
Spring 2018

Lecture 07. Reinforcement learning – part 3
Overview

• Project proposal presentations
• Reinforcement learning
  – Policy gradient
• References
  – Reinforcement learning – Sutton & Barto, Second edition, 2018
Learning a policy

• Dynamic programming
• Monte Carlo methods
• Temporal Difference (TD) learning
• Policy gradient

Tabular methods

Generalization methods
Policy gradient

\[ \pi(a \mid s, \theta) \]  
Parameterized policy

\[ \theta_{t+1} = \theta_t + \alpha \nabla J(\theta_t) \]  
Update parameters using gradient of performance w.r.t. \( \theta \)
Policy parameterization

• E.g., Softmax policy

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

where \( h(s, a, \theta) \) defines differentiable preferences such as:
  – Linear feature combination: \( h(s, a, \theta) = \theta^T x(s, a) \)
  – Deep neural network, e.g., AlphaGo

• Sutton & Barto 2nd ed. Ch 9.7 intro, Ch. 16 applications
Policy gradient theorem

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

On-policy state distribution: average time spent in state \( s \) under policy \( \pi \)

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

Performance gradient

\[ \mu(s) = \frac{\eta(s)}{\sum_{s'} \eta(s')} \]

Probability of episode beginning at state \( s \)

\[ \eta(s) = h(s) + \sum_{\bar{s}} \eta(\bar{s}) \sum_a \pi(a | \bar{s}) p(s | \bar{s}, a) \]

previous state
REINFORCE: Monte Carlo Policy Gradient

[Williams’92]

Goal: learn parameters $\theta$ that will give an optimal policy s.t. $\pi_\theta = \pi^*$.

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

Policy gradient theorem: gradient of performance is proportional to expected (sample) policy gradient
Expectation vs. Sampling

\[ V(S_t) \leftarrow E_\pi \left[ R_{t+1} + \gamma V(S_{t+1}) \right] = \sum_a \pi(a|S_t) \sum_{s', r|S_t, a} p(s', r|S_t, a)[r + \gamma V(s')] \]

\[ V(S_t) \leftarrow V(S_t) + \alpha [G_t - V(S_t)] \]

REINFORCE: Monte Carlo Policy Gradient

\[ \nabla J(\theta) \propto \sum_s \mu(s) \sum_a q_{\pi}(s,a) \nabla_{\theta} \pi(a \mid s, \theta) \]

\[ = E_\pi \left[ \sum_a q_{\pi}(S_t,a) \nabla_{\theta} \pi(a \mid S_t, \theta) \right] \]

Weighted sum over states

Sample state \( S_t \)
REINFORCE: Monte Carlo Policy Gradient

\[ \nabla J(\theta) \propto \sum_s \mu(s) \sum_a q_\pi(s, a) \nabla_\theta \pi(a \mid s, \theta) \]

\[ = E_\pi \left[ \sum_a q_\pi(S_t, a) \nabla_\theta \pi(a \mid S_t, \theta) \right] \]

\[ = E_\pi \left[ \sum_a \pi(a \mid S_t, \theta) q_\pi(S_t, a) \frac{\nabla_\theta \pi(a \mid S_t, \theta)}{\pi(a \mid S_t, \theta)} \right] \]

Weighted sum over states

Sample state \( S_t \)

Sum over actions under \( \pi \)

Sample action \( A_t \)

Sample return \( G_t \)
\[
\n\nabla J(\theta) \propto \sum_s \mu(s) \sum_a q_\pi(s, a) \nabla_\theta \pi(a | s, \theta)
\]

\[
= E_\pi \left[ \sum_a q_\pi(S_t, a) \nabla_\theta \pi(a | S_t, \theta) \right]
\]

\[
= E_\pi \left[ \sum_a \pi(a | S_t, \theta) q_\pi(S_t, a) \frac{\nabla_\theta \pi(a | S_t, \theta)}{\pi(a | S_t, \theta)} \right]
\]

\[
= E_\pi \left[ q_\pi(S_t, A_t) \frac{\nabla_\theta \pi(A_t | S_t, \theta)}{\pi(A_t | S_t, \theta)} \right]
\]

\[
= E_\pi \left[ G_t \frac{\nabla_\theta \pi(A_t | S_t, \theta)}{\pi(A_t | S_t, \theta)} \right]
\]
REINFORCE: Monte Carlo Policy Gradient

\[ \theta_{t+1} = \theta_t + \alpha G_t \frac{\nabla_{\theta_t} \pi(A_t | S_t, \theta_t)}{\pi(A_t | S_t, \theta_t)} \]

Move parameters to take \( A_t \) more often proportional to the return (i.e., towards direction of maximizing return)

Balance action coverage to avoid unfair advantage of frequent actions

\[ \frac{\nabla x}{x} = \theta_t + \alpha G_t \nabla_{\theta_t} \ln \pi(A_t | S_t, \theta_t) \]
REINFORCE: Monte Carlo Policy Gradient

\[ \theta_{t+1} = \theta_t + \alpha G_t \]

\[ \nabla \ln x = \frac{\nabla x}{x} \]

\[ = \theta_t + \alpha G_t \nabla_{\theta} \ln \pi(A_t | S_t, \theta_t) \]

Move parameters to take \( A_t \) more often proportional to the return (i.e., towards direction of maximizing return)

Balance action coverage to avoid unfair advantage of frequent actions

[Williams’92]
REINFORCE: Monte Carlo Policy Gradient

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 \( t = 0, \ldots, T - 1 \):
     - \( G \leftarrow \text{return from step } t \ (G_t) \)
     - \( \theta \leftarrow \theta + \alpha \gamma^t G \nabla_\theta \ln \pi(A_t|S_t, \theta) \)

[Williams’92]

[REINFORCE, A Monte-Carlo Policy-Gradient Method (episodic), for estimating \( \pi_\theta \approx \pi_* \)]

[Sutton & Barto ’18 Ch. 13.3]
Policy gradient advantages

• Approach deterministic policy whereas $\epsilon$-greedy always choose random actions with probability $\epsilon$
• Policy parameterization is generally simpler than action-value parameterization
• Whereas soft-max over action-values learn true values, action preferences approach optimal policy, e.g., 0.4, 0.6 vs. 0.01, 0.99
• Can be a good way to inject prior knowledge
Summary

• Reinforcement learning and MDP formulation
• Dynamic programming
• Sampling (Monte Carlo) methods
• Temporal difference learning
• On-policy vs. off-policy learning
• Policy gradient
• Next: Inverse reinforcement learning