Research Seminar

Policy gradient methods. Policy gradient theorem. Log-derivative trick. Baselines. Actor critic

Artur Goldman

Environment

Environment

$s \in \mathcal{S}$ - **state** space

Environment

$s \in \mathcal{S}$ - **state** space

$a \in \mathcal{A}$ - **action** space

Environment

$s \in \mathcal{S}$ - **state** space

$a \in \mathcal{A}$ - **action** space

$r \in \mathcal{R}$ - **reward** space

Goal

Goal

The goal is to maximize total reward

Interaction with environment

Interaction with environment

$t \in \{0, 1, \ldots, T\}$ - set of decision epochs

Interaction with environment

$t \in \{0, 1, \ldots, T\}$ - set of decision epochs

Interaction sequence is described by one **episode** (a.k.a **trajectory**) and the sequence ends at the terminal state S_T:

Interaction with environment

$t \in \{0, 1, \ldots, T\}$ - set of decision epochs
Interaction sequence is described by one **episode** (a.k.a **trajectory**) and the sequence ends at the terminal state S_T:

$$\tau = (S_0, A_0, R_1, S_1, A_1, \ldots, R_T)$$

Model: transition

Model: transition

Transition step is represented by tuple

$$(s, a, s', r)$$

Model: transition

Transition step is represented by tuple

$$(s, a, s', r)$$

Specific transition happens with probability

$$P(s', r|s, a) =$$

$$= P(S_{t+1} = s', R_{t+1} = r|S_t = s, A_t = a)$$

Policy

Policy

Deterministic: $\pi(s) = a$

Policy

Deterministic: $\pi(s) = a$

Stochastic: $\pi(a|s) = P_\pi(A = a|S = s)$

Value function

Value function

Future reward a.k.a **return**

Value function

Future reward a.k.a **return**

$$G_t = R_{t+1} + \gamma R_{t+2} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

Value function

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

Value function

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

State-value of state s at time t

$$V_\pi(s) = E_\pi[G_t | S_t = s], \ \forall t = 0, 1, 2, \ldots$$

Value function

State-value of state s at time t

$$V_\pi(s) = E_\pi[G_t | S_t = s], \ \forall t = 0, 1, 2, \ldots$$

Value function

State-value of state s at time t

$$V_\pi(s) = E_\pi[G_t | S_t = s], \ \forall t = 0, 1, 2, \ldots$$

Action-value of state-action pair at time t

$$Q_\pi(s, a) = E_\pi[G_t | S_t = s, A_t = a], \ \forall t = 0, 1, 2, \ldots$$

Common methods

Dynamic Programming

Monte-Carlo Methods

Temporal-Difference Learning

Policy Gradient Methods

Dynamic programming

Dynamic programming

Relates to solving **finite** MDP's

Iteratively performs **policy evaluation** and **policy improvement**

Classical DP methods operate in **sweeps** through state set: for $s \in S$ do ...

All values $p(s', r|s, a)$ should be known

Monte-Carlo methods

Monte-Carlo methods

Doesn't assume complete knowledge of the environment. It rather **simulates** experience

Learns optimal behaviour directly from interaction with environment with no model of environment dynamics

Can be used with simulation or **sample episodes**: sometimes it is much easier to simulate sample episodes even though it is difficult to calculate model's transition probabilities

Monte-Carlo methods

Monte-Carlo methods

Values $p(s', r|s, a)$ are not used explicitly, so they can be unknown

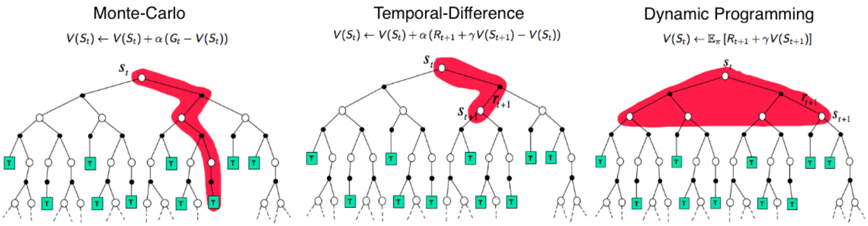
Opposed to DP methods MC methods don't bootstrap, e.g. they don't update their value estimates on the basis of other value estimates

Temporal-Difference Learning methods

Temporal-Difference Learning methods

Combination of DP and MC ideas: they learn from raw experience with environment (MC idea) and update estimates based in part on other learned estimates, without waiting for a final outcome (they bootstrap, DP idea)

Reviewed methods



Comparison of backup diagrams

Policy gradient

Policy gradient

Remember the goal? The goal is achieved by learning the best policy π possible.
Methods described above selected actions based on estimated action values

Policy gradient

Remember the goal? The goal is achieved by learning the best policy π possible.
Methods described above selected actions based on estimated action values
Can we learn policy without estimating values of actions first?

Parametrization of policy

Let $\theta \in \mathbb{R}^d$

Parametrization of policy

Let $\theta \in \mathbb{R}^d$

$$\pi(a|s,\theta) = Pr(A_t = a|S_t = s, \theta_t = \theta)$$

Parametrization of policy

For example

Parametrization of policy

For example

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

$$h(s,a,\theta) = \theta^T x(s,a)$$

$x(s,a) \in \mathbb{R}^d$ - feature vector

Parametrization of policy

Let $J(\theta)$ be a performance measure with respect to policy parameter

Parametrization of policy

Let $J(\theta)$ be a performance measure with respect to policy parameter

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

Gradient ascent update rule

Performance measure

Definition (Episodic performance)

$$J(\theta) = V_{\pi_\theta}(s_0) = E_{\pi_\theta}[\sum_{t=0}^{\infty} \gamma^t R_{t+1} | S_0 = s_0]$$

Performance measure

Definition (Episodic performance)

$$J(\theta) = V_{\pi_\theta}(s_0) = E_{\pi_\theta}[\sum_{t=0}^{\infty} \gamma^t R_{t+1} | S_0 = s_0]$$

Now it is time to calculate $\nabla_\theta J(\theta)$. From here let's assume, that $\gamma = 1$. Also index in π_θ will be omitted.

Policy gradient theorem

Theorem

$$\nabla_\theta J(\theta) \propto \sum_{s\in\mathcal{S}} d_\pi(s) \sum_{a\in\mathcal{A}} \nabla_\theta \pi(a|s) \cdot Q_\pi(s, a)$$

$$d_\pi(s) = \lim_{t\to\infty} Pr(S_t = s|s_0, \pi)$$

Policy gradient theorem

$$\nabla_\theta V_\pi(s) = \nabla_\theta \sum_{a \in \mathcal{A}} \pi(a|s) Q_\pi(s, a) =$$

Policy gradient theorem

$$\nabla_\theta V_\pi(s) = \nabla_\theta \sum_{a \in \mathcal{A}} \pi(a|s) Q_\pi(s, a) =$$

product rule

$$= \sum_{a \in \mathcal{A}} (\nabla_\theta \pi(a|s) Q_\pi(s, a) + \pi(a|s) \nabla_\theta Q_\pi(s, a)) =$$

Expand $Q_\pi(s, a)$ (Bellman equation)

Policy gradient theorem

$$= \sum_{a \in \mathcal{A}} (\nabla_\theta \pi(a|s) Q_\pi(s, a) +$$

$$\pi(a|s) \nabla_\theta \sum_{s', r} p(s', r|s, a)(r + V_\pi(s'))) =$$

$$\nabla_\theta p(s', r|s, a) r = 0$$

Policy gradient theorem

$$= \sum_{a \in \mathcal{A}} (\nabla_\theta \pi(a|s) Q_\pi(s,a) +$$

$$\pi(a|s) \sum_{s',r} p(s', r|s, a) \nabla_\theta V_\pi(s')) =$$

$$\sum_r p(s', r|s, a) \nabla_\theta V_\pi(s') =$$

$$\nabla_\theta V_\pi(s') \sum_r p(s', r|s, a) = p(s'|s, a) \nabla_\theta V_\pi(s')$$

Policy gradient theorem

$$= \sum_{a \in \mathcal{A}} (\nabla_\theta \pi(a|s) Q_\pi(s, a) +$$

$$\pi(a|s) \sum_{s'} p(s'|s, a) \nabla_\theta V_\pi(s'))$$

Policy gradient theorem

$$\nabla_\theta V_\pi(s) = \sum_{a\in\mathcal{A}} (\nabla_\theta \pi(a|s) Q_\pi(s,a) +$$

$$\pi(a|s) \sum_{s'} p(s'|s,a) \nabla_\theta V_\pi(s'))$$

Policy gradient theorem

Let $\phi(s) = \sum\limits_{a \in \mathcal{A}} \nabla_\theta \pi(a|s) Q_\pi(s, a)$. Then

$$\nabla_\theta V_\pi(s) = \phi(s) + \sum_a \pi(a|s) \sum_{s'} p(s'|s, a) \nabla_\theta V_\pi(s') =$$

Policy gradient theorem

Let $\phi(s) = \sum_{a \in \mathcal{A}} \nabla_\theta \pi(a|s) Q_\pi(s, a)$. Then

$$\nabla_\theta V_\pi(s) = \phi(s) + \sum_a \pi(a|s) \sum_{s'} p(s'|s, a) \nabla_\theta V_\pi(s') =$$

$$= \phi(s) + \sum_{s'} \sum_a \pi(a|s) p(s'|s, a) \nabla_\theta V_\pi(s') =$$

Policy gradient theorem

Let $\phi(s) = \sum_{a \in \mathcal{A}} \nabla_\theta \pi(a|s) Q_\pi(s, a)$. Then

$$\nabla_\theta V_\pi(s) = \phi(s) + \sum_a \pi(a|s) \sum_{s'} p(s'|s, a) \nabla_\theta V_\pi(s') =$$

$$= \phi(s) + \sum_{s'} \sum_a \pi(a|s) p(s'|s, a) \nabla_\theta V_\pi(s') =$$

$$= \phi(s) + \sum_{s'} p_\pi(s \to s', 1) \nabla_\theta V_\pi(s') =$$

Policy gradient theorem

$$= \phi(s) + \sum_{s'} p_\pi(s \to s', 1)[\phi(s') + \sum_{s''} p_\pi(s' \to s'', 1)\nabla_\theta V_\pi(s'')] =$$

Policy gradient theorem

$$= \phi(s) + \sum_{s'} p_\pi(s \to s', 1)[\phi(s') + \sum_{s''} p_\pi(s' \to s'', 1) \nabla_\theta V_\pi(s'')] =$$

$$= \phi(s) + \sum_{s'} p_\pi(s \to s', 1)\phi(s') +$$

$$+ \sum_{s'} p_\pi(s \to s', 1) \sum_{s''} p_\pi(s' \to s'', 1) \nabla_\theta V_\pi(s'') =$$

Policy gradient theorem

$$= \phi(s) + \sum_{s'} p_\pi(s \to s', 1)[\phi(s') + \sum_{s''} p_\pi(s' \to s'', 1)\nabla_\theta V_\pi(s'')] =$$

$$= \phi(s) + \sum_{s'} p_\pi(s \to s', 1)\phi(s') +$$

$$+ \sum_{s'} p_\pi(s \to s', 1) \sum_{s''} p_\pi(s' \to s'', 1)\nabla_\theta V_\pi(s'') =$$

$$= \ldots + \sum_{s''} \sum_{s'} p_\pi(s \to s', 1) p_\pi(s' \to s'', 1)\nabla_\theta V_\pi(s'') =$$

Policy gradient theorem

$$= \phi(s) + \sum_{s'} p_\pi(s \to s', 1)[\phi(s') + \sum_{s''} p_\pi(s' \to s'', 1)\nabla_\theta V_\pi(s'')] =$$

$$= \phi(s) + \sum_{s'} p_\pi(s \to s', 1)\phi(s') +$$

$$+ \sum_{s'} p_\pi(s \to s', 1) \sum_{s''} p_\pi(s' \to s'', 1)\nabla_\theta V_\pi(s'') =$$

$$= \ldots + \sum_{s''} \sum_{s'} p_\pi(s \to s', 1)p_\pi(s' \to s'', 1)\nabla_\theta V_\pi(s'') =$$

$$= \phi(s) + \sum_{s'} p_\pi(s \to s', 1)\phi(s') + \sum_{s''} p_\pi(s \to s'', 2)\nabla_\theta V_\pi(s'') =$$

Policy gradient theorem

$$= \phi(s) + \sum_{s'} p_\pi(s \to s', 1)\phi(s') + \sum_{s''} p_\pi(s \to s'', 2)\phi(s'') +$$

$$+ \sum_{s'''} p_\pi(s \to s''', 3)\nabla_\theta V_\pi(s''') =$$

Policy gradient theorem

$$= \phi(s) + \sum_{s'} p_\pi(s \to s', 1)\phi(s') + \sum_{s''} p_\pi(s \to s'', 2)\phi(s'') +$$

$$+ \sum_{s'''} p_\pi(s \to s''', 3) \nabla_\theta V_\pi(s''') =$$

$$= \ldots =$$

Policy gradient theorem

$$= \phi(s) + \sum_{s'} p_\pi(s \to s', 1)\phi(s') + \sum_{s''} p_\pi(s \to s'', 2)\phi(s'') +$$

$$+ \sum_{s'''} p_\pi(s \to s''', 3)\nabla_\theta V_\pi(s''') =$$

$$= \dots =$$

$$= \sum_{x \in \mathcal{S}} \sum_{k=0}^{\infty} p_\pi(s \to x, k)\phi(x)$$

Policy gradient theorem

$$\nabla_\theta J_\pi(\theta) = \nabla_\theta V_\pi(s_0) = \sum_s \sum_{k=0}^{\infty} p_\pi(s_0 \to s, k)\phi(s) =$$

Policy gradient theorem

$$\nabla_\theta J_\pi(\theta) = \nabla_\theta V_\pi(s_0) = \sum_s \sum_{k=0}^{\infty} p_\pi(s_0 \to s, k)\phi(s) =$$

$$= \sum_s \eta(s)\phi(s) = (\sum_s \eta(s)) \sum_s \frac{\eta(s)}{\sum_s \eta(s)}\phi(s) \propto$$

Policy gradient theorem

$$\nabla_\theta J_\pi(\theta) = \nabla_\theta V_\pi(s_0) = \sum_s \sum_{k=0}^{\infty} p_\pi(s_0 \to s, k)\phi(s) =$$

$$= \sum_s \eta(s)\phi(s) = (\sum_s \eta(s)) \sum_s \frac{\eta(s)}{\sum_s \eta(s)}\phi(s) \propto$$

$$\propto \sum_s \frac{\eta(s)}{\sum_s \eta(s)}\phi(s) = \sum_s d_\pi(s) \sum_a \nabla_\theta \pi(a|s) Q_\pi(s,a)$$

Policy gradient theorem

Lemma

Let X_s be a number of time steps spent in state s and event $A_k^s = \{\text{in time step k agent is in state s}\}$. Then

$$E_\pi[X_s] = \eta(s) = \sum_{k=0}^{\infty} p_\pi(s_0 \to s, k)$$

Policy gradient theorem

Lemma

Let X_s be a number of time steps spent in state s and event $A_k^s = \{$in time step k agent is in state s$\}$. Then

$$E_\pi[X_s] = \eta(s) = \sum_{k=0}^{\infty} p_\pi(s_0 \to s, k)$$

$$E[X_s] = E_\pi[\sum_{k=0}^{\infty} I\{A_k^s\}] = \sum_{k=0}^{\infty} E_\pi[I\{A_k^s\}] = \sum_{k=0}^{\infty} p_\pi(s_0 \to s, k)$$

Policy gradient theorem

Lemma

Let X_s be a number of time steps spent in state s and event $A_k^s = \{$in time step k agent is in state s$\}$. Then

$$E_\pi[X_s] = \eta(s) = \sum_{k=0}^{\infty} p_\pi(s_0 \to s, k)$$

$$E[X_s] = E_\pi[\sum_{k=0}^{\infty} I\{A_k^s\}] = \sum_{k=0}^{\infty} E_\pi[I\{A_k^s\}] = \sum_{k=0}^{\infty} p_\pi(s_0 \to s, k)$$

Then it becomes clear why $\frac{\eta(s)}{\sum\limits_s \eta(s)} = d_\pi(s) = \lim\limits_{t\to\infty} Pr(S_t = s | s_0, \pi)$

PGT (Continuing case)

Definition (Average value)

$$J_{avV}(\theta) = \sum_s d_\pi(s) V_\pi(s)$$

Definition (Average reward per time-step)

$$J_{avR}(\theta) = \sum_s d_\pi(s) \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a) r$$

PGT (Continuing case)

Theorem

Gradient of function $J = \frac{1}{1-\gamma} J_{avV}(\theta), J_{avR}(\theta)$ *is*

$$\nabla_\theta J = \sum_{s \in \mathcal{S}} d_\pi(s) \sum_{s \in \mathcal{A}} \nabla_\theta \pi(a|s) \cdot Q_\pi(s, a)$$

Log-derivative trick

$$\nabla_\theta J_\pi(\theta) \propto \sum_s d_\pi(s) \sum_a \nabla_\theta \pi(a|s) Q_\pi(s,a) =$$

Log-derivative trick

$$\nabla_\theta J_\pi(\theta) \propto \sum_s d_\pi(s) \sum_a \nabla_\theta \pi(a|s) Q_\pi(s,a) =$$

$$= E_\pi[\sum_a Q_\pi(S,a) \nabla_\theta \pi(a|S)] =$$

Log-derivative trick

$$\nabla_\theta J_\pi(\theta) \propto \sum_s d_\pi(s) \sum_a \nabla_\theta \pi(a|s) Q_\pi(s,a) =$$

$$= E_\pi[\sum_a Q_\pi(S,a) \nabla_\theta \pi(a|S)] =$$

$$= E_\pi[\sum_a \pi(a|S) Q_\pi(S,a) \frac{\nabla_\theta \pi(a|S)}{\pi(a|S)}] =$$

Log-derivative trick

$$\nabla_\theta J_\pi(\theta) \propto \sum_s d_\pi(s) \sum_a \nabla_\theta \pi(a|s) Q_\pi(s,a) =$$

$$= E_\pi[\sum_a Q_\pi(S,a) \nabla_\theta \pi(a|S)] =$$

$$= E_\pi[\sum_a \pi(a|S) Q_\pi(S,a) \frac{\nabla_\theta \pi(a|S)}{\pi(a|S)}] =$$

$$= E_\pi[Q_\pi(S,A) \frac{\nabla_\theta \pi(A|S)}{\pi(A|S)}] = E_\pi[Q_\pi(S,A) \nabla_\theta \ln \pi(A|S)]$$

REINFORCE

$$E_\pi[Q_\pi(S_t, A_t)\nabla_\theta \ln \pi(A_t|S_t)] = E_\pi[G_t \nabla_\theta \ln \pi(A_t|S_t)]$$

since

$$E_\pi[G_t|S_t = s, A_t = a] = Q_\pi(s, a)$$

$$E_\pi[G_t|S_t, A_t] = Q_\pi(S_t, A_t)$$

REINFORCE

$$E_\pi[Q_\pi(S_t, A_t)\nabla_\theta \ln \pi(A_t|S_t)] = E_\pi[G_t\nabla_\theta \ln \pi(A_t|S_t)]$$

since

$$E_\pi[G_t|S_t = s, A_t = a] = Q_\pi(s, a)$$

$$E_\pi[G_t|S_t, A_t] = Q_\pi(S_t, A_t)$$

$$E_\pi[Q_\pi(S_t, A_t)\nabla_\theta \ln \pi(A_t|S_t)] =$$

$$= E_\pi[E_\pi[G_t|S_t, A_t]\nabla_\theta \ln \pi(A_t|S_t)] =$$

$$= E_\pi[E_\pi[G_t\nabla_\theta \ln \pi(A_t|S_t)|S_t, A_t]] = E_\pi[G_t\nabla_\theta \ln \pi(A_t|S_t)]$$

REINFORCE

$$\nabla_\theta J(\theta) = E_\pi[G_t \nabla_\theta \ln \pi(A_t|S_t)]$$

REINFORCE

$$\nabla_\theta J(\theta) = E_\pi[G_t \nabla_\theta \ln \pi(A_t|S_t)]$$

$$\theta_{t+1} = \theta_t + \alpha \widehat{\nabla_\theta J(\theta)}$$

Update rule of REINFORCE algorithm (after Williams, 1992). Reinforce is a Monte-Carlo algorithm, because it uses G_t, which includes all rewards up until the end of episode.
REINFORCE = "REward Increment = Nonnegative Factor × Offset Reinforcement × Characteristic Eligibility"

REINFORCE

$$\nabla_\theta J_{st}(\theta) = E_\pi[G_t \nabla_\theta \ln \pi(A_t|S_t)]$$

REINFORCE

$$\nabla_\theta J_{st}(\theta) = E_\pi[G_t \nabla_\theta \ln \pi(A_t|S_t)]$$

Another way to view this ([6, 7, 8]) is

$$\nabla_\theta J_{tr}(\theta) = \nabla_\theta E_{\tau\sim\pi}[R(\tau)] = E_{\tau\sim\pi}[R(\tau)\nabla_\theta \log P(\tau)] =$$

$$= E_{\tau\sim\pi}[R(\tau)\left(\sum_{t=1}^{T} \nabla_\theta \log \pi(a_t|s_t)\right)]$$

where $\tau = (s_0, a_0, r_1, \ldots, s_{T-1}, a_{T-1}, r_T)$.

REINFORCE

$$\nabla_\theta J_{st}(\theta) = E_\pi[G_t \nabla_\theta \ln \pi(A_t|S_t)]$$

$$\nabla_\theta J_{tr}(\theta) = E_{\tau\sim\pi}[R(\tau)\left(\sum_{t=1}^{T} \nabla_\theta \log \pi(a_t|s_t)\right)]$$

Key observation is that $\nabla_\theta J_{tr}(\theta) = \sum_{t=1}^{T} \nabla_\theta J_{st}(\theta) =$ $|\text{each summand is expected value of same variable}| =$ $T \cdot \nabla_\theta J_{st}(\theta)$. Remember the $\sum_s \eta(s) \approx T$ constant we omitted (only in episodic case).

REINFORCE

Algorithm

Set hyperparameters: N - amount of trajectories, parametrised policy $\pi(a|s,\theta)$
Arbitrary initialisation of θ
while True:

1. Run N trajectories $\mathcal{T}_1, \dots, \mathcal{T}_N \sim \pi$
2. For each t in each $\mathcal{T}$ calculate $G_t(\mathcal{T}) = \sum_{k=t}^{T} \gamma^{k-t} r_k$
3. $$\widehat{\nabla_\theta J(\theta)} = \frac{1}{N} \sum_{\mathcal{T}} \sum_{t=0}^{T_i} \gamma^t \nabla_\theta \log \pi(a_t|s_t) G_t(\mathcal{T})$$
4. $\theta = \theta + \alpha \widehat{\nabla_\theta J(\theta)}$.

REINFORCE

$\widehat{\nabla_\theta J(\theta)}$ from the previous slide is an unbiased estimate of both $\nabla_\theta J_{st}(\theta)$ and $\nabla_\theta J_{tr}(\theta)$ up to some constant, which is regulated by learning rate α

Baseline

$$\nabla_\theta J_\pi(\theta) \propto \sum_s d_\pi(s) \sum_a \nabla_\theta \pi(a|s) Q_\pi(s, a)$$

Baseline

$$\nabla_\theta J_\pi(\theta) \propto \sum_s d_\pi(s) \sum_a \nabla_\theta \pi(a|s) Q_\pi(s,a)$$

Let's pick an arbitrary baseline $b(s)$ which doesn't depend on a and consider

$$\sum_s d_\pi(s) \sum_a \nabla_\theta \pi(a|s)(Q_\pi(s,a) - b(s))$$

Baseline

$$\nabla_\theta J_\pi(\theta) \propto \sum_s d_\pi(s) \sum_a \nabla_\theta \pi(a|s) Q_\pi(s,a)$$

Let's pick an arbitrary baseline $b(s)$ which doesn't depend on a and consider

$$\sum_s d_\pi(s) \sum_a \nabla_\theta \pi(a|s)(Q_\pi(s,a) - b(s))$$

The value doesn't change, because

$$\sum_a b(s) \nabla_\theta \pi(a|s) = b(s) \sum_a \nabla_\theta \pi(a|s) =$$

Baseline

$$\nabla_\theta J_\pi(\theta) \propto \sum_s d_\pi(s) \sum_a \nabla_\theta \pi(a|s) Q_\pi(s,a)$$

Let's pick an arbitrary baseline $b(s)$ which doesn't depend on a and consider

$$\sum_s d_\pi(s) \sum_a \nabla_\theta \pi(a|s)(Q_\pi(s,a) - b(s))$$

The value doesn't change, because

$$\sum_a b(s) \nabla_\theta \pi(a|s) = b(s) \sum_a \nabla_\theta \pi(a|s) =$$

$$= b(s) \nabla_\theta \sum_a \pi(a|s) = b(s) \nabla_\theta 1 = 0$$

REINFORCE with Baseline

$$\nabla_\theta J_\pi(\theta) \propto \sum_s d_\pi(s) \sum_a \nabla_\theta \pi(a|s)(Q_\pi(s,a) - b(s))$$

$$\widehat{\nabla_\theta J(\theta)} = \frac{1}{N} \sum_{\mathcal{T}} \sum_{t=0}^{T_i} \gamma^t \nabla_\theta \log \pi(a_t|s_t)(G_t(\mathcal{T}) - b(s_t))$$

Usually $b(s) = V_\pi\widehat{(S_t, w)}$, which is also a target function to learn.

Baseline

$$\nabla_\theta J_\pi(\theta) = E_\pi[(G_t - b(S_t))\nabla_\theta \ln \pi(A_t|S_t)]$$

Baseline

$$\nabla_\theta J_\pi(\theta) = E_\pi[(G_t - b(S_t))\nabla_\theta \ln \pi(A_t|S_t)]$$

$$Var[X] = E[X^2] - (E[X])^2$$

Remember, baseline doesn't add bias, so $E[X]$ is the same as before. Thus the difference is only in term $E[X^2]$

Baseline

$$\nabla_\theta J_\pi(\theta) = E_\pi[(G_t - b(S_t))\nabla_\theta \ln \pi(A_t|S_t)]$$

$$Var[X] = E[X^2] - (E[X])^2$$

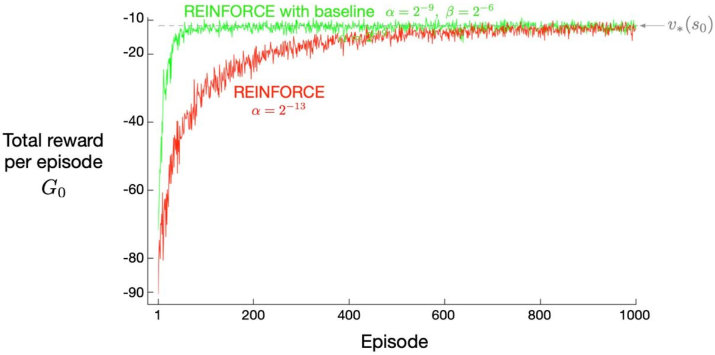
Remember, baseline doesn't add bias, so $E[X]$ is the same as before. Thus the difference is only in term $E[X^2]$
Lets approximate as

$$E_\pi[((G_t - b(S_t))\nabla_\theta \ln \pi(A_t|S_t))^2] =$$

$$= E_\pi[((G_t - b(S_t))^2]E_\pi[(\nabla_\theta \ln \pi(A_t|S_t))^2]$$

Baseline

By optimising baseline we can minimize term $E_\pi[(G_t - b(S_t))^2]$ and get lower variance, thus faster convergence



Actor-Critic

Actor-critic methods consist of two models, which may optionally share parameters:

Actor-Critic

Actor-critic methods consist of two models, which may optionally share parameters:
Critic updates the value function parameters w and depending on the algorithm it could be action-value $Q_w(s, a)$ or state-value $V_w(s)$.
Actor updates the policy parameters θ for $\pi_\theta(a|s)$, in the direction suggested by the critic.

Actor-Critic

One-step Actor-Critic method update rule (for $V_w(s)$):

Actor-Critic

One-step Actor-Critic method update rule (for $V_w(s)$):

$$\theta_{t+1} = \theta_t + \alpha(G_{t:t+1} - \hat{V}(S_t, w))\nabla_\theta \ln \pi(A_t|S_t) =$$

Actor-Critic

One-step Actor-Critic method update rule (for $V_w(s)$):

$$\theta_{t+1} = \theta_t + \alpha(G_{t:t+1} - \hat{V}(S_t, w))\nabla_\theta \ln \pi(A_t|S_t) =$$

$$= \theta_t + \alpha(R_{t+1} + \gamma \hat{V}(S_{t+1}, w) - \hat{V}(S_t, w))\nabla_\theta \ln \pi(A_t|S_t) =$$

$$= \theta_t + \alpha \delta_t \nabla_\theta \ln \pi(A_t|S_t)$$

Actor-Critic

One-step Actor–Critic (episodic)

Input: a differentiable policy parameterization $\pi(a|s,\boldsymbol{\theta})$
Input: a differentiable state-value 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 $\mathbf{w} \in \mathbb{R}^{d}$
Repeat forever:
  Initialize S (first state of episode)
  $I \leftarrow 1$
  While S is not terminal:
    $A \sim \pi(\cdot|S,\boldsymbol{\theta})$
    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 $\hat{v}(S',\mathbf{w}) \doteq 0$)
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}} I \delta \nabla_{\mathbf{w}}\hat{v}(S,\mathbf{w})$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} I \delta \nabla_{\boldsymbol{\theta}} \ln \pi(A|S,\boldsymbol{\theta})$
    $I \leftarrow \gamma I$
    $S \leftarrow S'$

Conclusion

I **Policy gradient methods** learn parametrized policy rather than estimates value functions

II **Policy gradient theorem** gives an analytic expression of gradient of performance measure. This expression does not involve derivatives of the state distribution

III In probabilistic algorithms efficiency can be increased by minimizing variance. The common option is to introduce a **baseline**.

IV **Actor-Critic** method uses both Monte-Carlo and bootstrapping ideas and acts like temporal-difference methods, while being a policy gradient method.

Conclusion. Policy Gradient approach

Advantages

I. Finds the best Stochastic Policy
II. Naturally explores due to Stochastic Policy representation
III. Effective in high-dimensional or continuous action spaces

Disadvantages

I. Typically converge to a local optimum rather than a global optimum
II. Policy Evaluation is typically inefficient and has high variance
III. Policy Improvement happens in small steps $\implies$ slow convergence

References

[1] Richard S. Sutton and Andrew G. Barto.
Reinforcement Learning: An Introduction; 2nd Edition.
2018.
[2] A (Long) Peek into Reinforcement Learning by Lilian Weng. https://lilianweng.github.io/lil-log/2018/02/19/a-long-peek-into-reinforcement-learning.html
[3] Policy Gradient Algorithms by Lilian Weng.
https://lilianweng.github.io/lil-log/2018/04/08/policy-gradient-algorithms.html

References

[4] Policy Gradient Algorithms by Ashwin Rao, ICME, Stanford University https://stanford.edu/~ashlearn/RLForFinanceBook/PolicyGradient.pdf
[5] OpenAI Gym https://gym.openai.com/
[6] Методы policy gradient и алгоритм асинхронного актора-критика http://neerc.ifmo.ru/wiki
[7] Policy Gradients. CS 294-112: Deep Reinforcement Learning by Sergey Levine
https://rll.berkeley.edu/deeprlcourse/f17docs/lecture_4_policy_gradient.pdf

References

[8] Обучение с подкреплением (курс лекций) / 2020
http://www.machinelearning.ru/wiki/index.php?title=Rl