Lecture 8
Policy Gradient Algorithms

B. Ravindran

Solution Methods

- Temporal Difference Methods
  - TD(λ)
  - Q-learning
  - SARSA
  - Actor-Critic
- Policy Search
  - Policy Gradient Methods
  - Evolutionary algorithms
- Model based methods
  - Stochastic Dynamic Programming
  - Bayesian approaches

Solution Methods

- Temporal Difference Methods
  - TD(λ)
  - Q-learning
  - SARSA
  - Actor-Critic
- Policy Search
  - Policy Gradient Methods
  - ~~Evolutionary algorithms~~
- Model based methods
  - Stochastic Dynamic Programming
  - ~~Bayesian approaches~~

Solution Methods

- Temporal Difference Methods
  - TD(λ)
  - Q-learning
  - SARSA
  - Actor-Critic
- Policy Search
  - **Policy Gradient Methods**
  - ~~Evolutionary algorithms~~
- Model based methods
  - Stochastic Dynamic Programming
  - ~~Bayesian approaches~~

Policy Search Methods

- Policy search: Instead of maintaining estimates of value functions, search in the space of policies
- **Why?**
  - Simpler description
  - Better convergence
  - Robust to partial observability
- Direct policy search
  - Genetic algorithms
- Policy Gradient Approaches

Policy Gradient Methods

- Policy depends on some parameters θ
  - Action preferences
  - Mean and variance
  - Weights of a neural network
- **Idea:** Modify policy parameters directly instead of estimating the action values
- Maximize:

$$J(\boldsymbol{\theta}) = \mathbb{E}(r_t)$$

- θ update:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \nabla J(\boldsymbol{\theta})$$

Simplified Setting
Immediate Reward or
Multi-arm bandits

Multi-Armed Bandit

- n-arm bandit problem is to learn to preferentially select a particular action (arm) from a set of n actions $(1, 2, 3, \ldots, n)$
- Each selection results in Rewards, with noise
  - Sometimes you get a reward, sometimes don't
- Pick arm that has the highest expected reward

Policy Gradient Methods

- Policy depends on some parameters θ
  - Action preferences
  - Mean and variance
  - Weights of a neural network
- **Idea:** Modify policy parameters directly instead of estimating the action values
- Maximize:

$$J(\boldsymbol{\theta}) = \mathbb{E}(r_t) = \sum_a Q^*(a)\pi_{\boldsymbol{\theta}}(a)$$

- θ update:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \nabla J(\boldsymbol{\theta})$$

Simplified Setting

Immediate Reward or
Multi-arm bandits

Stochastic Gradient Descent

- We compute the gradient of the performance ***J(θ)*** w.r.t the parameters ***θ***

$$
\begin{aligned}
J(\boldsymbol{\theta}) &= \mathbb{E}\left(r_t\right) \\
&= \sum_a Q^*(a)\pi_{\boldsymbol{\theta}}(a) \\
\nabla J(\boldsymbol{\theta}) &= \sum_a Q^*(a)\nabla\pi_{\boldsymbol{\theta}}(a) \\
&= \sum_a Q^*(a)\frac{\nabla\pi_{\boldsymbol{\theta}}(a)}{\pi_{\boldsymbol{\theta}}(a)}\pi_{\boldsymbol{\theta}}(a)
\end{aligned}
$$

- Estimate the gradient from ***N*** samples

$$
\hat{\nabla} J(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N} r_i \underbrace{\frac{\nabla\pi_{\boldsymbol{\theta}}\left(a_i\right)}{\pi_{\boldsymbol{\theta}}\left(a_i\right)}}_{\substack{\text{Likelihood}\\ \text{Ratio}}}
$$

REINFORCE (Williams ‘92)

- Incremental version

$$\Delta \boldsymbol{\theta}_t = \alpha r_t \frac{\nabla \pi_{\boldsymbol{\theta}}(a_t)}{\pi_{\boldsymbol{\theta}}(a_t)}$$

$$\Delta \boldsymbol{\theta}_t = \alpha r_t \frac{\partial \ln \pi_{\boldsymbol{\theta}}(a_t)}{\partial \boldsymbol{\theta}}$$

- REINFORCE with baseline

$$\Delta \boldsymbol{\theta}_t = \alpha (r_t - \boxed{b_t}) \boxed{\frac{\partial \ln \pi_{\boldsymbol{\theta}}(a_t)}{\partial \boldsymbol{\theta}}}$$

Reinforcement Baseline

Characteristic Eligibility

Policy Gradient Theorem

- In the episodic case, we define the performance by assuming that every episode starts from state s_0 (non-random), as follows:

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

where $v_{\pi_{\boldsymbol{\theta}}}(s_0)$ is the true value function given a parameterized policy $\pi_{\boldsymbol{\theta}}$

- **The Policy Gradient Theorem**: The gradient of the performance can be expressed in terms of the gradient of the policy, as follows

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

$\mu(s)$ is the fraction of time spent in state s (on policy distribution)

Policy Gradient Theorem

- The policy gradient theorem provides an analytic expression for the gradient of performance with respect to the policy parameter $\boldsymbol{\theta}$
- We begin the proof by expressing the gradient of the state-value function in terms of the action-value function

$$\begin{aligned}
\nabla v_\pi(s) &= \nabla\left[\sum_a \pi(a \mid s) q_\pi(s, a)\right], \quad \text{for all } s \in \mathcal{S} \\
&= \sum_a \left[\nabla \pi(a \mid s) q_\pi(s, a) + \pi(a \mid s) \nabla q_\pi(s, a)\right] \\
&= \sum_a \left[\nabla \pi(a \mid s) q_\pi(s, a) + \pi(a \mid s) \nabla \sum_{s', r} p\left(s', r \mid s, a\right)\left(r + v_\pi\left(s'\right)\right)\right]
\end{aligned}$$

Policy Gradient Theorem: Proof

$$\nabla v_\pi(s) = \sum_a \left[\nabla \pi(a \mid s) q_\pi(s,a) + \pi(a \mid s) \sum_{s'} p\left(s' \mid s, a\right) \nabla v_\pi\left(s'\right) \right]$$

$$= \sum_a \left[\nabla \pi(a \mid s) q_\pi(s,a) + \pi(a \mid s) \sum_{s'} p\left(s' \mid s, a\right) \right.$$

$$\left. \sum_{a'} \left[\nabla \pi\left(a' \mid s'\right) q_\pi\left(s', a'\right) + \pi\left(a' \mid s'\right) \sum_{s''} p\left(s'' \mid s', a'\right) \nabla v_\pi\left(s''\right) \right] \right]$$

$$= \sum_{x \in \mathcal{S}} \sum_{k=0}^{\infty} \Pr(s \to x, k, \pi) \sum_a \nabla \pi(a \mid x) q_\pi(x,a)$$

$\Pr(s \to x, k, \pi)$ is the probability of transitioning from state *s* to state *x* in *k* steps under policy π

Policy Gradient Theorem: Proof (contd.)

$$
\begin{aligned}
\nabla J(\boldsymbol{\theta}) &= \nabla v_\pi\left(s_0\right) \\
&= \sum_s\left(\sum_{k=0}^{\infty} \operatorname{Pr}\left(s_0 \rightarrow s, k, \pi\right)\right) \sum_a \nabla \pi(a \mid s) q_\pi(s, a) \\
&= \sum_s \eta(s) \sum_a \nabla \pi(a \mid s) q_\pi(s, a) \\
&= \sum_{s^{\prime}} \eta\left(s^{\prime}\right) \sum_s \frac{\eta(s)}{\sum_{s^{\prime}} \eta\left(s^{\prime}\right)} \sum_a \nabla \pi(a \mid s) q_\pi(s, a) \\
&= \sum_{s^{\prime}} \eta\left(s^{\prime}\right) \sum_s \mu(s) \sum_a \nabla \pi(a \mid s) q_\pi(s, a) \\
&\propto \sum_s \mu(s) \sum_a \nabla \pi(a \mid s) q_\pi(s, a)
\end{aligned}
$$

REINFORCE: MC Policy Gradient

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

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

Uses the complete return from time *t*, which includes all future rewards up until the end of the episode

REINFORCE: MC Policy Gradient Algorithm

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

- The algorithm computes an unbiased estimate of the gradient
- Can be very slow due to high variance in the estimates
- Variance is related to the “recurrence time" or the episode length
- For problems with large state spaces, the variance becomes unacceptably high.

REINFORCE: With Baseline

- The policy gradient theorem can be generalized to include a comparison of the action value to an arbitrary baseline *b(s)*

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

- The baseline should be a function (or a random variable) that does not depend on the action *a*, in which case, the subtracted quantity is 0

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

- Update rule of REINFORCE with baseline

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

- Doesn’t change the expected value, but has an effect on the variance

Actor-Critic Methods

- Actor-Critic methods learn both a policy and a state-value function simultaneously.
- The policy is referred to as the actor that suggests actions given a state.
- The estimated value function is referred to as the critic. It evaluates actions taken by the actor based on the given policy.

$$
\begin{aligned}
\boldsymbol{\theta}_{t+1} &\doteq \boldsymbol{\theta}_t + \alpha\Big(G_{t:t+1} - \hat{v}(S_t,\mathbf{w})\Big)\frac{\nabla\pi(A_t|S_t,\boldsymbol{\theta}_t)}{\pi(A_t|S_t,\boldsymbol{\theta}_t)} \\
&= \boldsymbol{\theta}_t + \alpha\Big(R_{t+1} + \gamma\hat{v}(S_{t+1},\mathbf{w}) - \hat{v}(S_t,\mathbf{w})\Big)\frac{\nabla\pi(A_t|S_t,\boldsymbol{\theta}_t)}{\pi(A_t|S_t,\boldsymbol{\theta}_t)} \\
&= \boldsymbol{\theta}_t + \alpha\,\delta_t\frac{\nabla\pi(A_t|S_t,\boldsymbol{\theta}_t)}{\pi(A_t|S_t,\boldsymbol{\theta}_t)}.
\end{aligned}
$$

One-step Actor-Critic Algorithm

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

Monte-Carlo Policy Gradient

- Samples are entire trajectories – $\{s_0, a_0, r_1, s_1, a_1, r_2, \cdots, s_T\}$
- Evaluation criterion is the return along the path, instead of the immediate rewards
- The gradient estimation equation hence becomes

$$\hat{\nabla}_\theta \eta(\theta) = \frac{1}{N} \sum_{i=1}^{N} G_i(s_0) \frac{\nabla_\theta p_{\theta,i}(s_0)}{p_{\theta,i}(s_0)}$$

where, $G_i(s_0)$ is the return starting from state s_0, and $p_{\theta,i}(s_0)$ is the probability of the i^{th} trajectory, starting from s_0, and using policy given by θ.

Monte-Carlo Policy Gradient (contd.)

- The "likelihood ratio" in this case evaluates to

$$\frac{\nabla_\theta p_{\theta,i}(s_0)}{p_{\theta,i}(s_0)} = \sum_{j=0}^{T-1} \frac{\nabla_\theta \pi_\theta(s_j, a_j)}{\pi_\theta(s_j, a_j)}$$

- The estimate depends on the starting state s_0. One way to address this issue is to assume a fixed initial state
- More common assumption is to use the average reward formulation

Policy Gradient Theorem

Let $\eta(\Theta) = E_\pi(R_t)$

$$\nabla \eta(\Theta) = E_\pi[\nabla_\Theta \log \pi(s,a;\Theta).Q^\pi(s,a)]$$

Can approximate $Q^\pi(s,a)$ using the return as we did in MC policy gradient.

Alternatively we can use the Q value itself, leant via a TD procedure (e.g. SARSA)!

$$\nabla \eta(\Theta) \approx E_\pi[\nabla_\Theta \log \pi(s,a;\Theta).Q^\pi(s,a;w)]$$

$$\Delta\Theta = \alpha \nabla_\Theta \log \pi(s,a;\Theta).Q^\pi(s,a;w)$$

But the value parameterization should be *compatible* to the action parameterization

Incremental Update

- We can incrementally compute the summation in Equation 1, over one trajectory as follows:

$$
\begin{aligned}
z_{t+1} &= z_t + \frac{\nabla \pi(s_t, a_t; \Theta)}{\pi(s_t, a_t; \Theta)} \\
R_{t+1} &= R_t + \frac{1}{t+1} [r_t - R_t]
\end{aligned}
$$

- z_T is known as an eligibility trace. Recall the characteristic eligibility term from REINFORCE:

$$
\frac{\partial \ln \pi(a_t; \Theta)}{\partial \Theta}.
$$

- z_T keeps track of this eligibility over time, hence is called a trace.

Verify this slide before typing

Simple MC Policy Gradient Algorithm

Algorithm 1 Simple MC Policy Gradient Algorithm

Set $j = 0, R_0 = 0, z_0 = \overline{0}, \Delta_0 = \overline{0}$
for each episode **do**
    for each transition s_t, a_t, r_t, s_{t+1} **do**
        $z_{t+1} = z_t + \frac{\nabla \pi(s_t, a_t; \Theta)}{\pi(s_t, a_t; \Theta)}$
        $R_{t+1} = R_t + \frac{1}{t+1}[r_t - R_t]$
    end for
    $\Delta_{j+1} = \Delta_j + R_T z_T$
    $j = j + 1$
end for
Return Δ_N / N, where N is the number of episodes

- The algorithm computes an unbiased estimate of the gradient
- Can be very slow due to high variance in the estimates
- Variance is related to the “recurrence time" or the episode length
- For problems with large state spaces, the variance becomes unacceptably high.

Deterministic Policy Gradient (Not part of this lecture)

- Problems with continuous action spaces
  - Hard to implement differentiable continuous controllers in many problems
  - Reduces the expectation to over only the states greatly simplifying the gradient estimation
  - Avoids max over actions
- Key difficulty: What is the notion of gradient for deterministic policies?
  - Continuous action spaces allow us to think of change in actions w.r.t. policy parameter
- Deterministic Policy Gradient Theorem
- <equation>
- Can be shown to be the limit of the Stochastic Policy Gradient Theorem in the limit the variance of the policy goes to zero
- Exploration?
  - If the environment is stochastic then not an issue
  - Otherwise use off-policy actor-critic, where the behavior policy differs from the estimation policy
  - Q – learning for value function updates
- Still require compatible parameterizations