Lecture Overview

1. Policy Gradient Methods
2. REINFORCE
3. Actor-Critic Methods
4. Proximal Policy Optimization
Acknowledgement

Slide contents are partially based on *Reinforcement Learning: An Introduction* by Sutton and Barto and the Reinforcement Learning lecture by David Silver.
Up to this point, we represented a model or a value function by some parameterized function approximator and extracted the policy implicitly.

Today, we are going to talk about **Policy Gradient Methods**: methods which consider a parameterized policy

\[ \pi(a|s, \theta) = \Pr\{A_t = a|S_t = s, \theta_t = \theta\}, \]

with parameters \( \theta \)

Policy Gradient Methods are able to represent stochastic policies and scale naturally to very large or continuous action spaces.
We update these parameters based on the gradient of some performance measure $J(\theta)$ that we want to maximize, i.e. via gradient ascent:

$$\theta_{t+1} = \theta_t + \alpha \hat{\nabla} J(\theta_t),$$

where $\hat{\nabla} J(\theta_t) \in \mathbb{R}^d$ is a stochastic estimate whose expectation approximates the gradient of the performance measure w.r.t. $\theta_t$. 
Policy Gradient Theorem

Policy Objective Functions:
▶ For episodic problems we define performance as: \( J(\theta) = \eta(\pi_\theta) = \mathbb{E}_{s_0 \sim \rho_0} [v_{\pi_\theta}(s_0)] \)
▶ For continuing problems: \( J(\theta) = \sum_s \mu(s) v_{\pi_\theta}(s) \)

Policy Gradient Theorem

For any differentiable policy \( \pi(a|s, \theta) \) and any of the above policy objective functions, the policy gradient is:

\[
\nabla_{\theta} J(\theta) = \mathbb{E}_{\pi}[\nabla_{\theta} \log \pi(a|s, \theta) q_{\pi}(s, a)]
\]

Reminder: \( v_{\pi_\theta} = \sum_a \pi(a|s) q_{\pi}(s, a) \)
Score Function

- Likelihood ratios exploit the following identity:

\[
\nabla_{\theta} \pi(a|s, \theta) = \pi(a|s, \theta) \frac{\nabla_{\theta} \pi(a|s, \theta)}{\pi(a|s, \theta)}
\]

\[
= \pi(a|s, \theta) \nabla_{\theta} \log \pi(a|s, \theta)
\]

Easy to take the expectation because we can sample from \(\pi\)!

- \(\nabla_{\theta} \log \pi(a|s, \theta)\) is called the **score function**
Consider a Gaussian policy, where the mean is a linear combination of state features:
\[ \pi(a|s, \theta) \sim \mathcal{N}(s^\top \theta, \sigma^2), \] 
i.e.
\[ \pi(a|s, \theta) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2} \frac{(s^\top \theta - a)^2}{\sigma^2}\right) \]

Derivation of the score function

The log yields
\[ \log \pi(a|s, \theta) = -\frac{1}{2} \log(2\pi\sigma^2) - \frac{1}{2\sigma^2} (s^\top \theta - a)^2 \]

and the gradient
\[ \nabla_{\theta} \log \pi(a|s, \theta) = -\frac{1}{2\sigma^2} (s^\top \theta - a) 2s = \frac{(a - s^\top \theta)s}{\sigma^2}. \]
REINFORCE

- REINFORCE: Monte Carlo Policy Gradient
- Builds upon Monte Carlo returns as an unbiased sample of $q_\pi$
- However, therefore REINFORCE can suffer from high variance
REINFORCE: Monte-Carlo Policy-Gradient Control (episodic) for $\pi_*$

Input: a differentiable policy parameterization $\pi(a|s, \theta)$
Algorithm parameter: step size $\alpha > 0$
Initialize policy parameter $\theta \in \mathbb{R}^{d'}$ (e.g., to 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 $\pi(\cdot|\cdot, \theta)$
  Loop for each step of the episode $t = 0, 1, \ldots, T - 1$:
    $G \leftarrow \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$
    $\theta \leftarrow \theta + \alpha \gamma^t G \nabla \ln \pi(A_t|S_t, \theta)$

$(G_t)$
Vanilla REINFORCE provides *unbiased* estimates of the gradient $\nabla J(\theta)$, but it can suffer from high variance

- **Goal:** reduce variance while remaining unbiased
- **Observation:** we can generalize the policy gradient theorem by including an arbitrary *action-independent baseline* $b(s)$, i.e.

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

$$
= \sum_s \mu(s) \left[ \sum_a q_{\pi}(s,a) \nabla \pi(a|s) - b(s) \nabla \sum_a \pi(a|s) \right]
$$

$$
= \sum_s \mu(s) \sum_a q_{\pi}(s,a) \nabla \pi(a|s)
$$

- Baselines can reduce the variance of gradient estimates significantly!
Variance Reduction with Baselines

- A constant value can be used as a baseline
- The state-value function can be used as a baseline

Question
Is the Q-function a valid baseline?

Question
Assume an approximation of the state-value function as a baseline. Is REINFORCE then biased?
Indeed, an estimate of the state value function, $\hat{v}(S_t, w)$, is a very reasonable choice for $b(s)$:

**REINFORCE with Baseline (episodic), for estimating $\pi_\theta \approx \pi_*$**

- Input: a differentiable policy parameterization $\pi(a|s, \theta)$
- Input: a differentiable state-value function parameterization $\hat{v}(s,w)$
- Algorithm parameters: step sizes $\alpha^\theta > 0$, $\alpha^w > 0$
- Initialize policy parameter $\theta \in \mathbb{R}^{d'}$ and state-value weights $w \in \mathbb{R}^d$ (e.g., to 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 $\pi(\cdot|\cdot, \theta)$
- Loop for each step of the episode $t = 0, 1, \ldots, T - 1$:
  
  $G \leftarrow \sum_{k = t+1}^{T} \gamma^{k-t-1} R_k$
  
  $\delta \leftarrow G - \hat{v}(S_t, w)$
  
  $w \leftarrow w + \alpha^w \delta \nabla \hat{v}(S_t, w)$
  
  $\theta \leftarrow \theta + \alpha^\theta \gamma^t \delta \nabla \ln \pi(A_t|S_t, \theta)$
Methods that learn approximations to both policy and value functions are called actor-critic methods:

- **actor**: learned policy
- **critic**: learned value function (usually a state-value function)

Question: Is REINFORCE-with-baseline considered as an actor-critic method?
Actor-Critic Methods

- REINFORCE-with-baseline is unbiased, but tends to learn slowly and has high variance
- To gain from advantages of TD methods we use actor-critic methods with a bootstrapping critic

One-step actor-critic methods

Replace the full return of REINFORCE with one-step return as follows:

\[
\theta_{t+1} = \theta_t + \alpha (G_{t:t+1} - \hat{v}(S_t, w)) \frac{\nabla \pi(A_t|S_t, \theta_t)}{\pi(A_t|S_t, \theta_t)} \\
= \theta_t + \alpha (R_{t+1} + \gamma \hat{v}(S_{t+1}, w) - \hat{v}(S_t, w)) \frac{\nabla \pi(A_t|S_t, \theta_t)}{\pi(A_t|S_t, \theta_t)} \\
= \theta_t + \alpha \delta_t \frac{\nabla \pi(A_t|S_t, \theta_t)}{\pi(A_t|S_t, \theta_t)}
\]
Actor-Critic Methods

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

Input: a differentiable policy parameterization $\pi(a|s, \theta)$
Input: a differentiable state-value function parameterization $\hat{v}(s,w)$
Parameters: step sizes $\alpha^\theta > 0$, $\alpha^w > 0$
Initialize policy parameter $\theta \in \mathbb{R}^d$ and state-value weights $w \in \mathbb{R}^d$ (e.g., to 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):
    $A \sim \pi(\cdot|S, \theta)$
    Take action $A$, observe $S', R$
    $\delta \leftarrow R + \gamma \hat{v}(S',w) - \hat{v}(S,w)$ (if $S'$ is terminal, then $\hat{v}(S',w) \equiv 0$)
    $w \leftarrow w + \alpha^w \delta \nabla \hat{v}(S,w)$
    $\theta \leftarrow \theta + \alpha^\theta I \delta \nabla \ln \pi(A|S, \theta)$
    $I \leftarrow \gamma I$
    $S \leftarrow S'$
Proximal Policy Optimization

- We collect data with $\pi_{\theta_{old}}$
- And we want to optimize some objective to get a new policy $\pi_{\theta}$
- We can write $\eta(\pi_{\theta})$ in terms of $\pi_{\theta_{old}}$:

\[
\eta(\pi_{\theta}) = \eta(\pi_{\theta_{old}}) + \mathbb{E}_{\pi_{\theta}} \left[ \sum_{t=0}^{\infty} \gamma^t A_{\pi_{\theta_{old}}}(s_t, a_t) \right]
\]

where the **advantage function** is defined as

\[
A_{\pi_{\theta_{old}}}(s, a) = \mathbb{E}_{\pi_{\theta}, s_{t+1} \sim p} [q_{\pi_{\theta_{old}}}(s, a) - v_{\pi_{\theta_{old}}}(s)]
\]

\[
= \mathbb{E}_{\pi_{\theta}, s_{t+1} \sim p} [r(s, a) + \gamma v_{\pi_{\theta_{old}}}(s') - v_{\pi_{\theta_{old}}}(s)]
\]

- Advantage: how much better or worse is every action than average?
Proof:

\[
\mathbb{E}_{\pi \theta} \left[ \sum_{t=0}^{\infty} \gamma^t A_{\pi_{\theta_{old}}} (s_t, a_t) \right] \\
= \mathbb{E}_{\pi \theta, s_{t+1} \sim p} \left[ \sum_{t=0}^{\infty} \gamma^t (r(s_t, a_t) + \gamma v_{\pi_{\theta_{old}}} (s_{t+1}) - v_{\pi_{\theta_{old}}} (s_t)) \right] \\
= \mathbb{E}_{\pi \theta, s_{t+1} \sim p} \left[ -v_{\pi_{\theta_{old}}} (s_0) + \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \right] \\
= \mathbb{E}_{s_0 \sim p_0} \left[ -v_{\pi_{\theta_{old}}} (s_0) \right] + \mathbb{E}_{\pi \theta, s_{t+1} \sim p} \left[ \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \right] \\
= -\eta(\pi_{\theta_{old}}) + \eta(\pi_{\theta})
\]
In PPO, we *ignore* the change in state distribution and optimize a **surrogate objective**:

\[
J_{\text{old}}(\theta) = \mathbb{E}_{s \sim \pi_{\theta_{\text{old}}}, a \sim \pi_{\theta}}[A_{\pi_{\theta_{\text{old}}}}(s, a)]
\]

\[
= \mathbb{E}_{(s, a) \sim \pi_{\theta_{\text{old}}}} \left[ \frac{\pi_{\theta}}{\pi_{\theta_{\text{old}}}} A_{\pi_{\theta_{\text{old}}}}(s, a) \right]
\]

**Improvement Theory:** \( \eta(\pi_{\theta}) \geq J_{\text{old}}(\theta) - c \cdot \max_s \text{KL}[\pi_{\theta_{\text{old}}} \| \pi_{\theta}] \)

**If we keep the KL-divergence between our old and new policies small, optimizing the surrogate is close to optimizing \( \eta(\pi_{\theta}) \)!
Proximal Policy Optimization

- **Clipped Surrogate Objective:**
  \[
  \mathbb{E}_{(s,a) \sim \pi_{\theta_{old}}} \left[ \min \left( \frac{\pi_{\theta}}{\pi_{\theta_{old}}} A_{\pi_{\theta_{old}}} (s, a), \text{clip}(\frac{\pi_{\theta}}{\pi_{\theta_{old}}}, 1 - \epsilon, 1 + \epsilon) A_{\pi_{\theta_{old}}} (s, a) \right) \right]
  \]

- **Adaptive Penalty Surrogate Objective:**
  \[
  \mathbb{E}_{(s,a) \sim \pi_{\theta_{old}}} \left[ \frac{\pi_{\theta}}{\pi_{\theta_{old}}} A_{\pi_{\theta_{old}}} (s, a) - \beta \text{KL}[\pi_{\theta_{old}} \| \pi_{\theta}] \right]
  \]

**for iteration** \( i = 1, 2, \ldots \) **do**

- Run policy for \( T \) timesteps of \( N \) trajectories
- Estimate advantage function at all timesteps
- Do SGD on one of the above objectives for some number of epochs
  - In case of the Adaptive Penalty Surrogate: Increase \( \beta \) if KL-divergence too high,
  - otherwise decrease \( \beta \)

**end**

**Algorithm 1:** PPO