Lecture 9: Policy Gradients and Actor Critics

Hado van Hasselt

UCL, 2021

Background reading: Sutton & Barto, 2018, Chapter 13

“Do no solve a more general problem as an intermediate step.”

— Vladimir Vapnik, 1998

If we care about optimal behaviour: why not learn a policy directly?

General overview

- **Model-based RL**
  - + 'Easy' to learn a model (supervised learning)
  - + Learns 'all there is to know' from the data
  - - Uses compute & capacity on irrelevant details
  - - Computing policy (=planning) is non-trivial and expensive (in compute)
- **Value-based RL**
  - + Easy to generate policy (e.g., $\pi(a|s) = \mathcal{I}(a = \operatorname*{argmax}_a q(s, a))$)
  - + Close to true objective
  - + Fairly well-understood, good algorithms exist
  - - Still not the true objective:
    - - May focus capacity on irrelevant details
    - - Small value error can lead to larger policy error
- **Policy-based RL**
  - + Right objective!
  - ◦ More pros and cons on later slide

General overview

Model-based RL **Value-based RL** **Policy-based RL**

- All of these generalise in different ways
- Sometimes learning a model is easier (e.g., simple dynamics)
- Sometimes learning a policy is easier (e.g., “always move forward” is optimal)

Policy-Based Reinforcement Learning

- Previously we approximated paramateric value functions

$$v_{\mathbf{w}}(s) \approx v_\pi(s)$$
$$q_{\mathbf{w}}(s, a) \approx q_\pi(s, a)$$

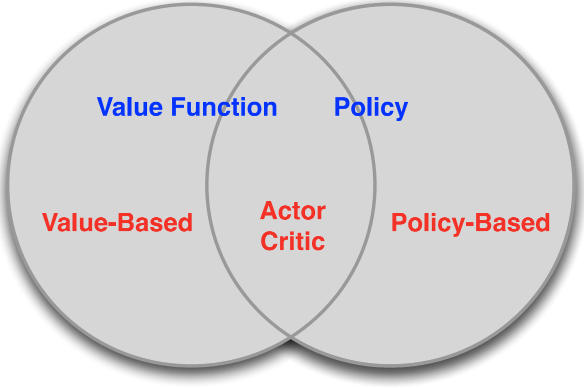
- A policy can be generated from these values (e.g., greedy)
- In this lecture we directly parametrise the **policy** directly

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

- This lecture, we focus on **model-free** reinforcement learning

Value-based and policy-based RL: terminology

- **Value Based**
  - Learn values
  - Implicit policy (e.g. ϵ-greedy)
- **Policy Based**
  - No values
  - Learn policy
- **Actor-Critic**
  - Learn values
  - Learn policy



Advantages and disadvantages of policy-based RL

Advantages:

- True objective
- Easy extended to **high-dimensional** or **continuous** action spaces
- Can learn **stochastic** policies
- Sometimes policies are **simple** while values and models are complex
  - E.g., complicated dynamics, but optimal policy is always "move forward"

Disadvantages:

- Could get stuck in local optima
- Obtained knowledge can be **specific**, does not always generalise well
- Does not necessarily extract all useful information from the data (when used in isolation)

Benefits of stochastic policies

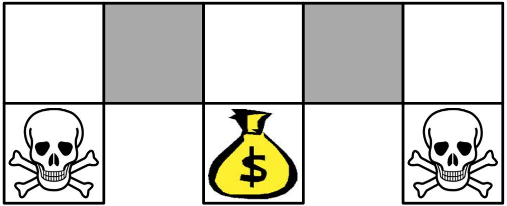
Stochastic policies

Why could we need stochastic policies?

- In MDPs, there is always an optimal **deterministic** policy
- But, most problems are **not fully observable**
  - This is the common case, especially with function approximation
  - The optimal policy may then be stochastic
- Search space is smoother for stochastic policies $\implies$ we can use gradients
- Provides some 'exploration' during learning

Stochastic Policy Example: Aliased Grid World

Example: Aliased Grid World

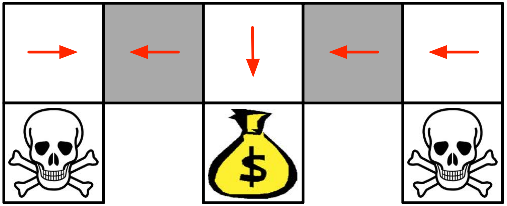


- The grey states look the same
- Consider features:

$$\phi(s) = \overbrace{\begin{pmatrix} \underbrace{1}_{\text{up}} & \underbrace{0}_{\text{right}} & \underbrace{1}_{\text{down}} & \underbrace{0}_{\text{left}} \end{pmatrix}}^{\text{walls=state}}$$

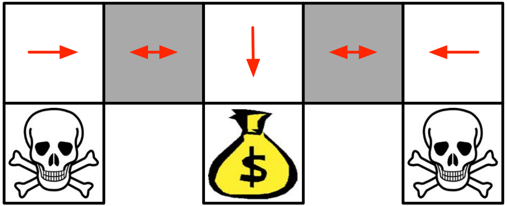
- Compare **deterministic** and **stochastic** policies

Example: Aliased Gridworld



- Under aliasing, an optimal **deterministic** policy will either
  - move left in both grey states (shown by red arrows)
  - or move right in both grey states
- Either way, it can get stuck and never reach the money

Example: Aliased Gridworld



- An optimal **stochastic** policy moves randomly left or right in grey states

$$\pi_{\theta}(\text{right} \mid \text{wall up and down}) = 0.5$$
$$\pi_{\theta}(\text{left} \mid \text{wall up and down}) = 0.5$$

- Will reach the goal state in a few steps with high probability
- Directly learning the policy parameters, we can learn an optimal stochastic policy
- Also when optimal policy does not give equal probability
  (So this differs from random tie-breaking with values.)

Policy Learning Objective

Policy Objective Functions

- Goal: given **policy $\pi_\theta(s, a)$**, find best **parameters θ**
- How do we measure the quality of a policy π_θ?
- In episodic environments we can use the **average total return per episode**
- In continuing environments we can use the **average reward per step**

Policy Objective Functions: Episodic

- **Episodic-return objective**:

$$\begin{aligned}
J_{\mathrm{G}}(\boldsymbol{\theta}) &= \mathbb{E}_{S_0 \sim d_0, \pi_{\boldsymbol{\theta}}}\left[\sum_{t=0}^{\infty} \gamma^t R_{t+1}\right] \\
&= \mathbb{E}_{S_0 \sim d_0, \pi_{\boldsymbol{\theta}}}[G_0] \\
&= \mathbb{E}_{S_0 \sim d_0}[\mathbb{E}_{\pi_{\boldsymbol{\theta}}}[G_t \mid S_t = S_0]] \\
&= \mathbb{E}_{S_0 \sim d_0}[v_{\pi_{\boldsymbol{\theta}}}(S_0)]
\end{aligned}$$

where d_0 is the start-state distribution This objective equals the expected value of the start state

Policy Objective Functions: Average Reward

- **Average-reward objective**

$$
\begin{aligned}
J_{\mathrm{R}}(\boldsymbol{\theta}) &= \mathbb{E}_{\pi_{\boldsymbol{\theta}}}\left[R_{t+1}\right] \\
&= \mathbb{E}_{S_t \sim d_{\pi_{\boldsymbol{\theta}}}}\left[\mathbb{E}_{A_t \sim \pi_{\boldsymbol{\theta}}(S_t)}\left[R_{t+1} \mid S_t\right]\right] \\
&= \sum_s d_{\pi_{\boldsymbol{\theta}}}(s) \sum_a \pi_{\boldsymbol{\theta}}(s, a) \sum_r p(r \mid s, a) r
\end{aligned}
$$

where $d_\pi(s) = p(S_t = s \mid \pi)$ is the probability of being in state s in the long run
Think of it as the ratio of time spent in s under policy π

Policy Gradients

Policy Optimisation

- Policy based reinforcement learning is an **optimization** problem
- Find $\boldsymbol{\theta}$ that maximises $J(\boldsymbol{\theta})$
- We will focus on **stochastic gradient ascent**, which is often quite efficient (and easy to use with deep nets)
- Some approaches do not use gradient
  - Hill climbing / simulated annealing
  - Genetic algorithms / evolutionary strategies

Policy Gradient

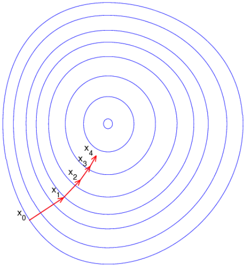
- Idea: ascent the gradient of the objective $J(\boldsymbol{\theta})$

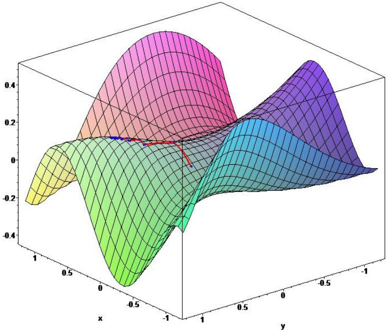
$$\Delta\boldsymbol{\theta} = \alpha \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$$

- Where $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ is the **policy gradient**

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

- and α is a step-size parameter
- Stochastic policies help ensure $J(\boldsymbol{\theta})$ is smooth (typically/mostly)





Gradients on parameterized policies

- How to compute this gradient $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$?
- Assume policy $\pi_{\boldsymbol{\theta}}$ is differentiable almost everywhere (e.g., neural net)
- For average reward

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}} \mathbb{E}_{\pi_{\boldsymbol{\theta}}}[R] \, .$$

- How does $\mathbb{E}[R]$ depend on $\boldsymbol{\theta}$?

Contextual Bandits Policy Gradient

- Consider a one-step case (a contextual bandit) such that $\boldsymbol{J(\theta) = \mathbb{E}_{\pi_\theta}[R(S, A)]}$.
  (Expectation is over d (states) and π (actions))
  (For now, d does **not** depend on π)
- We cannot sample R_{t+1} and then take a gradient:
  R_{t+1} is just a number and does not depend on $\boldsymbol{\theta}$!
- Instead, we use the identity:

$$\nabla_{\boldsymbol{\theta}} \mathbb{E}_{\pi_{\boldsymbol{\theta}}}[R(S, A)] = \mathbb{E}_{\pi_{\boldsymbol{\theta}}}[R(S, A) \nabla_{\boldsymbol{\theta}} \log \pi(A|S)] .$$

  (Proof on next slide)
- The right-hand side gives an expected gradient that can be sampled
- Also known as REINFORCE (Williams, 1992)

The score function trick

Let $r_{sa} = \mathbb{E}\left[R(S, A) \mid S = s, A = s\right]$

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}} \mathbb{E}_{\pi_{\boldsymbol{\theta}}}[R(S, A)] &= \nabla_{\boldsymbol{\theta}} \sum_s d(s) \sum_a \pi_{\boldsymbol{\theta}}(a|s)\, r_{sa} \\
&= \sum_s d(s) \sum_a r_{sa}\, \nabla_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}}(a|s) \\
&= \sum_s d(s) \sum_a r_{sa}\, \pi_{\boldsymbol{\theta}}(a|s) \frac{\nabla_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}}(a|s)}{\pi_{\boldsymbol{\theta}}(a|s)} \\
&= \sum_s d(s) \sum_a \pi_{\boldsymbol{\theta}}(a|s)\, r_{sa}\, \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a|s) \\
&= \mathbb{E}_{d,\pi_{\boldsymbol{\theta}}}[R(S, A)\, \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(A|S)]
\end{aligned}$$

Contextual Bandit Policy Gradient

$$\nabla_{\boldsymbol{\theta}} \mathbb{E}[R(S, A)] = \mathbb{E}[\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(A|S) R(S, A)] \qquad \text{(see previous slide)}$$

- This is something we **can** sample
- Our stochastic policy-gradient update is then

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha R_{t+1} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}_t}(A_t|S_t) \,.$$

- In expectation, this is the following the actual gradient
- So this is a pure (unbiased) stochastic gradient algorithm
- Intuition: increase probability for actions with high rewards

Policy gradients: reduce variance

- Note that, in general

$$\begin{aligned}
\mathbb{E}\left[b\nabla_{\boldsymbol{\theta}} \log \pi(A_t|S_t)\right] &= \mathbb{E}\left[\sum_a \pi(a|S_t) b \nabla_{\boldsymbol{\theta}} \log \pi(a|S_t)\right] \\
&= \mathbb{E}\left[b\nabla_{\boldsymbol{\theta}} \sum_a \pi(a|S_t)\right] \\
&= \mathbb{E}\left[b\nabla_{\boldsymbol{\theta}} 1\right] \qquad \mathbf{= 0}
\end{aligned}$$

- This is true if b does not depend on the action (but it can depend on the state)
- Implies we can subtract a **baseline** to reduce variance

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha(R_{t+1} - b(S_t))\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}_t}(A_t|S_t)\,.$$

- We will also use this fact in proofs below

Example: Softmax Policy

- Consider a softmax policy on action preferences $h(s, a)$ as an example
- Probability of action is proportional to exponentiated weight

$$\pi_{\boldsymbol{\theta}}(a|s) = \frac{e^{h(s,a)}}{\sum_b e^{h(s,b)}}$$

- The gradient of the log probability is

$$\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(A_t|S_t) = \underbrace{\nabla_{\boldsymbol{\theta}} h(S_t, A_t)}_{\text{gradient of preference}} - \underbrace{\sum_a \pi_{\boldsymbol{\theta}}(a|S_t) \nabla_{\boldsymbol{\theta}} h(S_t, a)}_{\text{expected gradient of preference}}$$

Policy Gradient Theorem

Policy Gradient Theorem

- The policy gradient approach also applies to (multi-step) MDPs
- Replaces reward R with long-term return G_t or value $q_\pi(s, a)$
- There are actually two policy gradient theorems (Sutton et al., 2000):

average return per episode & **average reward per step**

Policy gradient theorem (episodic)

Theorem

For any differentiable policy $\pi_{\boldsymbol{\theta}}(s, a)$, let d_0 be the starting distribution over states in which we begin an episode. Then, the policy gradient of $J(\boldsymbol{\theta}) = \mathbb{E}\left[G_0 \mid S_0 \sim d_0\right]$ is

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbb{E}_{\pi_{\boldsymbol{\theta}}}\left[\sum_{t=0}^{T} \gamma^t q_{\pi_{\boldsymbol{\theta}}}(S_t, A_t) \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(A_t | S_t) \mid S_0 \sim d_0\right]$$

where

$$\begin{aligned} q_\pi(s, a) &= \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a] \\ &= \mathbb{E}_\pi[R_{t+1} + \gamma q_\pi(S_{t+1}, A_{t+1}) \mid S_t = s, A_t = a] \end{aligned}$$

Policy gradients on trajectories

- Policy gradients do **not** need to know the MDP dynamics
- Kind of surprising; shouldn't we know how the policy influences the states?

Episodic policy gradients: proof

- Consider trajectory $\tau = S_0, A_0, R_1, S_1, A_1, R_1, S_2, \ldots$ with return $G(\tau)$

$$\nabla_{\boldsymbol{\theta}} J_{\boldsymbol{\theta}}(\pi) = \nabla_{\boldsymbol{\theta}} \mathbb{E}\left[G(\tau)\right] = \mathbb{E}\left[G(\tau) \nabla_{\boldsymbol{\theta}} \log p(\tau)\right] \qquad \text{(score function trick)}$$

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}} \log p(\tau) &= \nabla_{\boldsymbol{\theta}} \log \left[p(S_0) \pi(A_0|S_0) p(S_1|S_0, A_0) \pi(A_1|S_1) \cdots \right] \\
&= \nabla_{\boldsymbol{\theta}} \left[\log p(S_0) + \log \pi(A_0|S_0) + \log p(S_1|S_0, A_0) + \log \pi(A_1|S_1) + \cdots \right] \\
&= \nabla_{\boldsymbol{\theta}} \left[\log \pi(A_0|S_0) + \log \pi(A_1|S_1) + \cdots \right]
\end{aligned}$$

So:

$$\nabla_{\boldsymbol{\theta}} J_{\boldsymbol{\theta}}(\pi) = \mathbb{E}_{\pi}[G(\tau) \nabla_{\boldsymbol{\theta}} \sum_{t=0}^{T} \log \pi(A_t|S_t)]$$

Episodic policy gradients: proof (continued)

$$
\begin{aligned}
\nabla_{\boldsymbol{\theta}} J_{\boldsymbol{\theta}}(\pi) &= \mathbb{E}_\pi[G(\tau) \sum_{t=0}^{T} \nabla_{\boldsymbol{\theta}} \log \pi(A_t|S_t)] \\
&= \mathbb{E}_\pi[\sum_{t=0}^{T} G(\tau) \nabla_{\boldsymbol{\theta}} \log \pi(A_t|S_t)] \\
&= \mathbb{E}_\pi[\sum_{t=0}^{T} \left(\sum_{k=0}^{T} \gamma^k R_{k+1} \right) \nabla_{\boldsymbol{\theta}} \log \pi(A_t|S_t)] \\
&= \mathbb{E}_\pi[\sum_{t=0}^{T} \left(\sum_{k=\boldsymbol{t}}^{T} \gamma^k R_{k+1} \right) \nabla_{\boldsymbol{\theta}} \log \pi(A_t|S_t)] \\
&= \mathbb{E}_\pi[\sum_{t=0}^{T} \left(\gamma^t \sum_{k=t}^{T} \gamma^{k-t} R_{k+1} \right) \nabla_{\boldsymbol{\theta}} \log \pi(A_t|S_t)] \\
&= \mathbb{E}_\pi[\sum_{t=0}^{T} \left(\gamma^t G_t \right) \nabla_{\boldsymbol{\theta}} \log \pi(A_t|S_t)] \qquad = \mathbb{E}_\pi[\sum_{t=0}^{T} \gamma^t q_\pi(S_t, A_t) \nabla_{\boldsymbol{\theta}} \log \pi(A_t|S_t)]
\end{aligned}
$$

Episodic policy gradients algorithm

$$\nabla_{\boldsymbol{\theta}} J_{\boldsymbol{\theta}}(\pi) = \mathbb{E}_\pi[\sum_{t=0}^{T} \gamma^t q_\pi(S_t, A_t) \nabla_{\boldsymbol{\theta}} \log \pi(A_t|S_t)]$$

- We can sample this, given a whole episode
- Typically, people pull out the sum, and split up this into separate gradients, e.g.,

  $$\Delta\boldsymbol{\theta}_t = \gamma^t G_t \nabla_{\boldsymbol{\theta}} \log \pi(A_t|S_t)$$

  such that $\mathbb{E}_\pi[\sum_t \Delta\boldsymbol{\theta}_t] = \nabla_{\boldsymbol{\theta}} J_{\boldsymbol{\theta}}(\pi)$
- Typically, people ignore the γ^t term, use $\Delta\boldsymbol{\theta}_t = G_t \nabla_{\boldsymbol{\theta}} \log \pi(A_t|S_t)$
- This is actually okay-ish — we just partially pretend on each step that we could have started an episode in that state instead
  (alternatively, view it as a slightly biased gradient)

Policy gradient theorem (average reward)

Theorem

For any differentiable policy $\pi_{\boldsymbol{\theta}}(s,a)$, the policy gradient of $J(\boldsymbol{\theta}) = \mathbb{E}[R \mid \pi]$ is

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbb{E}_\pi[q_{\pi_\theta}(S_t, A_t)\nabla_\theta \log \pi_\theta(A_t|S_t)]$$

where

$$q_\pi(s,a) = \mathbb{E}_\pi[R_{t+1} - \rho + q_\pi(S_{t+1}, A_{t+1}) \mid S_t = s, A_t = a]$$
$$\rho = \mathbb{E}_\pi[R_{t+1}] \qquad \text{(Note: global average, not conditioned on state or action)}$$

(Expectation is over both states and actions)

Policy gradient theorem (average reward)

Alternatively (but equivalently):

Theorem

For any differentiable policy $\pi_{\boldsymbol{\theta}}(s, a)$, the policy gradient of $J(\boldsymbol{\theta}) = \mathbb{E}\left[R \mid \pi\right]$ is

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbb{E}_{\pi}\left[R_{t+1} \sum_{n=0}^{\infty} \nabla_{\theta} \log \pi_{\theta}(A_{t-n} | S_{t-n})\right]$$

(Expectation is over both states and actions)

Actor Critics

Policy gradients: reduce variance

- Recall $\mathbb{E}_\pi[b(S_t)\nabla \log \pi(A_t|S_t)] = 0$, for any $b(S_t)$ that does not depend on A_t
- A common baseline is $v_\pi(S_t)$

$$\nabla_{\boldsymbol{\theta}} J_{\boldsymbol{\theta}}(\pi) = \mathbb{E}\left[\sum_{t=0} \gamma^t (q_\pi(S_t, A_t) - \boldsymbol{v_\pi(S_t)})\nabla_{\boldsymbol{\theta}} \log \pi(A_t|S_t)\right]$$

- Typically, we estimate $v_{\mathbf{w}}(s) \approx v_\pi(s)$ explicitly, and sample

$$q_\pi(S_t, A_t) \approx G_t$$

- We can minimise variance further by **bootstrapping**, e.g., $G_t = R_{t+1} + \gamma v_{\mathbf{w}}(S_{t+1})$
- More on these techniques in the next lecture

Critics

- A critic is a value function, learnt via **policy evaluation**:
  What is the value $v_{\pi_{\boldsymbol{\theta}}}$ of policy $\pi_{\boldsymbol{\theta}}$ for current parameters $\boldsymbol{\theta}$?
- This problem was explored in previous lectures, e.g.
  - Monte-Carlo policy evaluation
  - Temporal-Difference learning
  - n-step TD

Actor-Critic

Critic Update parameters $\boldsymbol{w}$ of $v_{\boldsymbol{w}}$ by TD (e.g., one-step) or MC

Actor Update $\boldsymbol{\theta}$ by policy gradient

function ONE-STEP ACTOR CRITIC

Initialise s, $\boldsymbol{\theta}$

for t = 0, 1, 2, ... **do**

Sample $A_t \sim \pi_{\boldsymbol{\theta}}(S_t)$

Sample R_{t+1} and S_{t+1}

$\delta_t = R_{t+1} + \gamma v_{\boldsymbol{w}}(S_{t+1}) - v_{\boldsymbol{w}}(S_t)$ [one-step TD-error, or **advantage**]

$\boldsymbol{w} \leftarrow \boldsymbol{w} + \beta \, \delta_t \, \nabla_{\boldsymbol{w}} v_{\boldsymbol{w}}(S_t)$ [TD(0)]

$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \, \delta_t \, \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(A_t \mid S_t)$ [Policy gradient update (ignoring γ^t term)]

Policy gradient variations

- Many extensions and variants exist
- Take care: bad policies lead to bad data
- This is different from supervised learning
  (where learning and data are independent)

Increasing robustness with trust regions

- One way to increase stability is to **regularise**
- A popular method is to **limit the difference between subsequent policies**
- For instance, use the Kullbeck-Leibler divergence:

$$\mathrm{KL}(\pi_{\text{old}} \| \pi_{\boldsymbol{\theta}}) = \mathbb{E}\left[\int \pi_{\text{old}}(a \mid S) \log \frac{\pi_{\boldsymbol{\theta}}(a \mid S)}{\pi_{\text{old}}(a \mid S)} \, \mathrm{d}a \right].$$

(Expectation is over states)

- A divergence is like a distance between distributions
- Then maximise $J(\boldsymbol{\theta}) - \eta \mathrm{KL}(\pi_{\text{old}} \| \pi_{\boldsymbol{\theta}})$, for some hyperparameter η

c.f. **TRPO** (Schulman et al. 2015), **PPO** (Abbeel & Schulman 2016), **MPO** (Abdolmaleki et al. 2018)

Continuous action spaces

Continuous actions

- Pure value-based RL can be non-trivial to extend to **continuous action spaces**
  - How to approximate $q(s, a)$?
  - How to compute $\max_a q(s, a)$?
- When directly updating the policy parameters, continuous actions are easier
- Most algorithms discussed today can be used for discrete and continuous actions
- Note: exploration in high-dimensional continuous spaces can be challenging

Example: Gaussian policy

- As example, consider a **Gaussian policy**
- E.g., mean is some function of state $\mu_{\boldsymbol{\theta}}(s)$
- For simplicity, lets consider fixed variance of σ^2 (can be parametrized as well)
- Policy is Gaussian, $A_t \sim \mathcal{N}(\mu_{\boldsymbol{\theta}}(S_t), \sigma^2)$
  (here $\mu_{\boldsymbol{\theta}}$ is the mean — not to be confused with the behaviour policy!)
- The gradient of the log of the policy is then

$$\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(s, a) = \frac{A_t - \mu_{\boldsymbol{\theta}}(S_t)}{\sigma^2} \nabla \mu_{\boldsymbol{\theta}}(s)$$

- This can be used, for instance, in REINFORCE / actor critic

Example: Policy gradient with Gaussian policy

- Gaussian policy gradient update:

$$\begin{aligned}\boldsymbol{\theta}_{t+1} &= \boldsymbol{\theta}_t + \beta(G_t - v(S_t))\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(A_t|S_t) \\ &= \boldsymbol{\theta}_t + \beta(G_t - v(S_t))\frac{A_t - \mu_{\boldsymbol{\theta}}(S_t)}{\sigma^2}\nabla \mu_{\boldsymbol{\theta}}(S_t)\end{aligned}$$

- Intuition: if return was high, move $\mu_{\boldsymbol{\theta}}(S_t)$ toward A_t

Gradient ascent on value

- Policy gradients work well, but do not strongly exploit the critic
- If values generalise well, perhaps we can rely on them more?
  1. Estimate $q_{\mathbf{w}} \approx q_\pi$, e.g., with Sarsa
  2. Define **deterministic actor**: $A_t = \pi_{\boldsymbol{\theta}}(S_t)$
  3. Improve actor (**policy improvement**) by **gradient ascent on the value**:

$$\Delta\boldsymbol{\theta} \propto \frac{\partial Q_\pi(s,a)}{\partial\boldsymbol{\theta}} = \frac{\partial Q_\pi(s,\pi_{\boldsymbol{\theta}}(S_t))}{\partial\pi_{\boldsymbol{\theta}}(S_t)}\frac{\partial\pi_{\boldsymbol{\theta}}(S_t)}{\partial\boldsymbol{\theta}}$$

- Known under various names:
  "Action-dependent heuristic dynamic programming" (ADHDP; Werbos 1990, Prokhorov & Wunsch 1997)
  "Gradient ascent on the value" (van Hasselt & Wiering 2007)
  These days, mostly know as: "**Deterministic policy gradient**" (DPG; Silver et al. 2014)
- It's a form of **policy iteration**

Continuous actor-critic learning automaton (Cacla)

We can also define the error in action space, rather than parameter space

1. $a_t = \text{Actor}_{\boldsymbol{\theta}}(S_t)$ (get current (continuous) action proposal)
2. $A_t \sim \pi(\cdot|S_t, a_t)$ (e.g., $A_t \sim \mathcal{N}(a_t, \Sigma)$) (explore)
3. $\delta_t = R_{t+1} + \gamma v_{\mathbf{w}}(S_{t+1}) - v_{\mathbf{w}}(S_t)$ (compute TD error)
4. Update $v_{\mathbf{w}}(S_t)$ (e.g., using TD) (policy evaluation)
5. If $\delta_t > 0$, update $\text{Actor}_{\boldsymbol{\theta}}(S_t)$ towards A_t (policy improvement)

$$\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t + \beta(A_t - a_t)\nabla_{\boldsymbol{\theta}_t}\text{Actor}_{\boldsymbol{\theta}_t}(S_t)$$

6. If $\delta_t \leq 0$, do not update $\text{Actor}_{\boldsymbol{\theta}}$

   Note: update magnitude does not depend on the value magnitude

   Note: don't update 'away' from 'bad' actions

Video

(Peng, Berseth, van de Panne 2016)

End of Lecture