Policy Gradient [2]

DOROZHKO Anton

Novosibirsk State University

June 6, 2020

[2] SpinningUP RL research : Part 3

Outline

1. Recap Q-learning
2. Policy Gradient

Q-learning

MDP $(S, A, \cancel{P}, \cancel{R}, \gamma)$

goal $\max_{\pi_\theta} \mathbb{E}[G_t]$

$\hat{Q}(s,a) = \mathbb{E}_\pi[G_t \mid a, s]$

$\downarrow$

$\pi(a \mid s) = \arg\max_a \hat{Q}(s,a)$ | ε-greedy

Action-Value approximation

target NN

- Find w that minimizes loss between $Q^{\pi}(s,a)$ and $\underline{\hat{Q}(s,a;w)}$
- MSE

$$J(w) = \mathbb{E}_{\pi}[(Q^{\pi}(s,a) - \hat{Q}(s,a;w))^2]$$

- SGD samples the gradient

$$-\frac{1}{2}\nabla_w J(w) = \mathbb{E}_{\pi}[(Q^{\pi}(s,a) - \hat{Q}(s,a;w))\nabla_w \hat{Q}^{\pi}(s,a;w)]$$

$$\Delta w = -\frac{1}{2}\alpha \nabla_w J(w)$$

Novosibirsk State University
*THE REAL SCIENCE

Incremental Model-Free Control

$r_0 + \gamma r_1 + \ldots + \gamma^i r_i$

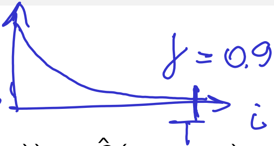
200 apm

5 minutes

1000 steps

50 steps

$0.9^{50} \sim 5 \cdot 10^{-3}$



$\gamma = 0.9$

$\gamma^i \approx 0$

- MC

$$\Delta w = \alpha(G_t - \hat{Q}^{\pi}(s_t, a_t; w))\nabla_w \hat{Q}(s_t, a_t; w)$$

- SARSA

$$\Delta w = \alpha(\underbrace{r + \gamma \hat{Q}(s', a'; w)} - \hat{Q}^{\pi}(s_t, a_t; w))\nabla_w \hat{Q}(s_t, a_t; w)$$

- Q-learning

$G_t = r_t + \gamma R_{t+1} + \ldots$

$$\Delta w = \alpha(r + \gamma \max_{a'} \hat{Q}(s', a'; w) - \hat{Q}^{\pi}(s_t, a_t; w))\nabla_w \hat{Q}(s_t, a_t; w)$$

$\gamma = 0$

$Q \sim r_t$

$\gamma \to 1$

$Q \approx r_t + r_{t+1} + \ldots r$

Novosibirsk State University *THE REAL SCIENCE

What if ?

$$\hat{Q}(s, a; w) \longrightarrow \pi$$

What if we can directly optimize our expected reward w.r.t. the parameters of our policy ?

$$\pi_\theta(a|s) \qquad J_\theta(\pi_\theta) = \mathbb{E}[G_t]$$

$$\nabla_\theta J_\theta(\pi)$$

Policy Optimization

- Stochastic, parametrized policy π_θ $\to f(\theta) : S \to \Delta A$
- maximize expected return

NN : $\cancel{x}(S) \to$

logits over action

$$J(\pi_\theta) = \mathbb{E}_{\tau\sim\pi_\theta}[G_t(\tau)]$$

τ trajectory

$(s_0, a_0, s_1 a_1)$

$r_0, r_1 \ldots.$

- SGD with **policy gradient**

$$\theta_{k+1} = \theta_k + \alpha \nabla_\theta J(\pi_\theta)|_{\theta_k}$$

How to compute policy gradient

- Derive the analitycal gradient
- Compute sample estimate

1) Probability of a Trajectory

ρ (to start at s_0)

$$s_0 \to a_0 \to s_1 \to a_1$$

P transition

Trajectory

$$\tau = (s_0, a_0, \ldots, s_{T+1})$$

Probability of a Trajectory

Markov property

$$P(\tau|\theta) = \rho_0(s_0) \prod_{t=0}^{T} P(s_{t+1}|s_t, a_t)\pi_\theta(a_t, s_t)$$

Take log

$$logP(\tau|\theta) =$$

$$\log \rho_0(s_0) + \sum_t \log P(s_{t+1}|s_t, a_t) + \sum_t \log \pi_\theta$$

2) The Log-Derivative Trick

$(\log x)' = \frac{1}{x}$

$\underbrace{\nabla_x \log f(x)}_{\text{easy}} = \frac{1}{f(x)} \underbrace{\nabla_x f(x)}_{?\ \text{hard}}$

$$\underline{\nabla_\theta log P(\tau|theta)} = \frac{\nabla_\theta P(\tau|\theta)}{P(\tau|\theta)}$$

Rearrange

$f(x)\ \nabla_x \log f(x)$

$$\nabla_\theta P(\tau|\theta) = \underline{P(\tau|\theta)}\nabla_\theta log P(\tau|\theta)$$

3) Gradients of Environment Functions

$$\nabla_\theta J(\pi_\theta) = \nabla_\theta \mathbb{E}_{\tau \sim \pi_\theta}[R(\tau)]$$

$$= \nabla_\theta \int R(\tau) \cdot P(\tau | \theta)$$

- $\rho_0(s_0)$, $P(s_{t+1}|s_t, a_t)$ and $G_t(\tau)$ $R(\tau)$
- no dependence on θ
- Gradients w.r.t θ are **zero**

$$\mathbb{E}_{x \sim p} f(x) = \int f(x) \cdot p(x)\, dx$$

Novosibirsk State University
*THE REAL SCIENCE

Grad-Log-Prob of Trajectory

$$
\begin{aligned}
\nabla_\theta \underline{\log P(\tau|\theta)} &= \cancelto{0}{\nabla_\theta \log \rho_0(s_0)} + \sum_{t=0}^{T} \Big(\cancelto{0}{\nabla_\theta \log P(s_{t+1}|s_t, a_t)} \\
&\quad + \nabla_\theta \log \pi_\theta(a_t|s_t) \Big) \\
&= \sum_{t=0}^{T} \underline{\nabla_\theta \log \pi_\theta(a_t|s_t)}.
\end{aligned}
$$

$\nabla_\theta J(\pi_\theta) = \dots$

$$G$$

$$\nabla_\theta \mathbb{E}_{\tau \sim \pi_\theta}[R(\tau)] = \nabla_\theta \int R(\tau) \cdot P(\tau|\theta)$$

$$= \int R(\tau) \nabla_\theta P(\tau|\theta)$$

$$= \int R(\tau) P(\tau|\theta) \nabla_\theta \log P(\tau|\theta)$$

$$= \mathbb{E}_{\tau \sim \pi_\theta}[R(\tau) \nabla_\theta \log P(\tau|\theta)]$$

$$= \mathbb{E}_{\tau \sim \pi}\left[R(\tau) \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t)\right]$$

Novosibirsk State University
*THE REAL SCIENCE

Basic Policy Gradient

Derivation for Basic Policy Gradient

$$
\begin{aligned}
\nabla_\theta J(\pi_\theta) &= \nabla_\theta \mathop{\mathrm{E}}_{\tau\sim\pi_\theta}[R(\tau)] && \text{def} \\
&= \nabla_\theta \int_\tau P(\tau|\theta)R(\tau) && \text{Expand expectation} \\
&= \int_\tau \nabla_\theta P(\tau|\theta)R(\tau) && \text{Bring gradient under integral} \\
&= \int_\tau P(\tau|\theta)\nabla_\theta \log P(\tau|\theta)R(\tau) && \text{Log-derivative trick} \\
&= \mathop{\mathrm{E}}_{\tau\sim\pi_\theta}[\nabla_\theta \log P(\tau|\theta)R(\tau)] && \text{Return to expectation form} \\
\therefore \nabla_\theta J(\pi_\theta) &= \mathop{\mathrm{E}}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T}\nabla_\theta \log \pi_\theta(a_t|s_t)R(\tau)\right] && \text{Expression for grad-log-prob}
\end{aligned}
$$

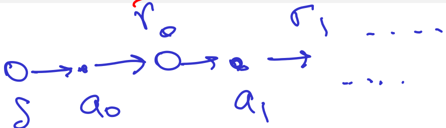
policy gradient

REINFORCE

Novosibirsk State University *THE REAL SCIENCE

How to compute expectation ?

$R(\tau) = r_0 + \gamma r_1 \ldots$



$$\mathbb{E}_{\tau \sim \pi_\theta}[\sum_{t=0}^{T} \nabla_\theta log \pi_\theta(a_t|s_t) R(\tau)]$$

$$\hat{g} = \frac{1}{N} \sum_{i=1}^{N} \sum_{t=0}^{T} \nabla_\theta log \pi_\theta(a_t|s_t) R(\tau)$$

$\nabla_\theta J$

N games with π_θ

$\theta_{k+1} = \theta_k - \alpha \nabla_\theta J$

Novosibirsk State University
*THE REAL SCIENCE

Basic Policy Gradient in words

$Q_w(s,a): S \times A \to \mathbb{R}$

$\pi_\theta(a|s): S \to \Delta A$

- Define policy as parametrized function
- Write the gradient of the loss function
- Play games collect trajectories (1 or more)
- Compute cumulative discounted rewards for each t $R(\tau)$
- Compute gradients $\nabla_\theta J(\pi_\theta)$
- after each step of SGD
- Update parameters of your policy $\theta_{k+1} = \theta_k - \alpha \nabla_\theta J$
- **Comment: throw out trajectories**

Novosibirsk State University
*THE REAL SCIENCE

Loss is not a normal loss

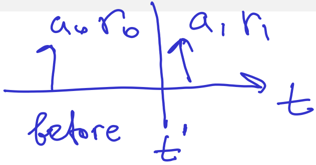
MSE loss

$$\frac{1}{N}\sum(f(x)-y)^2$$

1. Data is dependent on the parameters π_θ
2. Is not connected to performance $J(\pi_\theta)$ after one step
3. So, minimization of this "loss" itself for several steps on the same batch will not give better rewards.

Novosibirsk State University
*THE REAL SCIENCE

Consider only future rewards



for the whole game

$$\underline{\nabla_\theta J(\pi_\theta)} = \mathbb{E}_{\tau \sim \pi_\theta}[\sum_{t=0}^{T} \nabla_\theta log \pi_\theta(a_t|s_t) R(\tau)]$$

Consider only future rewards

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\tau \sim \pi_\theta}[\sum_{t=0}^{T} \nabla_\theta log \pi_\theta(a_t|s_t) R(\tau)]$$

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\tau \sim \pi_\theta}[\sum_{t=0}^{T} \nabla_\theta log \pi_\theta(a_t|s_t) \sum_{t'=t}^{T} \underline{R(s_{t'}, a_{t'}, s_{t'+1})}]$$

reward-to-go

[20] SpinningUP RL research : Part 3

Lemma

$$\mathbb{E}_{x \sim P_\theta}[\nabla_\theta log P_\theta(x)] = 0$$

Proof: distributions are normalized $\rightarrow$ take gradient $\rightarrow$ log derivative trick

$$\int_x P_\theta(x) = 1$$

$$\nabla_\theta \int_x P_\theta(x) = \nabla_\theta 1 = 0$$

$$\nabla_\theta \int_x P_\theta(x) = \int_x P_\theta(x) \nabla_\theta log P_\theta(x) = \underline{\mathbb{E}_{x \sim P_\theta}}[\nabla_\theta log P_\theta(x)] \quad \square$$

log deriv
trick

Proof of go-to-reward usage

$$\mathbb{E}_\tau \left[\sum \nabla_\theta \log \pi_\theta \sum_{t'=1}^{T} R \right] =$$

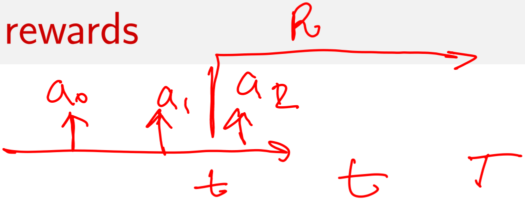
$$\mathbb{E}_\tau \left[\sum_t \nabla_\theta \log \pi_\theta \left[\sum_{t'=1}^{t-1} R + \sum_{t'=t}^{T} R \right] \right]$$

$$\nabla_\theta J_\theta(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}[\sum_{t=0}^{T} \nabla_\theta log \pi_\theta(a_t|s_t) \sum_{t'=t}^{T} R(s_{t'}, a_{t'}, s_{t'+1})]$$

$$= \sum_t \underbrace{\mathbb{E}_\tau \cancel{\nabla_\theta \log \pi_\theta}}_{0} \left[\sum_{t'=1}^{t-1} r \right] +$$

doesn't depend on τ after t

Consider only future rewards



$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\tau \sim \pi_\theta}[\sum_{t=0}^{T} \nabla_\theta log \pi_\theta(a_t|s_t) \sum_{t'=t}^{T} R(s_{t'}, a_{t'}, s_{t'+1})]$$

$\sum_{t=0}^{T} R \sim r.v \quad Var(R)$

Why it's better ?

Recall the sum of random variables with 0 mean and some variance.

Improvement: Baseline

Consequence of lemma

For $\forall b(s)$:

const w.r.t. a_t

$$\mathbb{E}_{a_t \sim \pi_\theta}[\nabla_\theta log \pi_\theta(a_t|s_t) b(s_t)] = 0$$

This allows:

$$\frac{\partial Var}{\partial b} = 0$$

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\tau \sim \pi_\theta}[\sum_{t=0}^{T} \nabla_\theta log \pi_\theta(a_t|s_t) \left(\overbrace{\sum_{t'=t}^{T} \underbrace{R(s_{t'}, a_{t'}, s_{t'+1})}_{\hat{Q}} - b(s_t)}^{Var} \right)]$$

We can choose b to reduce variance. In practice $b(s_t) = V^\pi(s_t)$ value function works well.

Basic Policy Gradient

Algorithm 1 Vanilla Policy Gradient Algorithm

Input: initial policy parameters θ_0, initial value function parameters ϕ_0

for $k = 0, 1, 2, \ldots$ **do**

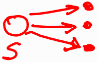
play

Collect set of trajectories $\mathcal{D}_k = \{\tau_i\}$ by running policy $\pi_k = \pi(\theta_k)$ in the environment.

Compute rewards-to-go $\hat{R}_t$.

$t \to T$

Compute advantage estimates, $\hat{A}_t$ (using any method of advantage estimation) based on the current value function V_{ϕ_k}.



$Q(s,a) = V(s) + A(a)$

const $\forall a$

Estimate policy gradient as

$$\hat{g}_k = \frac{1}{|\mathcal{D}_k|} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t)|_{\theta_k} \hat{A}_t.$$

101   +100   +1

102   +2

99   −1

Compute policy update, either using standard gradient ascent,

$$\theta_{k+1} = \theta_k + \alpha_k \hat{g}_k,$$

or via another gradient ascent algorithm like Adam.

Fit value function by regression on mean-squared error:

$$\phi_{k+1} = \arg\min_{\phi} \frac{1}{|\mathcal{D}_k| T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \left(V_\phi(s_t) - \hat{R}_t \right)^2,$$

typically via some gradient descent algorithm.

end for

Policy Gradient with automatic differentiation

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta log \pi_\theta(a_{i,t}|s_{i,t}) \hat{G}_{i,t}$$

How can we compute policy gradient with autodiff ?
Create a function s.t. it's gradient is policy gradient.
You will implement "pseudo-loss"

$$J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} log \pi_\theta(a_{i,t}|s_{i,t}) \hat{G}_{i,t}$$

cross entropy (discrete) or squared error (gaussian)

It is good if you can ...

1. Define Markov Decision Process
2. Describe some task in terms of MDP
3. Understand value and action-value functions
4. Be able to apply Policy Iteration and Value Iteration
5. Understand model-based vs model-free
6. Be able to apply MC, TD(0) algorithms to Q-function
7. Be able to apply Q-learning, SARSA, Expected value SARSA
8. Describe the "Exploitation / Exploration" dilemma
9. Understand ϵ-Greedy and "Optimism in the face of uncertainty" (UCB) ideas
10. Understand the idea of Policy Optimization (Policy Gradient)