ORCS 4529: Reinforcement Learning

Shipra Agrawal

Columbia University
Industrial Engineering and Operations Research

Contents I

Contents I

Contents I

Parameterizing the Policy space

$\pi_\theta : S \to \Delta^A$ with parameter vector $\theta \in \mathbb{R}^d$:

- $\pi_\theta(s)$ denotes a probability vector of dimension A with each component being the probability of taking the corresponding action.

Or, $\pi_\theta : S \times A \to \mathbb{R}$,

- $\pi_\theta(s, a)$ being the probability of taking action a, and $\sum_a \pi_\theta(s, a) = 1$.

For a scalable formulation, we want $d << |S|$.

Examples

Let x_s denote the features of state s.

- Softmax policy (N discrete actions): Parameters $\theta = (\theta_1, \dots, \theta_N)$ for N actions. Probability of playing action a in state s

$$\pi_\theta(s, a) = \frac{e^{f_{\theta_a}(x_s)}}{\sum_{a' \in A} e^{f_{\theta_{a'}}(x_s)}}$$

  where $f_\theta(x)$ is a function e.g., the outcome of a neural network

- Gaussian policy (continuous unrestricted action space): Distribution over actions given state s

$$\pi_\theta(s) = \mathcal{N}(f_\theta(s), \sigma^2)$$

  (Single mode distribution centered at one action)

Policy Optimization

Let Π_θ is a collection of all policies in a given parameteric class with parameter θ. Then, among all policies in the given parametric class, the policy that optimizes infinite horizon discounted value:

$$\max_\theta V^{\pi_\theta}(s_1)$$

Similarly for the average reward objective:

$$\max_\theta \rho^{\pi_\theta}(s_1)$$

Similarly for the Finite reward objective:

$$\max_\theta V_H^{\pi_\theta}(s_1)$$

Can we compute/estimate the gradient $\nabla_\theta V^{\pi_\theta}(s_1)$? (Or $\nabla_\theta V_H^{\pi_\theta}(s_1)$, $\nabla_\theta \rho^{\pi_\theta}(s_1)$ etc.)

Finite horizon MDP

Theorem

For finite horizon MDP (S, A, s_1, P, R, H)*, for a policy* π_θ*,*

$$\nabla_\theta V_H^{\pi_\theta}(s_1) = \mathbb{E}_\tau \left[R(\tau) \sum_{t=1}^{H} \nabla_\theta \log(\pi_\theta(s_t, a_t)) | s_1 \right]$$

where τ *denotes a random sample trajectory of states-actions*

$$\tau = (s_1, a_1, s_2, a_2, \ldots, s_H, a_H)$$

on starting from state s_1 *and following policy* π_θ *and* $R(\tau)$ *is the total expected discounted reward* $R(\tau) = \sum_{t=1}^{T} \gamma^{t-1} R(s_t, a_t)$ *on the given trajectory.*

Vanilla Policy gradient algorithm

aka REINFORCE [Williams 1988, Williams 1992

Initialize policy parameter θ_1, .

In each iteration $k = 1, 2, \ldots$,

- Execute current policy π^θ to obtain several sample trajectories τ^i, $i = 1, \ldots, m$ where

$$\tau^i = (s_1, a_1^i, s_2^i, \ldots, s_H^i, a_H^i), \quad \hat{R}(\tau^i) = r_1^i + \gamma r_2^i + \cdots + \gamma^{H-1} r_H^i$$

- Use these sample trajectories and chosen baseline to compute an unbiased gradient estimator $\hat{\mathbf{g}}$ using Policy gradient theorem

$$\hat{\mathbf{g}}_k = \frac{1}{m} \sum_{i=1}^{m} \hat{R}(\tau^i) \sum_{t=1}^{H} \nabla_\theta \log(\pi_\theta(s_t^i, a_t^i))$$

- Update $\theta_{k+1} \leftarrow \theta_k + \alpha_k \ \hat{\mathbf{g}}_k$
- Update baseline as required.

Vanilla Policy gradient algorithm

aka REINFORCE [Williams 1988, Williams 1992

Initialize policy parameter θ_1, and baseline.
In each iteration $k = 1, 2, \ldots,$

- Execute current policy π^θ to obtain several sample trajectories τ^i, $i = 1, \ldots, m$ where

$$\tau^i = (s_1, a_1^i, s_2^i, \ldots, s_H^i, a_H^i), \quad \hat{R}(\tau^i) = r_1^i + \gamma r_2^i + \cdots + \gamma^{H-1} r_H^i$$

- Use these sample trajectories and chosen baseline to compute an unbiased gradient estimator $\hat{\mathbf{g}}$ using Policy gradient theorem

$$\hat{\mathbf{g}}_k = \frac{1}{m} \sum_{i=1}^{m} \sum_{t=1}^{H} (\hat{R}(\tau^i) - b_t(s_t^i)) \nabla_\theta \log(\pi_\theta(s_t^i, a_t^i))$$

- Update $\theta_{k+1} \leftarrow \theta_k + \alpha_k \ \hat{\mathbf{g}}_k$
- Update baseline as required.

Baseline

Introducing a baseline does not change the expectation of gradient, but may improve variance if selected carefully.

An example of a good state-dependent baseline

$$b_t(s) = V_{H-t}^{\pi_\theta}(s)$$

i.e., the (Estimated) value of policy π_θ, starting from state s at time t. (Some more insights into this later)

Infinite horizon discounted rewards

The Policy optimization problem

$$\max_{\theta} V_{\gamma}^{\pi_\theta}(s_1)$$

where

$$V_{\gamma}^{\pi_\theta}(s_1) = \lim_{T\to\infty} \mathbb{E}[\sum_{t=1}^{T} \gamma^{t-1} r_t | s_1; a_t \sim \pi_\theta(s_t)]$$

Equivalently

$$V_{\gamma}^{\pi_\theta}(s_1) = \sum_{s} d^{\pi_\theta}(s) \sum_{a} \pi_\theta(s,a) R(s,a)$$

where $d^{\pi}(s) = \lim_{T\to\infty} \sum_{t=1}^{T} \gamma^{t-1} \Pr(s_t = s | s_1, \pi)$, the total discounted probability of being in state s under policy π.

Policy Gradient Theorem [Sutton, 1999]

Theorem

For infinite horizon MDP discounted reward case,

$$\nabla_\theta V_\gamma^{\pi_\theta}(s_1) = \sum_s d^{\pi_\theta}(s) \sum_a Q_\gamma^{\pi_\theta}(s,a) \nabla_\theta \pi_\theta(s,a)$$
$$= \frac{1}{(1-\gamma)} \mathbb{E}_{s\sim(1-\gamma)d^{\pi_\theta}, a\sim\pi_\theta(s)} \left[Q_\gamma^{\pi_\theta}(s,a) \nabla_\theta \log(\pi_\theta(s,a)) \right]$$

That is gradient of gain with respect to θ can be expressed in terms of gradient of (log of) policy function with respect to θ.

Remark: $(1-\gamma)d^\pi$ is a distribution over states, in particular $\sum_s (1-\gamma) d^\pi(s) = 1$

Remarks on Policy Gradient Theorem

- The key aspect of the expression for the policy gradient is that there are no terms of the form $\nabla_\theta d^{\pi_\theta}(s)$: the effect of policy changes on the (unknown) distribution over states does not appear.
- The distribution over actions given a state s is known, and its gradient $\nabla_\theta \log \pi_\theta(s, a)$ can be conveniently calculated e.g., by autodiff.
- The expectation is over the trajectories collected from the current policy which is convenient for approximating the gradient by sampling (on-policy).
- A difficulty compared to the finite horizon case is that current policy's Q-value $Q^{\pi_\theta}(s, a)$ is also not normally known, but it can be estimated e.g., by Monte Carlo or TD-learning.

Policy gradient estimation

Run policy π several times starting from s_1 to observe sample trajectories $\{\tau^i\}$ of length T for some large T (small γ^T)

- Q-value estimation using Monte Carlo method: at each time step t in a trajectory τ, set

$$\hat{Q}_t := \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'}$$

- Estimate of policy gradient from single trajectory (using Policy Gradient Theorem)

$$\hat{\mathbf{g}} = \sum_{t=1}^{T} \gamma^{t-1} \hat{Q}_t \nabla_\theta \log(\pi_\theta(s_t, a_t))$$

- Can add a baseline without introducing bias

$$\hat{\mathbf{g}} = \sum_{t=1}^{T} \gamma^{t-1} (\hat{Q}_t - b_t(s_t)) \nabla_\theta \log(\pi_\theta(s_t, a_t))$$

REINFORCE algorithm

Vanilla Policy gradient algorithm

Initialize policy parameter θ, .

In each iteration k,

1. Execute current policy π^{θ} to obtain several sample trajectories τ^{i}, $i = 1, \ldots, m$.

   For any given sample trajectory i, use observed rewards $r_1, r_2, \ldots$, to compute $\hat{Q}_t^i := \sum_{t'=t}^{T-1} \gamma^{t'-t} r_{t'}$.
2. Use $\hat{Q}_t^i$ and baseline function $b_t(s)$ to compute a gradient estimator $\hat{\mathbf{g}}_k$ using Policy gradient theorem.

$$\hat{\mathbf{g}} = \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \gamma^{t-1} \hat{Q}_t^i \, \nabla_\theta \log(\pi_\theta(s_t^i, a_t^i)) \tag{1}$$

   Update $\theta_{k+1} \leftarrow \theta_k + \alpha \hat{\mathbf{g}}_k$.

REINFORCE algorithm

Vanilla Policy gradient algorithm

Initialize policy parameter θ, and baseline function $b_t(s), \forall s$.
In each iteration k,

1. Execute current policy π^θ to obtain several sample trajectories τ^i, $i = 1, \ldots, m$.

   For any given sample trajectory i, use observed rewards $r_1, r_2, \ldots$, to compute $\hat{Q}_t^i := \sum_{t'=t}^{T-1} \gamma^{t'-t} r_{t'}$.
2. Use $\hat{Q}_t^i$ and baseline function $b_t(s)$ to compute a gradient estimator $\hat{\mathbf{g}}_k$ using Policy gradient theorem.

$$\hat{\mathbf{g}} = \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \gamma^{t-1} (\hat{Q}_t^i - b_t(s_t^i)) \nabla_\theta \log(\pi_\theta(s_t^i, a_t^i)) \qquad (1)$$

   Update $\theta_{k+1} \leftarrow \theta_k + \alpha \hat{\mathbf{g}}_k$.

   Re-optimize baseline.

REINFORCE algorithm

Vanilla Policy gradient algorithm

Initialize policy parameter θ, and baseline function $b_t(s), \forall s$.
In each iteration k,

1. Policy (π_θ) evaluation Execute current policy π^θ to obtain several sample trajectories τ^i, $i = 1, \ldots, m$.

   For any given sample trajectory i, use observed rewards $r_1, r_2, \ldots$, to compute $\hat{Q}_t^i := \sum_{t'=t}^{T-1} \gamma^{t'-t} r_{t'}$.
2. Policy Improvement Use $\hat{Q}_t^i$ and baseline function $b_t(s)$ to compute a gradient estimator $\hat{\mathbf{g}}_k$ using Policy gradient theorem.

$$\hat{\mathbf{g}} = \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \gamma^{t-1} (\hat{Q}_t^i - b_t(s_t^i)) \nabla_\theta \log(\pi_\theta(s_t^i, a_t^i)) \qquad (1)$$

   Update $\theta_{k+1} \leftarrow \theta_k + \alpha \hat{\mathbf{g}}_k$.

   Re-optimize baseline.

Variance reduction using baseline

Theorem

Let

$$A_t = (\hat{Q}_t - b_t(s_t))\nabla_\theta \log(\pi_\theta(s_t, a_t)).$$

the estimator of policy gradient Then, $Var(A_t|s_t)$ *is mimimized by base line:*

$$b_t(s_t) = \frac{\mathbb{E}\left(Q^{\pi_\theta}(s_t, a_t)\nabla_\theta \log(\pi_\theta(s_t, a_t))^2|s_t\right)}{\mathbb{E}\left(\log(\pi_\theta(s_t, a_t))^2|s_t\right)}$$

The value function $V^{\pi_\theta}(s) = \mathbb{E}\left(Q^{\pi_\theta}(s_t, a_t)|s_t\right)$ *is an approximation of the above optimal baseline.*

Baseline optimization

We use the following lemma:

Lemma

For any two random variables A, B*, such that for some filtration* $\mathcal{F}$, $\mathbb{E}[B|\mathcal{F}] = B, \mathbb{E}[A|\mathcal{F}] = B$*, almost surely,*

$$Var(A) = \mathbb{E}[(A - B)^2] + Var(B)$$

Contents I

Actor critic methods

- Actor-only methods (REINFORCE vanilla policy gradient) work with a parameterized family of policies. The gradientof the performance, with respect to the actor parameters, is directly estimated by simulation, and the parameters are updated in a direction of improvement. Maintain Policy Network.
- Critic-only methods (e.g., Q-learning, TD-learning) use TD-updates with function approximation to estimate optimal Q-values Q^* or Q-value of a given policy Q^π. Maintain Deep Q-network

Actor-critic algorithm

Maintain two networks: a policy network π_θ, and Q-network f_ω that estimates Q-value Q^{π_θ} of the current policy.

In iteration $k = 1, 2, 3, \ldots,$

- **Policy evaluation (train a critic network):** Use TD-learning (Q-learning with fixed policy) or Monte-Carlo to fit parameters ω such that $f_\omega(s,a) \approx Q^{\pi_{\theta_k}}(s,a)$
- **Policy improvement (Update the actor network):**

$$\theta_{k+1} \leftarrow \theta_k + \alpha_k \, \hat{\mathbb{E}}_{s\sim(1-\gamma)d^{\pi_k}, a\sim\pi_k(s)} \left[f_\omega(s,a) \nabla_\theta \log(\pi_{\theta_k}(s,a)) \right]$$

Q-network can also be used to estimate the current baseline (value function).

Contents I

Example: Tabular case

- $\theta \in \mathbb{R}^{S\times A}$ (Scores) Arg max policy:

$$\pi_\theta(s,a) = \frac{e^{\theta_{s,a}}}{\sum_{a'} e^{\theta_{s,a'}}}$$

$$\frac{\partial}{\partial \theta_{s',a'}} \log(\pi_\theta(s,a)) = \begin{cases} 1-\pi_\theta(s,a), & s,a = s',a', \\ -\pi_\theta(s,a'), & s=s', a\neq a' \\ 0 & otherwise \end{cases}$$

Example: Tabular case

- $\theta \in \mathbb{R}^{S \times A}$ (Scores) Arg max policy:

$$\pi_\theta(s,a) = \frac{e^{\theta_{s,a}}}{\sum_{a'} e^{\theta_{s,a'}}}$$

$$\frac{\partial}{\partial \theta_{s',a'}} \log(\pi_\theta(s,a)) = \begin{cases} 1 - \pi_\theta(s,a), & s,a = s',a', \\ -\pi_\theta(s,a'), & s = s', a \neq a' \\ 0 & \textit{otherwise} \end{cases}$$

- Policy evaluation step in REINFORCE: Estimate $\hat{Q}^{\pi_\theta}(s,a)$ using Monte Carlo method

Example: Tabular case

- $\theta \in \mathbb{R}^{S \times A}$ (Scores) Arg max policy:

$$\pi_\theta(s, a) = \frac{e^{\theta_{s,a}}}{\sum_{a'} e^{\theta_{s,a'}}}$$

$$\frac{\partial}{\partial \theta_{s',a'}} \log(\pi_\theta(s, a)) = \begin{cases} 1 - \pi_\theta(s, a), & s, a = s', a', \\ -\pi_\theta(s, a'), & s = s', a \neq a' \\ 0 & otherwise \end{cases}$$

- Policy evaluation step in REINFORCE: Estimate $\hat{Q}^{\pi_\theta}(s, a)$ using Monte Carlo method
- Policy improvement step in REINFORCE:

$$\theta_{k+1} \leftarrow \theta_k + \alpha_k \hat{\mathbb{E}}_{s,a \sim \pi_\theta} \left[\hat{Q}^{\pi_\theta}(s, a) \nabla_\theta \log(\pi_\theta(s, a)) \right]$$

Which is same as (approximate greedy policy improvement)

$$\theta_{k+1}(s, a) \approx \theta_k(s, a) + \alpha_k d^{\pi_\theta}(s) \pi_\theta(s, a) (Q^{\pi_\theta}(s, a) - V^{\pi_\theta}(s))$$

Example: Tabular case

- $\theta \in \mathbb{R}^{S\times A}$ (Scores) Arg max policy:

$$\pi_\theta(s,a) = \frac{e^{\theta_{s,a}}}{\sum_{a'} e^{\theta_{s,a'}}}$$

$$\frac{\partial}{\partial \theta_{s',a'}} \log(\pi_\theta(s,a)) = \begin{cases} 1-\pi_\theta(s,a), & s,a = s',a', \\ -\pi_\theta(s,a'), & s=s', a\neq a' \\ 0 & otherwise \end{cases}$$

- Policy evaluation step in REINFORCE: Estimate $\hat{Q}^{\pi_\theta}(s,a)$ using Monte Carlo method
- Policy improvement step in REINFORCE:

$$\theta_{k+1} \leftarrow \theta_k + \alpha_k \hat{\mathbb{E}}_{s,a\sim\pi_\theta}\left[\hat{Q}^{\pi_\theta}(s,a)\nabla_\theta \log(\pi_\theta(s,a))\right]$$

Which is same as (approximate greedy policy improvement)

$$\theta_{k+1}(s,a) \approx \theta_k(s,a) + \alpha_k d^{\pi_\theta}(s)\pi_\theta(s,a)(Q^{\pi_\theta}(s,a) - V^{\pi_\theta}(s))$$

Gradient is already normalized, no baseline needed

Connection to tabular policy iteration method

$$\theta_{k+1}(s,a) \approx \theta_k(s,a) + \alpha_k d^{\pi_\theta}(s)\pi_\theta(s,a)(Q^{\pi_\theta}(s,a) - V^{\pi_\theta}(s))$$

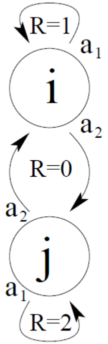
Moving towards but not jumping to

$$\arg\max_a Q^{\pi_\theta}(s,a)$$

Why not jump? Note that we only have estimates of $Q^{\pi_\theta}(s,a)$ for states and actions visited often under current policy.

Example illustrating difficulty in convergence

Kakade and Langford 2002



Assume initial policy is

$$\pi(i, a_1) = 0.8, \pi(i, a_2) = 0.2, \; \pi(j, a_1) = 0.2, \pi(j, a_2) = 0.8.$$

with stationary distribution $p(i) = 0.8, p(j) = 0.2$.
Optimal policy $\pi^*(i, a_2) = 1, \pi^*(j, a_1) = 1$.

Example cont.

Figure from Kakade and Langford, 2002

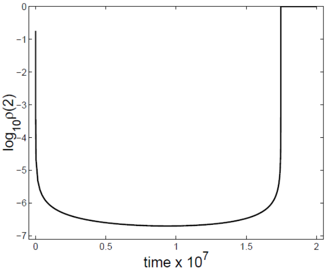


Figure: Stationary probability of state j under policy gradient algorithm

What happens under policy gradient?

$$\theta_{k+1}(s,a) \approx \theta_k(s,a) + \alpha_k d^{\pi_\theta}(s)\pi_\theta(s,a)(Q^{\pi_\theta}(s,a) - V^{\pi_\theta}(s))$$

- Suppose the current policy favors a_1 on state i and a_2 on state j (like the initial policy),

What happens under policy gradient?

$$\theta_{k+1}(s,a) \approx \theta_k(s,a) + \alpha_k d^{\pi_\theta}(s)\pi_\theta(s,a)(Q^{\pi_\theta}(s,a) - V^{\pi_\theta}(s))$$

- Suppose the current policy favors a_1 on state i and a_2 on state j (like the initial policy),
- For both states $Q^{\pi_\theta}(s,a_1)$ is more than $Q^{\pi_\theta}(s,a_2)$ (because immediate reward is 0 for a_2).

What happens under policy gradient?

$$\theta_{k+1}(s,a) \approx \theta_k(s,a) + \alpha_k d^{\pi_\theta}(s)\pi_\theta(s,a)(Q^{\pi_\theta}(s,a) - V^{\pi_\theta}(s))$$

- Suppose the current policy favors a_1 on state i and a_2 on state j (like the initial policy),
- For both states $Q^{\pi_\theta}(s,a_1)$ is more than $Q^{\pi_\theta}(s,a_2)$ (because immediate reward is 0 for a_2).
- For state i, $\pi_\theta(i,a_1)$ is also more than a_2, so $\theta(i,a_1)$ increases more than a_2 (goes farther from optimal)

What happens under policy gradient?

$$\theta_{k+1}(s,a) \approx \theta_k(s,a) + \alpha_k d^{\pi_\theta}(s)\pi_\theta(s,a)(Q^{\pi_\theta}(s,a) - V^{\pi_\theta}(s))$$

- Suppose the current policy favors a_1 on state i and a_2 on state j (like the initial policy),
- For both states $Q^{\pi_\theta}(s,a_1)$ is more than $Q^{\pi_\theta}(s,a_2)$ (because immediate reward is 0 for a_2).
- For state i, $\pi_\theta(i,a_1)$ is also more than a_2, so $\theta(i,a_1)$ increases more than a_2 (goes farther from optimal)
- For state j, even though $Q^{\pi_\theta}(s,a_1)$ favors a_1, $\pi_\theta(j,a_2)$ is more. So there might be no or small improvement in favor of a_1.

What happens under policy gradient?

$$\theta_{k+1}(s,a) \approx \theta_k(s,a) + \alpha_k d^{\pi_\theta}(s)\pi_\theta(s,a)(Q^{\pi_\theta}(s,a) - V^{\pi_\theta}(s))$$

- Suppose the current policy favors a_1 on state i and a_2 on state j (like the initial policy),
- For both states $Q^{\pi_\theta}(s,a_1)$ is more than $Q^{\pi_\theta}(s,a_2)$ (because immediate reward is 0 for a_2).
- For state i, $\pi_\theta(i,a_1)$ is also more than a_2, so $\theta(i,a_1)$ increases more than a_2 (goes farther from optimal)
- For state j, even though $Q^{\pi_\theta}(s,a_1)$ favors a_1, $\pi_\theta(j,a_2)$ is more. So there might be no or small improvement in favor of a_1.
- Also $d^\pi(j) < d^\pi(i)$, so any improvement for state j may be overshadowed by the decline for state i.

How does it compare to the greedy policy improvement?

Greedy policy improvement

$$\pi(s) = \arg\max_a Q^{\pi_\theta}(s, a)$$

(Not necessarily a good idea if we don't have good estimates of $Q^{\pi_\theta}(s, a)$ for s, a not visited)

- In the given example, $\pi(j, a_1) = 1, \pi(j, a_2) = 0$ after the first iteration.
- And, for the new policy $Q^\pi(i, a_2) > Q^\pi(i, a_1)$, so that $\pi(i, a_2) = 1, \pi(i, a_1) = 0$ in second iteration.

Optimal policy in two iterations.

Approximately Optimal Approximate RL [Kakade and Langford 2002]

Can we design an algorithm that is guaranteed to improve some performance measure at every step?

Conservative Greedy policy improvement algorithm

Main idea: Move to the greedy policy but not fully: New policy and old policy are same with probability α.

If current policy is π_k, conservative policy improvement:

$$\pi^{k+1} \leftarrow (1-\alpha)\pi^k + \alpha\pi_*^{k+1}$$

where π_*^{k+1} is the greedy policy

$$\pi_*^{k+1}(s) = \arg\max_a Q^{\pi_k}(s,a)$$

Regular policy iteration has $\alpha = 1$.

Conservative Greedy policy improvement algorithm

Main idea: Move to the greedy policy but not fully: New policy and old policy are same with probability α.

If current policy is π_k, conservative policy improvement:

$$\pi^{k+1} \leftarrow (1-\alpha)\pi^k + \alpha\pi_*^{k+1}$$

where π_*^{k+1} is the greedy policy

$$\pi_*^{k+1}(s) == \arg\max_a \quad \underbrace{Q^{\pi_k}(s,a) - V^{\pi_k}(s)}_{Advantage\ \ function\ \ A^{\pi_k}(s,a)}$$

Regular policy iteration has $\alpha = 1$.

Generalizing this idea to large state space

If current policy parameter is θ_k, conservative greedy policy improvement:

$$\pi^{k+1} \leftarrow (1-\alpha)\pi^k + \alpha\pi_{\theta^*_{k+1}}$$

where

$$\theta^*_{k+1}(s) = \arg\max_{\theta} \mathbb{E}_{s\sim(1-\gamma)d^{\pi_k}}[\sum_a \pi_\theta(s,a)A^{\pi_k}(s,a)]$$

That is, $\pi_{\theta^*_{k+1}}$ is an approximate greedy policy – close to greedy in expectation over states.

Generalizing this idea to large state space

If current policy parameter is θ_k, conservative greedy policy improvement:

$$\pi^{k+1} \leftarrow (1-\alpha)\pi^k + \alpha\pi_{\theta^*_{k+1}}$$

where

$$\theta^*_{k+1}(s) = \arg\max_{\theta} \mathbb{E}_{s\sim(1-\gamma)d^{\pi_k}}[\sum_a \pi_{\theta}(s,a)A^{\pi_k}(s,a)]$$

That is, $\pi_{\theta^*_{k+1}}$ is an approximate greedy policy – close to greedy in expectation over states.

After k iterations, the policy π^k is a randomized policy which plays policy π^i with probability $(1-\alpha)^{i-1}\alpha$.

Trust Region Policy Optimization (TRPO)

(To be disucssed later)

Fit a θ so that π_θ is close to greedy in expectation over states, *and* is not too far from the current policy.

$$\theta_{k+1} = \begin{array}{cc} \arg\max_{\theta} & \mathbb{E}_{s\sim(1-\gamma)d^{\pi_{\theta_k}}}\left[\sum_a \pi_\theta(s,a)A^{\pi_{\theta_k}}(s,a)\right] \\ s.t. & \max_s KL(\pi_\theta(s)\|\pi_{\theta_k}(s)) \leq \alpha \end{array}$$

This is the popular TRPO (Trust Region Policy Optimization algorithm) [Schulman et al. 2015]

Conservative Greedy Policy Improvement

Lemma 4.1 of Kakade and Langford, 2002

Let π be the current policy, π' be the greedy policy,

$$\pi' = \arg\max_{\hat{\pi} \in \Pi} A_\pi(\hat{\pi}) := \mathbb{E}_{s \sim (1-\gamma) d^\pi} [\sum_a \hat{\pi}(s, a) A^\pi(s, a)]$$

and π^{new} is conservative greedy policy which is same as π with probability $1 - \alpha$.

Theorem (Policy improvement theorem)

$$V^{\pi^{new}}(s_1) - V^\pi(s_1) \geq \frac{\alpha}{1-\gamma} A_\pi(\pi') - \frac{\alpha^2}{(1-\gamma)^2} 2\gamma A_\pi^{\max}$$

where $A_\pi^{\max}$ is the maximum advantage over all states:

$$A_\pi^{\max} = \max_s \left| \sum_a \pi'(s, a) A^\pi(s, a) \right|$$

Intuitive proof sketch for policy improvement theorem

using policy gradient theorem

Given π, π', consider the set of policies

$$\pi_\alpha = (1-\alpha)\pi + \alpha\pi'$$

for all $\alpha \in (0, 1)$.

Note that $\pi^{new} = \pi_\alpha, \pi = \pi_0, \pi' = \pi_1$.

How much does the value of policy V^{π_α} change if we change the policy parameter from 0 to α? Policy gradient theorem!!

Intuitive proof through policy gradient

By policy gradient theorem, policy gradient at $\alpha = 0$

$$\nabla_\alpha V^{\pi_\alpha}(s_1)\Big|_{\alpha=0} = \sum_s d^{\pi_\alpha}(s) \sum_a \left(\nabla_\alpha \pi_\alpha(s,a)\right) A^{\pi_\alpha}(s,a)\Big|_{\alpha=0}$$

where $\pi_0 = \pi$, and

$$\nabla_\alpha \pi_\alpha(s,a) = \pi'(s,a) - \pi(s,a)$$

Intuitive proof through policy gradient

By policy gradient theorem, policy gradient at $\alpha = 0$

$$\nabla_\alpha V^{\pi_\alpha}(s_1)\Big|_{\alpha=0} = \sum_s d^{\pi_\alpha}(s) \sum_a \left(\nabla_\alpha \pi_\alpha(s,a)\right) A^{\pi_\alpha}(s,a)\Big|_{\alpha=0}$$

where $\pi_0 = \pi$, and

$$\nabla_\alpha \pi_\alpha(s,a) = \pi'(s,a) - \pi(s,a)$$

$$\begin{aligned}\nabla_\alpha V^{\pi_\alpha}(s_1)\Big|_{\alpha=0} &= \sum_s d^{\pi}(s) \sum_a \left(\pi'(s,a) - \pi(s,a)\right) A^{\pi}(s,a) \\ &= \sum_s d^{\pi}(s) \sum_a \pi'(s,a) A^{\pi}(s,a) =: \frac{A_\pi(\pi')}{1-\gamma}\end{aligned}$$

Intuitive proof through policy gradient

By policy gradient theorem, policy gradient at $\alpha = 0$

$$\nabla_\alpha V^{\pi_\alpha}(s_1)\Big|_{\alpha=0} = \sum_s d^{\pi_\alpha}(s) \sum_a \left(\nabla_\alpha \pi_\alpha(s,a)\right) A^{\pi_\alpha}(s,a)\Big|_{\alpha=0}$$

where $\pi_0 = \pi$, and

$$\nabla_\alpha \pi_\alpha(s,a) = \pi'(s,a) - \pi(s,a)$$

$$\begin{aligned} \nabla_\alpha V^{\pi_\alpha}(s_1)\Big|_{\alpha=0} &= \sum_s d^{\pi}(s) \sum_a \left(\pi'(s,a) - \pi(s,a)\right) A^{\pi}(s,a) \\ &= \sum_s d^{\pi}(s) \sum_a \pi'(s,a) A^{\pi}(s,a) =: \frac{A_\pi(\pi')}{1-\gamma} \end{aligned}$$

Then, for small enough α, the lemma we want to prove follows (roughly) from Taylor approximation ($\pi_\alpha = \pi_{new}, \pi_0 = \pi$).

$$V^{\pi_{new}}(s_1) - V^{\pi}(s_1) \geq \alpha \frac{A_\pi(\pi')}{1-\gamma} - O(\alpha^2)$$

Proof of Policy improvement lemma

Lemma (Policy improvement lemma)

$$V^{\pi^{new}}(s_1) - V^{\pi}(s_1) \geq \frac{\alpha}{(1-\gamma)} A_{\pi}(\pi') - \frac{\alpha^2}{(1-\gamma)^2} 2\gamma A_{\pi}^{\max}$$

$$A_{\pi}(\pi') = \mathbb{E}_{s\sim(1-\gamma)d^{\pi}}[\sum_a \pi'(s,a) A^{\pi}(s,a)]$$

Proof of Policy improvement lemma

Lemma (Policy improvement lemma)

$$V^{\pi^{new}}(s_1) - V^{\pi}(s_1) \geq \frac{\alpha}{(1-\gamma)} A_\pi(\pi') - \frac{\alpha^2}{(1-\gamma)^2} 2\gamma A_\pi^{\max}$$

$$A_\pi(\pi') = \mathbb{E}_{s\sim(1-\gamma)d^\pi}[\sum_a \pi'(s,a)A^\pi(s,a)]$$

Proof Outline:

- A "Performance Difference Lemma" characterizes the exact difference in in the two value functions, but involves new state distribution $d^{\pi_{new}}$.
- Proof of "policy improvement lemma" is by showing that new state distribution $d^{\pi_{new}}$ is close to the old state distribution d^π.

Performance difference Lemma

Lemma 6.1 of Kakade and Langford 2002

Following characterizes the exact change in value of policy

Lemma (Performance difference Lemma)

For any two policies π^{new}, π,

$$
\begin{aligned}
V^{\pi^{new}}(s_1) - V^{\pi}(s_1) &= \sum_s d^{\pi^{new}}(s) \sum_a \pi^{new}(s,a) A^{\pi}(s,a) \\
&= \frac{\alpha}{(1-\gamma)} \mathbb{E}_{s \sim (1-\gamma) d^{\pi^{new}}} \left[\sum_a \pi'(s,a) A^{\pi}(s,a) \right]
\end{aligned}
$$

Policy improvement lemma proof is by comparing the distribution over states under the new and old policies: $d^{\pi^{new}}$ and d^{π}.

How is the policy improvement lemma useful?

Algorithm Design

Important: $A_\pi(\pi')$ can be estimated by simulation/sampling from current policy!

We can select a step size to always have a positive improvement as long as $A_\pi(\pi') > 0$. Let R be an upper bound on rewards, so that $A_\pi(\pi') \leq A_\pi^{\max} \leq \frac{R}{1-\gamma}$. Then, setting

$$\alpha = \frac{A_\pi(\pi')(1-\gamma)^2}{4R},$$

and substituting in policy improvement lemma 6, we get

$$V(\pi^{new}) - V(\pi) \geq \frac{\alpha}{(1-\gamma)} A_\pi(\pi') - \frac{2\alpha^2 A_\pi^{\max}}{(1-\gamma)^2} \geq \frac{A_\pi(\pi')^2(1-\gamma)}{8R} \tag{2}$$

Conservative policy improvement algorithm

Initialize π. Repeat:

1. (Policy evaluation) Play policy π from starting state distribution μ to generate sample trajectories.
2. Estimate advantage function $\hat{A}^{\pi}(s, a)$, e.g., by Monte Carlo or TD-learning. Compute

$$\hat{A} := \max_{\pi' \in \Pi} \hat{\mathbb{E}}_{s \sim (1-\gamma) d^{\pi,\mu}} \left[\sum_a \pi'(s, a) \hat{A}^{\pi}(s, a) \right]$$

   with π' be the arg max policy in the above.

3. If $\hat{A}$ is very small , STOP.
4. (Policy improvement) Update policy:

$$\pi \leftarrow (1-\alpha)\pi + \alpha\pi'$$

$$\text{where } \alpha = \left(\hat{A}\right) \frac{(1-\gamma)^2}{4R}$$

Conservative policy improvement algorithm

Initialize π. Repeat:

1. (Policy evaluation) Play policy π from starting state distribution μ to generate sample trajectories.
2. Estimate advantage function $\hat{A}^{\pi}(s, a)$, e.g., by Monte Carlo or TD-learning. Compute

$$\hat{A} := \max_{\pi' \in \Pi} \hat{\mathbb{E}}_{s \sim (1-\gamma) d^{\pi,\mu}} \left[\sum_a \pi'(s, a) \hat{A}^{\pi}(s, a) \right]$$

   with π' be the arg max policy in the above. Assume guarantee $\hat{A} \geq A_{\pi}(\pi') - \frac{\delta}{3}$
3. If $\hat{A}$ is very small ($\hat{A} < \frac{2\delta}{3}$), STOP.
4. (Policy improvement) Update policy:

$$\pi \leftarrow (1 - \alpha)\pi + \alpha \pi'$$

$$\text{where } \alpha = \left(\hat{A} - \frac{\delta}{3} \right) \frac{(1-\gamma)^2}{4R}$$

Conservative policy improvement algorithm

Lemma

The conservative greedy policy improvement algorithm terminates in at most $\frac{72R^2}{\delta^2(1-\gamma)^3}$ iterations to find a policy π such that

$$\max_{\pi'} A_\pi(\pi') \leq \delta$$

How good is the policy? local improvement vs. optimality

Theorem (Theorem 6.2 of Kakade and Langford 2002)

Let $(1-\gamma)d^{\pi,\mu}$ denote the discounted state distribution for policy π when starting state distribution is μ. We run our algorithm using starting state distribution μ and obtain a policy π with

$$\max_{\pi'} A_{\pi,\mu}(\pi') \le \delta$$

where $A_\pi(\pi') = \mathbb{E}_{s\sim(1-\gamma)d^{\pi,\mu}(s)}[\sum_a \pi'(s,a)A^\pi(s,a)]$. Then, for any policy π^* and starting state distribution μ^*,

$$\begin{aligned}\mathbb{E}_{s\sim\mu^*}[V^{\pi^*}(s) - V^\pi(s)] &\le \frac{\delta}{1-\gamma}\left\|\frac{d^{\pi^*,\mu^*}}{d^{\pi,\mu}}\right\|_\infty \\ &\le \frac{\delta}{(1-\gamma)}\left\|\frac{d^{\pi^*,\mu^*}}{\mu}\right\|_\infty\end{aligned}$$

TRPO [Schulman et al., ICML 2015]

- Provides policy improvement guarantees similar to the Conservative greedy method
- Gets rid of unwieldy mixture policies
- In every iteration moves to a new policy within α distance of the old policy.

Some definitions

Distance between two distributions p, q:

$$D_{TV}(p||q) = \frac{1}{2}\sum_i |p_i - q_i|$$

$$D_{KL}(p||q) = \sum_i p_i \log \frac{p_i}{q_i}$$

Known result:

$$D_{TV}(p||q)^2 \leq D_{KL}(p||q)$$

Distance between two policies π^{new}, π^{old}:

$$D_{TV}^{\max}(\pi^{old}, \pi^{new}) = \max_s D_{TV}(\pi^{old}(\cdot|s), \pi^{new}(\cdot|s))$$

$$D_{KL}^{\max}(\pi^{old}, \pi^{new}) = \max_s D_{KL}(\pi^{old}(\cdot|s), \pi^{new}(\cdot|s))$$

New theorem for bounding policy improvement

[Schulman eta al. 2015]

Theorem

Let $\alpha = D_{TV}^{\max}(\pi, \tilde{\pi})$. Then, the following bound holds for any starting state s_1:

$$V^{\tilde{\pi}}(s_1) - V^{\pi}(s_1) \geq \frac{1}{(1-\gamma)} A_{\pi}(\tilde{\pi}) - \frac{\alpha^2}{(1-\gamma)^2} 4\gamma\epsilon$$

where

$$A_{\pi}(\tilde{\pi}) = \mathbb{E}_{s \sim (1-\gamma) d^{\pi}} [\sum_a \tilde{\pi}(s,a) A^{\pi}(s,a)]$$

$$\epsilon = \max_{s,a} |A^{\pi}(s,a)|$$

TRPO algorithm design

In each iteration $k+1$:

$$\theta_{k+1} = \begin{array}{ll} \arg\max_{\theta} & \mathbb{E}_{s \sim \pi_{\theta_k}} \left[\sum_a \pi_{\theta}(s,a) A^{\pi_{\theta_k}}(s,a)\right] \\ s.t. & \mathbb{E}_{s \sim \pi_{\theta_k}} \left[D_{KL}(\pi_{\theta}(\cdot|s), \pi_{\theta_k}(\cdot|s))\right] \leq \delta \end{array}$$

- Relaxes $D_{KL}^{\max}$ to expected KL divergence over states sampled from old policy.

Some implementation details

- Monte Carlo estimates of $A^{\pi_{\theta_k}}(s, a)$ or $Q^{\pi_{\theta_k}}(s, a)$], and expectation terms from sample trajectories generated from π_θ.

Some implementation details

- Monte Carlo estimates of $A^{\pi_{\theta_k}}(s,a)$ or $Q^{\pi_{\theta_k}}(s,a)]$, and expectation terms from sample trajectories generated from π_θ.
- 'Vine" simulation and Importance sampling
  - multiple alternate actions can be tried from the observed states in simulation settings.
  - objective replaced by

$$\mathbb{E}_{s\sim\pi_{\theta_k}, a\sim q}\left[\frac{\pi_{\theta}(s,a)}{q(s,a)} A^{\pi_{\theta_k}}(s,a)\right]$$

Some implementation details

- Monte Carlo estimates of $A^{\pi_{\theta_k}}(s,a)$ or $Q^{\pi_{\theta_k}}(s,a)]$, and expectation terms from sample trajectories generated from π_θ.
- 'Vine" simulation and Importance sampling
  - multiple alternate actions can be tried from the observed states in simulation settings.
  - objective replaced by

$$\mathbb{E}_{s\sim\pi_{\theta_k},a\sim q}\left[\frac{\pi_{\theta}(s,a)}{q(s,a)}A^{\pi_{\theta_k}}(s,a)\right]$$

- Conjugate gradient method for constrained optimization
  - Requires computing Hessian of the KL divergence term.
  - Several approximations proposed for making it efficient.

Proximal Policy Optimization (PPO)

Schulman et al 2017

Recall TRPO:

$$\theta_{k+1} = \begin{array}{rl} \arg\max_{\theta} & \mathbb{E}_{s\sim\pi_{\theta_k}}\left[\sum_a \pi_{\theta}(s,a) A^{\pi_{\theta_k}}(s,a)\right] \\ s.t. & \mathbb{E}_{s\sim\pi_{\theta_k}}\left[D_{KL}(\pi_{\theta}(\cdot|s), \pi_{\theta_k}(\cdot|s))\right] \leq \delta \end{array}$$

Proximal Policy Optimization (PPO)

Schulman et al 2017

Recall TRPO:

$$\theta_{k+1} = \begin{array}{rl} \arg\max_{\theta} & \mathbb{E}_{s\sim\pi_{\theta_k}}[\sum_a \pi_{\theta}(s,a)A^{\pi_{\theta_k}}(s,a)] \\ s.t. & \mathbb{E}_{s\sim\pi_{\theta_k}}[D_{KL}(\pi_{\theta}(\cdot|s),\pi_{\theta_k}(\cdot|s))] \le \delta \end{array}$$

Equivalently

$$\theta_{k+1} = \begin{array}{rl} \arg\max_{\theta} & \mathbb{E}_{s,a\sim\pi_{\theta_k}}[\frac{\pi_{\theta}(s,a)}{\pi_{\theta_k}(s,a)}A^{\pi_{\theta_k}}(s,a)] \\ s.t. & \mathbb{E}_{s,a\sim\pi_{\theta_k}}\left[\frac{\pi_{\theta}(s,a)}{\pi_{\theta_k}(s,a)}\log(\frac{\pi_{\theta}(s,a)}{\pi_{\theta_k}(s,a)})\right] \le \delta \end{array}$$

PPO algorithm design

For a sample state action pair s^i, a^i, let ratio

$$r_i(\theta) := \frac{\pi_\theta(s^i, a^i)}{\pi_{\theta_k}(s^i, a^i)}$$

PPO replaces the trust region constrained by clipped ratio in the objective and solve the unconstrained problem:

$$\theta_{k+1} = \max_\theta \hat{E}_i \left[\min\left(r_i(\theta)\hat{A}_i, \text{clip}(r_i(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_i\right)\right]$$

where $\hat{E}$ is expectation over state and action pairs s^i, a^i generated from policy π_{θ_k}, and $\hat{A}^i$ is a Monte Carlo estimate of $A^{\pi_{\theta_k}}(s_i, a_i)$.

PPO algorithm

Initialize θ^1. In iteration k

- Generate sample trajectories from π_{θ_k}.
- For each sample s^i, a^i in the trajectories, construct Monte Carlo estimate $\hat{A}^i$ for $A^{\pi_{\theta_k}}(s_i, a_i)$.
- Let $r_i(\theta) := \frac{\pi_\theta(s^i,a^i)}{\pi_{\theta_k}(s^i,a^i)}$.
- Solve

$$\theta_{k+1} = \max_\theta \hat{E}_i \left[\min\left(r_i(\theta)\hat{A}_i, \text{clip}(r_i(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_i\right)\right]$$

Comparisons on MuJoCo

Figure from [Schulman et al. 2017]

PPO used with $\epsilon = 0.2$

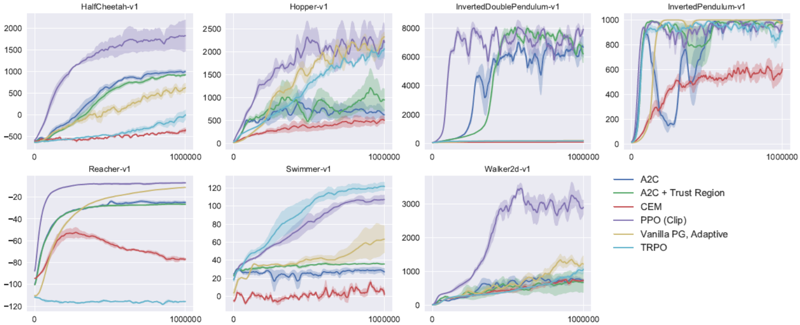


Figure 3: Comparison of several algorithms on several MuJoCo environments, training for one million timesteps.

Other related methods

- Soft policy iteration: softmax instead of greedy policy update. Project back to policy space using KL divergence

$$\pi^{soft-greedy}(\cdot|s^i) \propto \exp(\hat{Q}^i)$$

$\hat{Q}^i$ is a Monte Carlo estimate of $Q^{\pi_{\text{old}}}(s^i, a^i)$

$$\pi^{new} = \arg\min_{\pi' \in \Pi} \hat{\mathbb{E}}_{s^i \sim \pi^{old}} \left[D_{KL}(\pi'(\cdot|s^i) \parallel \pi^{soft-greedy}(\cdot|s^i)) \right]$$

- Soft actor-critic [Harnooja et al. 2018]: (Deep) Q-learning/TD learning to estimate Q-values

$$\pi^{new} = \arg\min_{\pi' \in \Pi} \hat{\mathbb{E}}_{s^i \sim \pi^{old}} \left[D_{KL}(\pi'(\cdot|s^i) \parallel \frac{\exp(Q_\theta(s^i, a^i))}{Z_\theta}) \right]$$

Gradient descent methods for optimizing the above objective. Removes the need to estimate normalization constant Z_θ.