θ

Expanding $V_\pi(s')$ creates deeply nested computation:

At every step, compute every state you could get to from every state you could have been in

↓

Transform into simple sum over time steps and states:

What is total prob of being at each state at each time step?

Proof of policy gradient theorem

$$
\begin{aligned}
\nabla v_\pi(s) &= \nabla \left[\sum_a \pi(a|s) q_\pi(s,a) \right], \quad \text{for all } s \in \mathcal{S} && \text{(Exercise 3.18)} \\
&= \sum_a \Big[\nabla\pi(a|s) q_\pi(s,a) + \pi(a|s) \nabla q_\pi(s,a) \Big] && \text{(product rule of calculus)} \\
&= \sum_a \Big[\nabla\pi(a|s) q_\pi(s,a) + \pi(a|s) \nabla \sum_{s',r} p(s',r|s,a)\big(r + v_\pi(s')\big) \Big] \\
& && \text{(Exercise 3.19 and Equation 3.2)} \\
&= \sum_a \Big[\nabla\pi(a|s) q_\pi(s,a) + \pi(a|s) \sum_{s'} p(s'|s,a) \nabla v_\pi(s') \Big] && \text{(Eq. 3.4)} \\
&= \sum_a \Big[\nabla\pi(a|s) q_\pi(s,a) + \pi(a|s) \sum_{s'} p(s'|s,a) \\
&\qquad \sum_{a'} \big[\nabla\pi(a'|s') q_\pi(s',a') + \pi(a'|s') \sum_{s''} p(s''|s',a') \nabla v_\pi(s'') \big] \Big] && \text{(unrolling)} \\
&= \sum_{x \in \mathcal{S}} \sum_{k=0}^{\infty} \Pr(s \to x, k, \pi) \sum_a \nabla\pi(a|x) q_\pi(x,a),
\end{aligned}
$$

Marginalize R, push in gradient

Dynamics + Reward constant w.r.t. θ

Expanding $V_\pi(s')$ creates deeply nested computation:

At every step, compute every state you could get to from every state you could have been in

↓

Transform into simple sum over time steps and states:

What is total prob of being at each state at each time step?

after repeated unrolling, where $\Pr(s \to x, k, \pi)$ is the probability of transitioning from state s to state x in k steps under policy π. It is then immediate that

$$
\begin{aligned}
\nabla J(\boldsymbol{\theta}) &= \nabla v_\pi(s_0) \\
&= \sum_s \left(\sum_{k=0}^{\infty} \Pr(s_0 \to s, k, \pi) \right) \sum_a \nabla\pi(a|s) q_\pi(s,a) \\
&= \sum_s \eta(s) \sum_a \nabla\pi(a|s) q_\pi(s,a) && \text{(box page 199)} \\
&= \sum_{s'} \eta(s') \sum_s \frac{\eta(s)}{\sum_{s'} \eta(s')} \sum_a \nabla\pi(a|s) q_\pi(s,a) \\
&= \sum_{s'} \eta(s') \sum_s \mu(s) \sum_a \nabla\pi(a|s) q_\pi(s,a) && \text{(Eq. 9.3)} \\
&\propto \sum_s \mu(s) \sum_a \nabla\pi(a|s) q_\pi(s,a) && \text{(Q.E.D.)}
\end{aligned}
$$

unnormalized steady-state prob of s

normalized version

Reading Responses

Alekhya Kuchimanchi
Why does the action probabilities changing as a function of the learned parameter lead to stronger convergence guarantees? (page 324)

Abhinav Peri
With the policy gradient, it seems that as long as we move in this direction in parameter space, we can guarantee improvement in the policy. Does this give us a replacement for the policy improvement theorem that we lost when switching to function approximation?

Reading Responses

Alekhya Kuchimanchi
Why does the action probabilities changing as a function of the learned parameter lead to stronger convergence guarantees? (page 324)

Abhinav Peri
With the policy gradient, it seems that as long as we move in this direction in parameter space, we can guarantee improvement in the policy. Does this give us a replacement for the policy improvement theorem that we lost when switching to function approximation?

We're relying on convergence guarantees of SGD to explain why policy gradient methods converge by showing that policy gradient corresponds to performing gradient ascent on the objective of maximizing return.

REINFORCE

$$\nabla J(\boldsymbol{\theta}) \propto \sum_s \mu(s) \sum_a q_\pi(s,a) \nabla \pi(a|s,\boldsymbol{\theta})$$
$$= \mathbb{E}_\pi \left[\sum_a q_\pi(S_t,a) \nabla \pi(a|S_t,\boldsymbol{\theta}) \right].$$

$$\boldsymbol{\theta}_{t+1} \doteq \boldsymbol{\theta}_t + \alpha \sum_a \hat{q}(S_t,a,\mathbf{w}) \nabla \pi(a|S_t,\boldsymbol{\theta})$$

All actions

Q approx, not a sample return

REINFORCE

$$\nabla J(\boldsymbol{\theta}) \propto \sum_s \mu(s) \sum_a q_\pi(s,a) \nabla \pi(a|s,\boldsymbol{\theta})$$
$$= \mathbb{E}_\pi \left[\sum_a q_\pi(S_t,a) \nabla \pi(a|S_t,\boldsymbol{\theta}) \right].$$

$$\boldsymbol{\theta}_{t+1} \doteq \boldsymbol{\theta}_t + \alpha \sum_a \hat{q}(S_t,a,\mathbf{w}) \nabla \pi(a|S_t,\boldsymbol{\theta})$$

$$\nabla J(\boldsymbol{\theta}) = \mathbb{E}_\pi \left[\sum_a \pi(a|S_t,\boldsymbol{\theta}) q_\pi(S_t,a) \frac{\nabla \pi(a|S_t,\boldsymbol{\theta})}{\pi(a|S_t,\boldsymbol{\theta})} \right]$$
$$= \mathbb{E}_\pi \left[q_\pi(S_t,A_t) \frac{\nabla \pi(A_t|S_t,\boldsymbol{\theta})}{\pi(A_t|S_t,\boldsymbol{\theta})} \right] \qquad \text{(replacing } a \text{ by the sample } A_t \sim \pi)$$
$$= \mathbb{E}_\pi \left[G_t \frac{\nabla \pi(A_t|S_t,\boldsymbol{\theta})}{\pi(A_t|S_t,\boldsymbol{\theta})} \right], \qquad \text{(because } \mathbb{E}_\pi[G_t|S_t,A_t] = q_\pi(S_t,A_t))$$

REINFORCE

$$\nabla J(\boldsymbol{\theta}) \propto \sum_s \mu(s) \sum_a q_\pi(s,a) \nabla \pi(a|s,\boldsymbol{\theta})$$
$$= \mathbb{E}_\pi \left[\sum_a q_\pi(S_t,a) \nabla \pi(a|S_t,\boldsymbol{\theta}) \right].$$

$$\boldsymbol{\theta}_{t+1} \doteq \boldsymbol{\theta}_t + \alpha \sum_a \hat{q}(S_t,a,\mathbf{w}) \nabla \pi(a|S_t,\boldsymbol{\theta})$$

$$\begin{aligned}
\nabla J(\boldsymbol{\theta}) &= \mathbb{E}_\pi \left[\sum_a \pi(a|S_t,\boldsymbol{\theta}) q_\pi(S_t,a) \frac{\nabla \pi(a|S_t,\boldsymbol{\theta})}{\pi(a|S_t,\boldsymbol{\theta})} \right] \\
&= \mathbb{E}_\pi \left[q_\pi(S_t,A_t) \frac{\nabla \pi(A_t|S_t,\boldsymbol{\theta})}{\pi(A_t|S_t,\boldsymbol{\theta})} \right] && \text{(replacing } a \text{ by the sample } A_t \sim \pi) \\
&= \mathbb{E}_\pi \left[G_t \frac{\nabla \pi(A_t|S_t,\boldsymbol{\theta})}{\pi(A_t|S_t,\boldsymbol{\theta})} \right], && \text{(because } \mathbb{E}_\pi[G_t|S_t,A_t] = q_\pi(S_t,A_t))
\end{aligned}$$

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 $t = 0, 1, \ldots, T-1$:
    $G \leftarrow \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$ $\qquad (G_t)$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \gamma^t G \nabla \ln \pi(A_t|S_t,\boldsymbol{\theta})$

REINFORCE

Pros and cons?

REINFORCE

Pros and cons?

SGD-based theoretical convergence properties: assures an improvement in expected performance for sufficiently small learning rate, and convergence to a local optimum under standard stochastic approximation conditions for decreasing learning rate.

However, as a Monte Carlo method REINFORCE may be of high variance and thus produce slow learning.

Gradient Bandits + Baseline

$$H_{t+1}(a) \doteq H_t(a) + \alpha \frac{\partial \mathbb{E}\left[R_t\right]}{\partial H_t(a)}$$

$$\begin{aligned}
\frac{\partial \mathbb{E}[R_t]}{\partial H_t(a)} &= \frac{\partial}{\partial H_t(a)} \left[\sum_x \pi_t(x) q_*(x) \right] \\
&= \sum_x q_*(x) \frac{\partial \pi_t(x)}{\partial H_t(a)} \\
&= \sum_x \left(q_*(x) - B_t \right) \frac{\partial \pi_t(x)}{\partial H_t(a)},
\end{aligned}$$

Mean of Samples

Expectation zero

Gradient Bandits + Baseline

$$H_{t+1}(a) \doteq H_t(a) + \alpha \frac{\partial \mathbb{E}\left[R_t\right]}{\partial H_t(a)}$$

$$\begin{aligned}
\frac{\partial \mathbb{E}[R_t]}{\partial H_t(a)} &= \frac{\partial}{\partial H_t(a)} \left[\sum_x \pi_t(x) q_*(x) \right] \\
&= \sum_x q_*(x) \frac{\partial \pi_t(x)}{\partial H_t(a)} \\
&= \sum_x \left(q_*(x) - B_t \right) \frac{\partial \pi_t(x)}{\partial H_t(a)},
\end{aligned}$$

Mean of Samples

Expectation Zero

REINFORCE + Baseline

$$\nabla J(\boldsymbol{\theta}) \propto \sum_s \mu(s) \sum_a q_\pi(s,a) \nabla \pi(a|s,\boldsymbol{\theta})$$

$$\nabla J(\boldsymbol{\theta}) \propto \sum_s \mu(s) \sum_a \Big(q_\pi(s,a) - b(s) \Big) \nabla \pi(a|s,\boldsymbol{\theta}).$$

$$\sum_a b(s) \nabla \pi(a|s,\boldsymbol{\theta}) \;=\; b(s) \nabla \sum_a \pi(a|s,\boldsymbol{\theta}) \;=\; b(s) \nabla 1 \;=\; 0.$$

$$\boldsymbol{\theta}_{t+1} \doteq \boldsymbol{\theta}_t + \alpha \Big(G_t - b(S_t) \Big) \frac{\nabla \pi(A_t|S_t,\boldsymbol{\theta}_t)}{\pi(A_t|S_t,\boldsymbol{\theta}_t)}.$$

$\hat{v}(S_t)$

Reading Responses

Abhi Sridhar
Discuss how introducing a baseline can impact the performance of the REINFORCE algorithm. What are the criteria for choosing an effective baseline?

Reading Responses

Abhi Sridhar

Discuss how introducing a baseline can impact the performance of the REINFORCE algorithm. What are the criteria for choosing an effective baseline?

A baseline can be any scalar, as long as it doesn't depend on the action.

Goals: reduce its high variance but not change its direction

Reading Responses

Jackson Paul
Is there any additional intuition behind why a baseline would be useful for computation? It seems intuitive that maximizing J(theta) would also (in expectation) minimize J(theta) - V(s) since they ultimately have the same goal.

Reading Responses

Jackson Paul

Is there any additional intuition behind why a baseline would be useful for computation? It seems intuitive that maximizing J(theta) would also (in expectation) minimize J(theta) - V(s) since they ultimately have the same goal.

They do! The point of the baseline is not to modify the convergence point, but to lower the variance of the updates to get there.

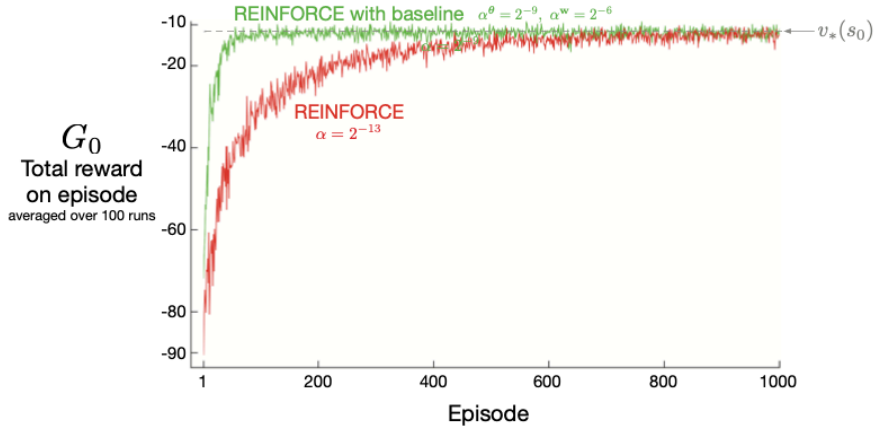


Figure 13.2: Adding a baseline to REINFORCE can make it learn much faster, as illustrated here on the short-corridor gridworld (Example 13.1). The step size used here for plain REINFORCE is that at which it performs best (to the nearest power of two; see Figure 13.1).

From lecture 2!

Why does the variance of the gradient matter?

Theory: upper bounds on convergence rate of SGD are directly related to the variance of gradient estimates

Intuition: variance causes "overshooting" that destabilizes learning

$$H_{t+1}(A_t) \doteq H_t(A_t) + \alpha\big(R_t - \bar{R}_t\big)\big(1 - \pi_t(A_t)\big), \qquad \text{and}$$
$$H_{t+1}(a) \doteq H_t(a) - \alpha\big(R_t - \bar{R}_t\big)\pi_t(a), \qquad \text{for all } a \neq A_t$$

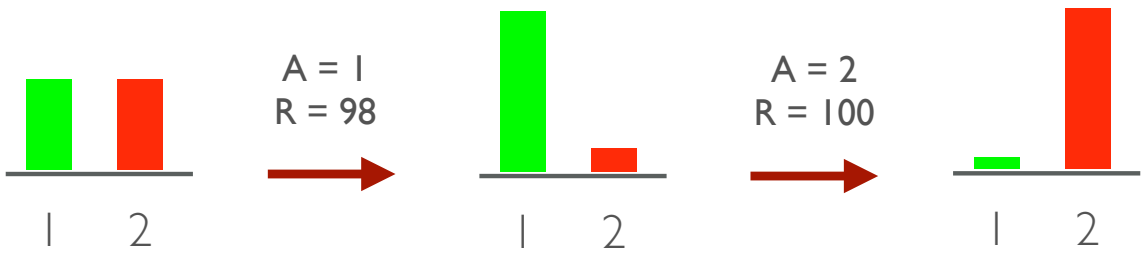
From lecture 2!

Why does the variance of the gradient matter?

Theory: upper bounds on convergence rate of SGD are directly related to the variance of gradient estimates

Intuition: variance causes "overshooting" that destabilizes learning

$$H_{t+1}(A_t) \doteq H_t(A_t) + \alpha\big(R_t - \bar{R}_t\big)\big(1 - \pi_t(A_t)\big), \qquad \text{and}$$
$$H_{t+1}(a) \doteq H_t(a) - \alpha\big(R_t - \bar{R}_t\big)\pi_t(a), \qquad \text{for all } a \neq A_t$$



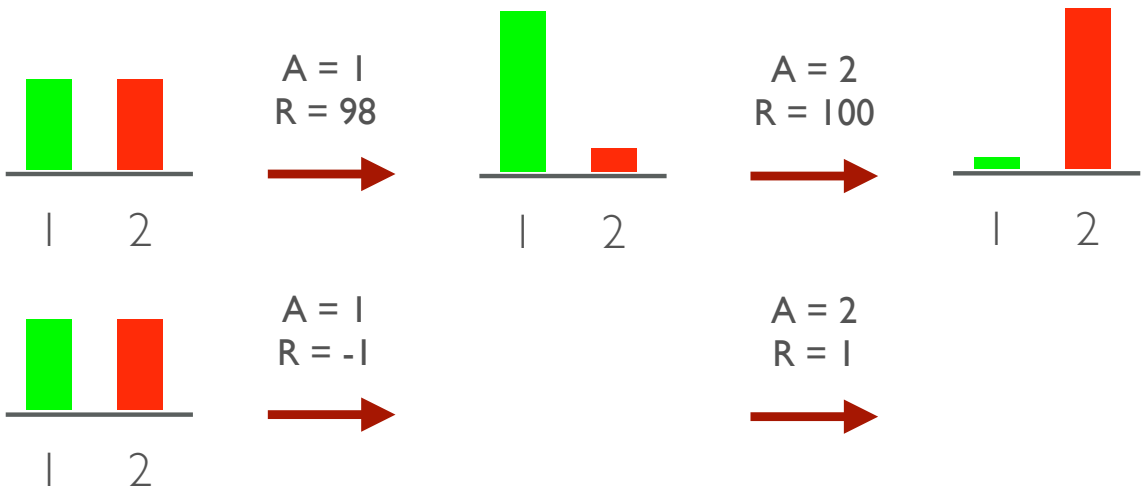
From lecture 2!

Why does the variance of the gradient matter?

Theory: upper bounds on convergence rate of SGD are directly related to the variance of gradient estimates

Intuition: variance causes "overshooting" that destabilizes learning

$$H_{t+1}(A_t) \doteq H_t(A_t) + \alpha\big(R_t - \bar{R}_t\big)\big(1 - \pi_t(A_t)\big), \qquad \text{and}$$
$$H_{t+1}(a) \doteq H_t(a) - \alpha\big(R_t - \bar{R}_t\big)\pi_t(a), \qquad \text{for all } a \neq A_t$$



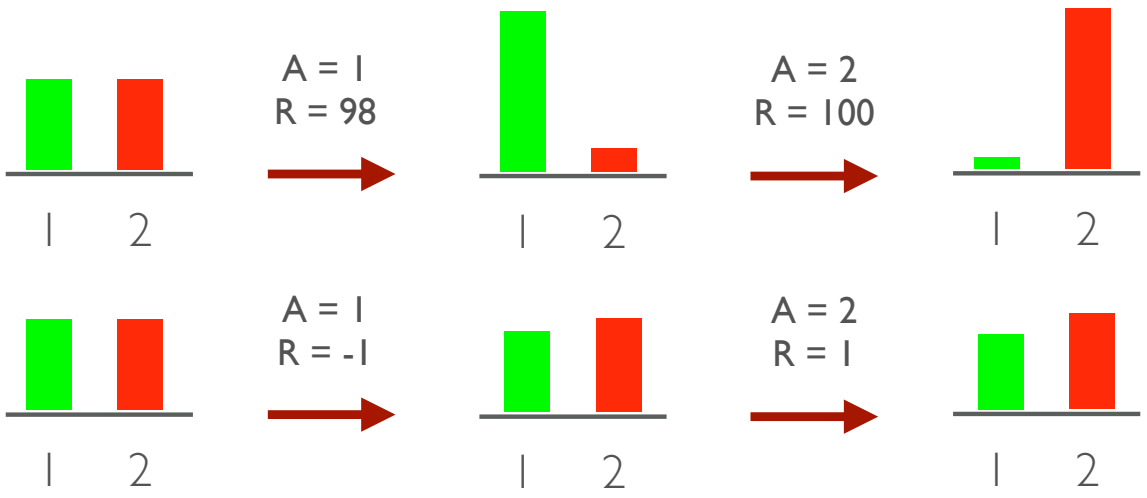
From lecture 2!

Why does the variance of the gradient matter?

Theory: upper bounds on convergence rate of SGD are directly related to the variance of gradient estimates

Intuition: variance causes "overshooting" that destabilizes learning

$$H_{t+1}(A_t) \doteq H_t(A_t) + \alpha\big(R_t - \bar{R}_t\big)\big(1 - \pi_t(A_t)\big), \qquad \text{and}$$
$$H_{t+1}(a) \doteq H_t(a) - \alpha\big(R_t - \bar{R}_t\big)\pi_t(a), \qquad \text{for all } a \neq A_t$$



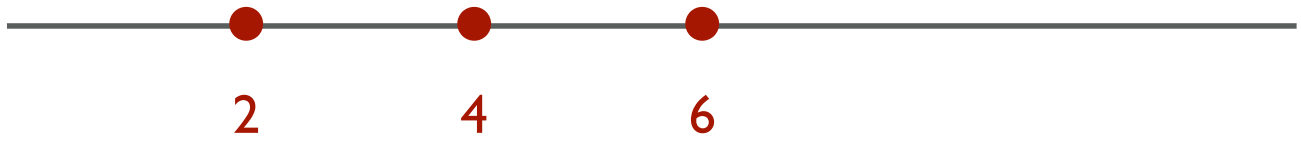
From lecture 2!

Why does baseline reduce variance?

From lecture 2!

Why does baseline reduce variance?

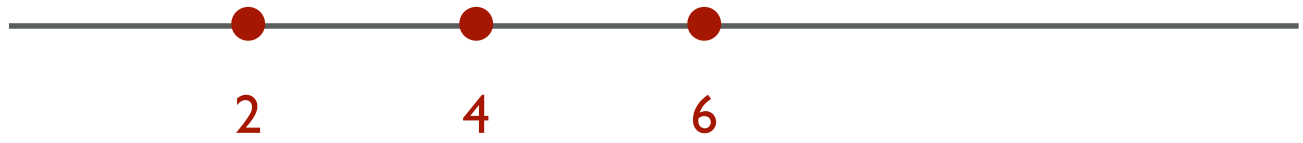
Original "gradients"



From lecture 2!

Why does baseline reduce variance?

Original "gradients"

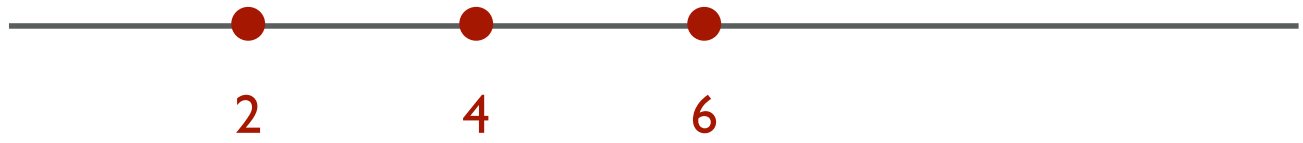


We are NOT subtracting from the gradient

From lecture 2!

Why does baseline reduce variance?

Original "gradients"



We are NOT subtracting from the gradient

We are subtracting from a number that **multiplies** the gradient

$$\frac{\partial \mathbb{E}[R_t]}{\partial H_t(a)} = \sum_x \left(q_*(x) - B_t\right) \frac{\partial \pi_t(x)}{\partial H_t(a)}$$

REINFORCE with Baseline (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 $\hat{v}(s,\mathbf{w})$
Algorithm 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):
  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 $t = 0, 1, \ldots, T-1$:
    $G \leftarrow \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$ $\qquad (G_t)$
    $\delta \leftarrow G - \hat{v}(S_t,\mathbf{w})$
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}} \delta \nabla \hat{v}(S_t,\mathbf{w})$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} \gamma^t \delta \nabla \ln \pi(A_t|S_t,\boldsymbol{\theta})$

REINFORCE with Baseline (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 $\hat{v}(s,\mathbf{w})$
Algorithm 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):
  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 $t = 0, 1, \ldots, T-1$:
    $G \leftarrow \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$ $\qquad (G_t)$
    $\delta \leftarrow G - \hat{v}(S_t,\mathbf{w})$
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}} \delta \nabla \hat{v}(S_t,\mathbf{w})$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} \gamma^t \delta \nabla \ln \pi(A_t|S_t,\boldsymbol{\theta})$

This is Monte Carlo. How do we incorporate TD?

REINFORCE with Baseline (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 $\hat{v}(s,\mathbf{w})$
Algorithm 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):
  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 $t = 0, 1, \ldots, T-1$:
    $G \leftarrow \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$ $\qquad (G_t)$
    $\delta \leftarrow G - \hat{v}(S_t,\mathbf{w})$
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}} \delta \nabla \hat{v}(S_t,\mathbf{w})$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} \gamma^t \delta \nabla \ln \pi(A_t|S_t,\boldsymbol{\theta})$

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
    $\delta \leftarrow R + \gamma \hat{v}(S',\mathbf{w}) - \hat{v}(S,\mathbf{w})$ $\qquad$ (if S' is terminal, then $\hat{v}(S',\mathbf{w}) \doteq 0$)
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}} \delta \nabla \hat{v}(S,\mathbf{w})$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} I \delta \nabla \ln \pi(A|S,\boldsymbol{\theta})$
    $I \leftarrow \gamma I$
    $S \leftarrow S'$

Reading Responses

Yash Saxena
Why is REINFORCE with Baseline not considered an actor-critic method? The algorithm learns an approximation of the policy and value function, so why wouldn't it fall under the actor-critic umbrella?

Reading Responses

Yash Saxena
Why is REINFORCE with Baseline not considered an actor-critic method? The algorithm learns an approximation of the policy and value function, so why wouldn't it fall under the actor-critic umbrella?
Actor-critic methods are temporal difference (TD) learning methods that represent the policy function independent of the value function. REINFORCE with Baseline does use a critic, but not TD.

Actor only

Critic only

Actor - Critic

Actor only

- policy search
- Directly parameterized policy
- No value functions (except baseline in REINFORCE)
- Continuous actions natural to represent
- High variance, No bootstrapping
- Scales w/ policy complexity, not size of state space

Critic only

Actor - Critic

Actor only

- policy search
- Directly parameterized policy
- No value functions (except baseline in REINFORCE)
- Continuous actions natural to represent
- High variance, No bootstrapping
- Scales w/ policy complexity, not size of state space

Critic only

- value function methods
- Indirect policy via VF
- Discrete actions only
- Lower variance, bootstrapping
- Scales with size of state space

Actor - Critic

Actor only

- policy search
- Directly parameterized policy
- No value functions (except baseline in REINFORCE)
- Continuous actions natural to represent
- High variance, No bootstrapping
- Scales w/ policy complexity, not size of state space

Critic only

- value function methods
- Indirect policy via VF
- Discrete actions only
- Lower variance, bootstrapping
- Scales with size of state space

Actor - Critic

- Policy Search + value function
- Benefits of both!
- Continuous actions
- Bootstrapping
- Scales primarily with policy complexity

Actor only

- policy search
- Directly parameterized policy
- No value functions (except baseline in REINFORCE)
- Continuous actions natural to represent
- High variance, No bootstrapping
- Scales w/ policy complexity, not size of state space

Critic only

- value function methods
- Indirect policy via VF
- Discrete actions only
- Lower variance, bootstrapping
- Scales with size of state space

Actor-Critic

- Policy Search + value function
- Benefits of both!
- Continuous actions
- Bootstrapping
- Scales primarily with policy complexity

Many of most popular contempory methods are A-C:

- Proximal Policy Optimization
- A3C
- Soft Actor Critic
- DDPG

Reading Responses

Ian Symsmith

While you would run into similar problems as was described in earlier chapters, could you use policy gradient methods with off-policy learning as well?

Yes! Many actor-critic methods are off-policy.

Off-Policy Actor-Critic

Thomas Degris THOMAS.DEGRIS@INRIA.FR
Flowers Team, INRIA, Talence, ENSTA-ParisTech, Paris, France

Martha White WHITEM@CS.UALBERTA.CA
Richard S. Sutton SUTTON@CS.UALBERTA.CA
RLAI Laboratory, Department of Computing Science, University of Alberta, Edmonton, Canada

$$
\begin{aligned}
\nabla_\theta J(\theta) &= \nabla_\theta \mathbb{E}_{s\sim d^\beta}\Big[\sum_{a\in\mathcal{A}} Q^\pi(s,a)\pi_\theta(a|s)\Big] \\
&= \mathbb{E}_{s\sim d^\beta}\Big[\sum_{a\in\mathcal{A}} \big(Q^\pi(s,a)\nabla_\theta\pi_\theta(a|s) + \pi_\theta(a|s)\nabla_\theta Q^\pi(s,a)\big)\Big] \\
&\overset{(i)}{\approx} \mathbb{E}_{s\sim d^\beta}\Big[\sum_{a\in\mathcal{A}} Q^\pi(s,a)\nabla_\theta\pi_\theta(a|s)\Big] \\
&= \mathbb{E}_{s\sim d^\beta}\Big[\sum_{a\in\mathcal{A}} \beta(a|s)\frac{\pi_\theta(a|s)}{\beta(a|s)} Q^\pi(s,a)\frac{\nabla_\theta\pi_\theta(a|s)}{\pi_\theta(a|s)}\Big] \\
&= \mathbb{E}_\beta\Big[\frac{\pi_\theta(a|s)}{\beta(a|s)} Q^\pi(s,a)\nabla_\theta \ln\pi_\theta(a|s)\Big]
\end{aligned}
$$

Reading Responses

Adeline Foote
For continuous actions we can learn probability distributions of the actions. How do we choose how to represent these distributions? Not everything will follow a normal distribution. For example there may be some situations where a bimodal distribution is helpful? Can we add and weight multiple distributions to learn more complex probability distributions?

Anthony Bao
What is the current landscape of using generative models in policy gradient methods for continuing problems? When is it fruitful to learn distributions for action selection using a normalizing flow or diffusion model?

Reading Responses

We can parameterize the policy in many ways:
Gaussians, mixture of Gaussians, diffusion models, SVGD (particle-based method)

IDQL: Implicit Q-Learning as an Actor-Critic Method with Diffusion Policies

Philippe Hansen-Estruch Ilya Kostrikov Michael Janner
Jakub Grudzien Kuba Sergey Levine
UC Berkeley

S2AC: Energy-Based Reinforcement Learning with Stein Soft Actor Critic

Safa Messaoud, Billel Mokeddem, Zhenghai Xue, Linsey Pang, Bo An, Haipeng Chen, Sanjay Chawla

Distributional Policy Optimization: An Alternative Approach for Continuous Control

Chen Tessler*, Guy Tennenholtz* and Shie Mannor

Final Logistics

Next lecture:
Review for midterm!
Reading assignments due **2PM Monday**

No Amy office hours for next three weeks: 3/6, 3/13, and 3/20.

Final Logistics

Final project proposal due at **11:59pm on Thursday, 3/7**

Complete the reading response on Canvas by Monday at 2pm CST.

Complete Homework for Chapter 13 on edx by Friday 11:59 PM CST