New Idea: instead deriving policy from estimated value function, learn a parameterized policy directly

\[ \pi(a|s, \theta) \doteq \text{Pr}\{A_t = a \mid S_t = s, \theta_t = \theta\} \quad \theta \in \mathbb{R}^n \]

Sometimes we’ll still learn a value function, with parameter vector \( w \): \( \hat{v}(s, w) \)

We can consider some performance measure \( \eta(\theta) \) and perform gradient ascent w.r.t. the policy parameters
Performance Measures

- for episodic case: value of the start state under the parameterized policy (assuming each episode starts in the same state):

  \[ \eta(\theta) = v_{\pi_\theta}(s_0) \]

- For continuing case: average reward:

  \[ \eta(\theta) = r(\theta) \]

- Gradient ascent:

  \[ \theta_{t+1} = \theta_t + \alpha \nabla \eta(\theta_t) \]

  The problem is to estimate the gradient
13.1 Policy Approximation and its Advantages

- policy can be parameterized in any way as long as the gradient $\nabla_\theta \pi(a|s, \theta)$ exists and is always finite.
- ensure exploration by requiring policy from never becoming deterministic:

$$\pi(a|s, \theta) \in (0, 1) \quad \forall s, a, \theta$$

- for discrete actions we can introduce numerical preferences:

$$h(s, a, \theta) \in \mathbb{R}$$

- we can select actions via a softmax distribution:

$$\pi(a|s, \theta) \propto \frac{\exp(h(s, a, \theta))}{\sum_b \exp(h(s, b, \theta))}$$
Advantage of softmax action selection: approximate policy can approach determinism (if learning drives preferences to become very big), whereas this is not possible for \( \epsilon \)-greedy methods.

Action preferences can be parameterized in many different ways, e.g.:

- Deep neural network
- Linear function of features:
  \[
  h(s, a, \theta) = \theta^\top \phi(s, a) \quad \phi(s, a) \in \mathbb{R}^n
  \]
- Etc.
Advantages

- policy can be a simpler function to approximate compared to action value function in some cases
- the best policy may be stochastic in some cases; action value methods have no way of finding stochastic optimal policies
- choice of policy parameterization is a way of injecting prior knowledge in the system
13.2 The Policy Gradient Theorem

- Policy parameters change how actions are chosen and therefore what rewards we get and which states we will see how often.

- But effect of policy on state distribution is a function of the environment and unknown.

- How can we still estimate the dependence of the performance gradient on the policy weights?

- Answer: Policy Gradient Theorem (proof: see black board)

\[
\nabla \eta(\theta) = \sum_s d_\pi(s) \sum_a q_\pi(s, a) \nabla_\theta \pi(a|s, \theta)
\]
How can we turn this into a stochastic gradient ascent algorithm?

We need samples whose expected value is equal to this gradient.

Step 1: sample over states

$$\nabla \eta(\theta) = \sum_s d_\pi(s) \sum_a q_\pi(s, a) \nabla_\theta \pi(a | s, \theta)$$

$$= \mathbb{E}_\pi \left[ \gamma^t \sum_a q_\pi(S_t, a) \nabla_\theta \pi(a | S_t, \theta) \right]$$
Step 2: sample over actions. From before:

\[ \nabla \eta(\theta) = \mathbb{E}_{\pi} \left[ \gamma^t \sum_a q_\pi(S_t, a) \nabla_\theta \pi(a|S_t, \theta) \right] \]

\[ \nabla \eta(\theta) = \mathbb{E}_{\pi} \left[ \gamma^t \sum_a \pi(a|S_t, \theta) q_\pi(S_t, a) \frac{\nabla_\theta \pi(a|S_t, \theta)}{\pi(a|S_t, \theta)} \right] \]

\[ = \mathbb{E}_{\pi} \left[ \gamma^t q_\pi(S_t, A_t) \frac{\nabla_\theta \pi(A_t|S_t, \theta)}{\pi(A_t|S_t, \theta)} \right] \quad \text{(replacing } a \text{ by the sample } A_t \sim \pi) \]

\[ = \mathbb{E}_{\pi} \left[ \gamma^t G_t \frac{\nabla_\theta \pi(A_t|S_t, \theta)}{\pi(A_t|S_t, \theta)} \right] \quad \text{(because } \mathbb{E}_{\pi}[G_t|S_t, A_t] = q_\pi(S_t, A_t)) \]

resulting stochastic gradient ascent rule:

\[ \theta_{t+1} = \theta_t + \alpha \gamma^t G_t \frac{\nabla_\theta \pi(A_t|S_t, \theta)}{\pi(A_t|S_t, \theta)} \]
REINFORCE Algorithm (episodic case)

### REINFORCE, A Monte-Carlo Policy-Gradient Method (episodic)

Input: a differentiable policy parameterization \( \pi(a|s, \theta), \forall a \in \mathcal{A}, s \in \mathcal{S}, \theta \in \mathbb{R}^n \)

Initialize policy weights \( \theta \)

Repeat forever:
- Generate an episode \( S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T \), following \( \pi(\cdot|\cdot, \theta) \)
- For each step of the episode \( t = 0, \ldots, T - 1 \):
  - \( G_t \leftarrow \) return from step \( t \)
  - \( \theta \leftarrow \theta + \alpha \gamma^t G_t \nabla_\theta \log \pi(A_t|S_t, \theta) \)

\[
\frac{\nabla_\theta \pi(A_t|S_t, \theta)}{\pi(A_t|S_t, \theta)} = \nabla_\theta \log \pi(A_t|S_t, \theta) \quad \nabla \log x = \frac{\nabla x}{x}
\]

“eligibility vector”

R. S. Sutton and A. G. Barto: Reinforcement Learning: An Introduction
13.4 REINFORCE with Baseline

- We can subtract an arbitrary baseline \( b(s) \) in the gradient calculation (can reduce variance of gradient estimate):

\[
\nabla \eta(\theta) = \sum_s d_\pi(s) \sum_a \left( q_\pi(s, a) - b(s) \right) \nabla_\theta \pi(a|s, \theta)
\]

- This baseline does not change the gradient because:

\[
\sum_a b(s) \nabla_\theta \pi(a|s, \theta) = b(s) \nabla_\theta \sum_a \pi(a|s, \theta) = b(s) \nabla_\theta 1 = 0 \quad \forall s \in S
\]

- This gives a new update rule:

\[
\theta_{t+1} = \theta_t + \alpha \left( G_t - b(S_t) \right) \frac{\nabla_\theta \pi(A_t|S_t, \theta)}{\pi(A_t|S_t, \theta)}
\]
What to choose as the baseline?

- One natural choice is an estimate of the value of the current state, estimated with state value weights $\mathbf{w}$: $\hat{v}(S_t, \mathbf{w})$

**REINFORCE with Baseline (episodic)**

| Input: a differentiable policy parameterization $\pi(a|s, \theta), \forall a \in A, s \in S, \theta \in \mathbb{R}^n$ |
|---|
| Input: a differentiable state-value parameterization $\hat{v}(s, \mathbf{w}), \forall s \in S, \mathbf{w} \in \mathbb{R}^m$ |
| Parameters: step sizes $\alpha > 0$, $\beta > 0$ |
| Initialize policy weights $\theta$ and state-value weights $\mathbf{w}$ |
| Repeat forever: |
| - Generate an episode $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$, following $\pi(\cdot|\cdot, \theta)$ |
| - For each step of the episode $t = 0, \ldots, T-1$: |
| - $G_t \leftarrow$ return from step $t$ |
| - $\delta \leftarrow G_t - \hat{v}(S_t, \mathbf{w})$ |
| - $\mathbf{w} \leftarrow \mathbf{w} + \beta \delta \nabla_{\mathbf{w}} \hat{v}(S_t, \mathbf{w})$ |
| - $\theta \leftarrow \theta + \alpha \gamma^t \delta \nabla_{\theta} \log \pi(A_t|S_t, \theta)$ |
13.5 Actor-Critic Methods

- **Idea:** introduce bootstrapping into policy gradient approach

- First consider one-step version. For this, replace the Monte Carlo Return $G_t$ with the one-step TD return estimate:

\[
\theta_{t+1} = \theta_t + \alpha \left( G_t^{(1)} - \hat{v}(S_t, w) \right) \frac{\nabla_{\theta} \pi (A_t | S_t, \theta)}{\pi (A_t | S_t, \theta)}
\]

\[
= \theta_t + \alpha \left( R_{t+1} + \gamma \hat{v}(S_{t+1}, w) - \hat{v}(S_t, w) \right) \frac{\nabla_{\theta} \pi (A_t | S_t, \theta)}{\pi (A_t | S_t, \theta)}
\]
# One-step Actor-Critic Algorithm

## One-step Actor-Critic (episodic)

Input: a differentiable policy parameterization $\pi(a|s, \theta), \forall a \in A, s \in S, \theta \in \mathbb{R}^n$

Input: a differentiable state-value parameterization $\hat{v}(s,w), \forall s \in S, w \in \mathbb{R}^m$

Parameters: step sizes $\alpha > 0$, $\beta > 0$

Initialize policy weights $\theta$ and state-value weights $w$

Repeat forever:

- Initialize $S$ (first state of episode)
- $I \leftarrow 1$

While $S$ is not terminal:

- $A \sim \pi(\cdot|S, \theta)$
- Take action $A$, observe $S', R$
- $\delta \leftarrow R + \gamma \hat{v}(S',w) - \hat{v}(S,w)$ (if $S'$ is terminal, then $\hat{v}(S',w) \doteq 0$)
- $w \leftarrow w + \beta \delta \nabla_w \hat{v}(S,w)$
- $\theta \leftarrow \theta + \alpha I \delta \nabla_\theta \log \pi(A|S, \theta)$
- $I \leftarrow \gamma I$
- $S \leftarrow S'$
### Actor-Critic with Eligibility Traces

**Actor-Critic with Eligibility Traces (episodic)**

Input: a differentiable policy parameterization $\pi(a|s, \theta), \forall a \in A, s \in S, \theta \in \mathbb{R}^n$

Input: a differentiable state-value parameterization $\hat{v}(s, w), \forall s \in S, w \in \mathbb{R}^m$

Parameters: step sizes $\alpha > 0, \beta > 0$

Initialize policy weights $\theta$ and state-value weights $w$

Repeat forever:

- Initialize $S$ (first state of episode)
- $e^\theta \leftarrow 0$ (n-component eligibility trace vector)
- $e^w \leftarrow 0$ (m-component eligibility trace vector)
- $I \leftarrow 1$

While $S$ is not terminal:

- $A \sim \pi(\cdot|S, \theta)$
- Take action $A$, observe $S', R$
- $\delta \leftarrow R + \gamma \hat{v}(S', w) - \hat{v}(S, w)$ (if $S'$ is terminal, then $\hat{v}(S', w) \doteq 0$)
- $e^w \leftarrow \lambda^w e^w + I \nabla_w \hat{v}(S, w)$
- $e^\theta \leftarrow \lambda^\theta e^\theta + I \nabla_\theta \log \pi(A|S, \theta)$
- $w \leftarrow w + \beta \delta e^w$
- $\theta \leftarrow \theta + \alpha \delta e^\theta$
- $I \leftarrow \gamma I$
- $S \leftarrow S'$

---

13.6 Policy Gradient for Continuing Problems (Average Reward Rate)

For continuing problems without episode boundaries, as discussed in Chapter 10, we need to define performance in terms of the average rate of reward per time step:

$$\gamma(\theta) = \lim_{n \to \infty} \frac{1}{n} \sum_{t=1}^{n} E[R_t | \theta_0 = \theta_1 = \cdots = \theta_t = \theta_t]$$ (13.12)

$$= \lim_{t \to \infty} E[R_t | \theta_0 = \theta_1 = \cdots = \theta_t = \theta_t],$$

where the limits are assumed to exist and to be the same for any $s_0 \in S$, $\gamma(\theta) = \sum_s d^\pi(s) \sum_a \pi(a|s, \theta) \sum_{s', r} p(s', r|s, a)$.

Note that this is the special distribution under which, if you select actions according to $\pi$, you remain in the same distribution:

$$d^\pi(s) = \lim_{t \to \infty} \Pr\{S_t = s | A_0: t \sim \pi\},$$

which is assumed to exist and to be independent of $S_0$ (an ergodicity assumption).
13.6 Policy Gradient for Continuing Problems (Average Reward Rate)

- For the continuing case, all results and algorithms are analogous.
- Objective function: average reward per time step

\[ \eta(\theta) \doteq r(\theta) \doteq \lim_{n \to \infty} \frac{1}{n} \sum_{t=1}^{n} \mathbb{E}[R_t \mid \theta_0 = \theta_1 = \cdots = \theta_{t-1} = \theta] \]

\[ = \lim_{t \to \infty} \mathbb{E}[R_t \mid \theta_0 = \theta_1 = \cdots = \theta_{t-1} = \theta], \]

\[ = \sum_s d_{\pi\theta}(s) \sum_a \pi\theta(a \mid s, \theta) \sum_{s,r'} p(s,r' \mid s,a) r, \]

- Values:

\[ v_{\pi}(s) \doteq \sum_{k=1}^{\infty} \mathbb{E}_{\pi}[R_{t+k} - r(\pi) \mid S_t = s] \]

\[ q_{\pi}(s,a) \doteq \sum_{k=1}^{\infty} \mathbb{E}_{\pi}[R_{t+k} - r(\pi) \mid S_t = s, A_t = a] \]
Actor-Critic algorithm with Eligibility Trace for the continuing case

We also define all values di
\[ \begin{align*}
    v(\pi)(s) &= \frac{1}{N} \sum_{k=1}^{N} E[ \sum_{t} R_t + k r(\pi) | S_t = s ] \quad (13.14)
\end{align*} \]
\[ \begin{align*}
    q(\pi)(s,a) &= \frac{1}{N} \sum_{k=1}^{N} E[ \sum_{t} R_t + k r(\pi) | S_t = s, A_t = a ] . \quad (13.15)
\end{align*} \]

With these alternate definitions, the policy gradient theorem as given for the episodic case (13.5) remains true for the continuing case. A proof is given in the box on the next page. The forward and backward view equations also remain the same.

Complete pseudocode for the backward view is given in the box below.

A very simple worked example should go here.

---

**Actor-Critic with Eligibility Traces (continuing)**

Input: a differentiable policy parameterization \( \pi(a|s, \theta), \forall a \in A, s \in S, \theta \in \mathbb{R}^n \)

Input: a differentiable state-value parameterization \( \hat{v}(s,w), \forall s \in S, w \in \mathbb{R}^m \)

Parameters: step sizes \( \alpha > 0, \beta > 0, \eta > 0 \)

\[ \begin{align*}
    \text{e}^\theta &\leftarrow 0 \text{ (n-component eligibility trace vector)} \\
    \text{e}^w &\leftarrow 0 \text{ (m-component eligibility trace vector)} \\
    \text{Initialize } R &\in \mathbb{R} \text{ (e.g., to 0)} \\
    \text{Initialize policy weights } \theta \text{ and state-value weights } w \text{ (e.g., to 0)} \\
    \text{Initialize } S &\in S \text{ (e.g., to } s_0) \\
    \text{Repeat forever:} \\
    & A \sim \pi(\cdot|S, \theta) \\
    & \text{Take action } A, \text{ observe } S', R \\
    & \delta \leftarrow R - R + \hat{v}(S',w) - \hat{v}(S,w) \quad \text{(if } S' \text{ is terminal, then } \hat{v}(S',w) \doteq 0) \\
    & R \leftarrow R + \eta \delta \\
    & \text{e}^w \leftarrow \lambda^w \text{e}^w + \nabla_w \hat{v}(S,w) \\
    & \text{e}^\theta \leftarrow \lambda^\theta \text{e}^\theta + \nabla_\theta \log \pi(A|S, \theta) \\
    & w \leftarrow w + \beta \delta \text{e}^w \\
    & \theta \leftarrow \theta + \alpha \delta \text{e}^\theta \\
    & S \leftarrow S'
\end{align*} \]
13.7 Policy Parameterization for Continuous Actions

- How can we deal with continuous actions?
- Idea: instead of learning probabilities for each action in a given state, learn the parameters of a distribution from which actions are sampled.
- If, e.g., actions are real numbers, sample actions from a Gaussian distribution, where the mean and variance depend on the parameters $\theta$:

$$
\pi(a|s, \theta) \doteq \frac{1}{\sigma(s, \theta) \sqrt{2\pi}} \exp\left(-\frac{(a - \mu(s, \theta))^2}{2\sigma(s, \theta)^2}\right)
$$

$$
\mu(s, \theta) \doteq \theta^\top \phi(s) \quad \text{and} \quad \sigma(s, \theta) \doteq \exp\left(\theta^{\sigma^\top} \phi(s)\right)
$$

(makes sure variance is positive)
Summary

- Policy gradient: rather than relying on estimating value functions, use a parameterized policy and improve it via stochastic gradient ascent
- Policy gradient theorem
- REINFORCE algorithms
- Actor-Critic algorithms
- Policy parameterization for continuous actions