REINFORCE, Baselines and Actor-Critic

Roberto Capobianco

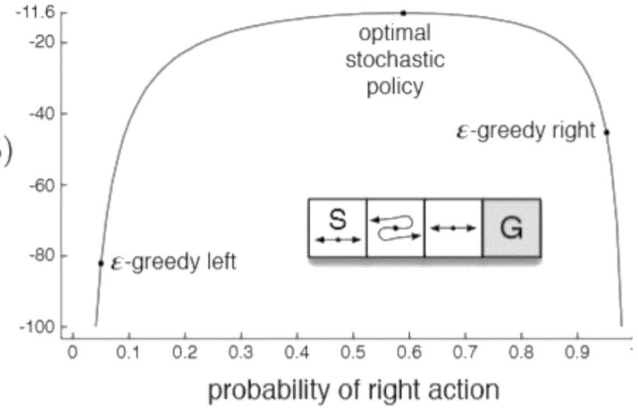
SAPIENZA
UNIVERSITÀ DI ROMA

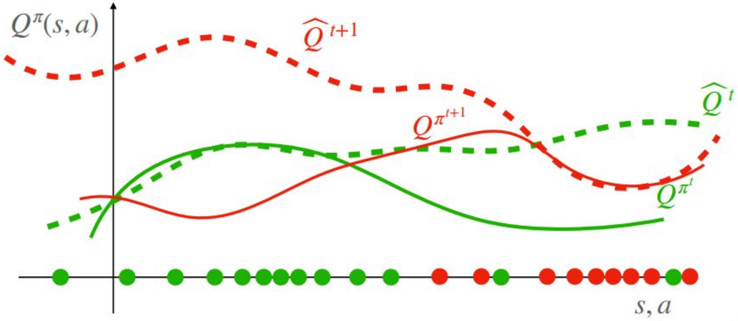
Recap

Policy from Value: Problems

So far, we derived a policy out of a tabular or parameterized state-action value function

$J(\boldsymbol{\theta}) = v_{\pi_{\boldsymbol{\theta}}}(\mathrm{S})$





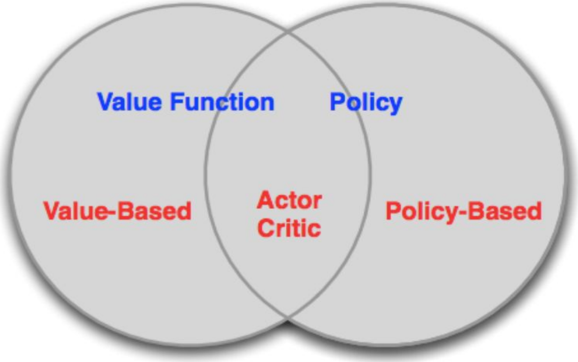
Green dots: (s,a) from π^t

Red dots: (s,a) from π^{t+1}

Parameterized Policy

Can we directly represent and learn a parameterized policy?

$$a \sim \pi_{\theta}(s) \text{ or } p(.|s,\theta)$$



Credits: David Silver

Finding a Parameterized Policy

How do we search for a parameterized policy π_θ(s)?

We want to find the parameters θ such that π_θ is the best policy

Quality of a Parameterized Policy

How do we measure the quality for a policy?

In other words, what is our objective function to maximize through the policy parameters?

- For episodic tasks:
  - Value at start state (undiscounted)
- For continuing tasks:
  - Value at start state (discounted)
  - Average value

    see [Sutton&Barto 10.4: Deprecating the Discounted Setting]

  - Average reward per timestep

    see [Sutton&Barto 10.4: Deprecating the Discounted Setting]

$$J(\pi) := \mathbb{E}\left[\sum_{h=0}^{H-1} r(s_h, a_h) \,\middle|\, s_0 \sim \mu, s_{h+1} \sim P_{s_h,a_h}, a_h \sim \pi(\cdot \mid s_h)\right]$$

$$J(\pi) := \mathbb{E}\left[\sum_{h=0}^{\infty} \gamma^h r(s_h, a_h) \,\middle|\, s_0 \sim \mu, s_{h+1} \sim P_{s_h,a_h}, a_h \sim \pi(\cdot \mid s_h)\right]$$

Policy Optimization

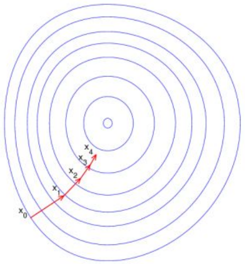
Can use gradient free optimization

- Hill climbing Simplex / Nelder Mead Genetic algorithms
- Cross-Entropy method (CEM)
- Covariance Matrix Adaptation (CMA)

Evolution strategies can rival the performance of standard RL techniques on modern RL benchmarks (e.g. Atari/MuJoCo): https://openai.com/blog/evolution-strategies/

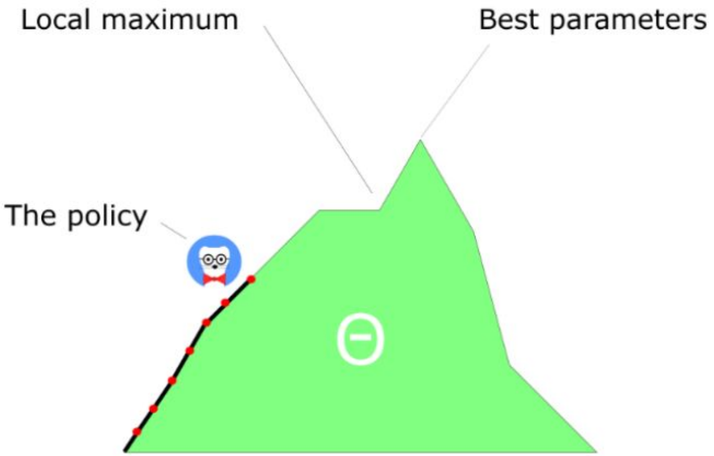
Can work with any policy parameterizations, including non-differentiable ones, but can be sample inefficient and requires higher outcome variability

Policy Gradient



An efficient and widely used technique to find a parameterized policy is the policy gradient

Guaranteed to converge to local maximum or global maximum, *but it often converges only to a local maximum*



Policy Gradient

$$\pi_\theta(a \mid s) = \pi(a \mid s; \theta) \qquad J(\pi_\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{h=0}^{\infty} \gamma^h r_h\right]$$

$$\theta_{t+1} = \theta_t + \eta \boxed{\nabla_\theta J(\pi_\theta)\,|_{\theta=\theta_t}}$$

HOW?

Finite Differences

$$\pi_\theta(a \mid s) = \pi(a \mid s; \theta) \qquad J(\pi_\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{h=0}^{\infty} \gamma^h r_h\right]$$

$$\theta_{t+1} = \theta_t + \eta \nabla_\theta J(\pi_\theta)\,|_{\theta=\theta_t}$$

Simple approach: compute $\nabla_\theta J(\pi_\theta)$ using finite differences

IDEA: perturb θ by small amount in k-th dimension

$$\frac{\partial V(s_0,\theta)}{\partial \theta_k} \approx \frac{V(s_0, \theta + \epsilon \boxed{u_k}) - V(s_0,\theta)}{\epsilon} \qquad \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \begin{array}{l} \\ \\ \\ \\ \leftarrow k\text{-th entry} \\ \\ \\ \\ \end{array}$$

Uses n evaluations to compute policy gradient in n dimensions

Works for arbitrary policies, even if policy is not differentiable. Noise can dominate, but can be reduced by averaging over many samples or by reducing randomness if possible

Policy Gradient

$$\pi_\theta(a \mid s) = \pi(a \mid s; \theta) \qquad J(\pi_\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{h=0}^{\infty} \gamma^h r_h\right]$$

$$\theta_{t+1} = \theta_t + \eta \nabla_\theta J(\pi_\theta)\,|_{\theta=\theta_t}$$

Can we compute the gradient analytically?

Differentiable Policy Classes

Softmax Linear Policy

$$\pi_\theta(a \mid s) = \frac{\exp(\theta^\top \phi(s,a))}{\sum_{a'} \exp(\theta^\top \phi(s,a'))}$$

Softmax Policy

$$\pi_\theta(a \mid s) = \frac{\exp(f_\theta(s,a))}{\sum_{a'} \exp(f_\theta(s,a'))}$$

Gaussian Policy

$$\pi_\theta(a \mid s) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(a - f_\theta(s))^2}{2\sigma^2}\right)$$

f can be a neural network

Likelihood Ratio Policy Gradient

Suppose we have a trajectory $\tau = \{s_0, a_0, s_1, a_1, \dots\}$

it's probability distribution, depending on θ is

$$\rho_\theta(\tau) = \mu(s_0)\pi_\theta(a_0 \mid s_0)P(s_1 \mid s_0, a_0)\pi_\theta(a_1 \mid s_1)\dots$$

$$J(\pi_\theta) = \mathbb{E}_{\tau \sim \rho_\theta(\tau)} \underbrace{\left[\sum_{h=0}^{\infty} \gamma^h r(s_h, a_h) \right]}_{R(\tau)} = \sum_\tau \boxed{P(\tau;\theta)} R(\tau)$$

probability of
each trajectory
from the
distribution

SAPIENZA
UNIVERSITÀ DI ROMA

Likelihood Ratio Policy Gradient

$$\nabla_\theta J(\theta) \approx \hat{g} = (1/m) \sum_{i=1}^{m} R(\tau^{(i)}) \nabla_\theta \log P(\tau^{(i)}; \theta)$$

$$= (1/m) \sum_{i=1}^{m} \boxed{R(\tau^{(i)})} \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t^{(i)} | s_t^{(i)})$$

Measure how good the sample trajectory is

Policy Gradient Theorem (Finite Setting)

The policy gradient theorem generalizes the likelihood ratio approach

$$\nabla_\theta J(\pi_\theta) = \sum_{h=0}^{H-1} \mathbb{E}_{s_h, a_h \sim \mathbb{P}_h^{\pi_\theta}} \left[\nabla \ln \pi_\theta(a_h \mid s_h) \cdot Q_h^{\pi_\theta}(s_h, a_h) \right]$$

End Recap

Policy Gradient Theorem (Infinite Setting)

The policy gradient theorem generalizes the likelihood ratio approach

$$\nabla_\theta J(\pi_\theta) = \sum_{h=0}^{\infty} \gamma^h \mathbb{E}_{s,a \sim \mathbb{P}_h^{\pi_\theta}} \nabla_\theta \ln \pi_\theta(a \mid s) \cdot Q^{\pi_\theta}(s,a)$$

$$= \frac{1}{1-\gamma} \mathbb{E}_{s,a \sim d_\mu^{\pi_\theta}} \left[\nabla_\theta \ln \pi_\theta(a \mid s) \cdot \boxed{Q^{\pi_\theta}(s,a)} \right]$$

HOW?

Policy Gradient Theorem (Infinite Setting)

The policy gradient theorem generalizes the likelihood ratio approach

$$\nabla_\theta J(\pi_\theta) = \sum_{h=0}^{\infty} \gamma^h \mathbb{E}_{s,a \sim \mathbb{P}_h^{\pi_\theta}} \nabla_\theta \ln \pi_\theta(a \mid s) \cdot Q^{\pi_\theta}(s,a)$$

$$= \frac{1}{1-\gamma} \mathbb{E}_{s,a \sim d_\mu^{\pi_\theta}} \left[\nabla_\theta \ln \pi_\theta(a \mid s) \cdot \boxed{Q^{\pi_\theta}(s,a)} \right]$$

This is a familiar problem!

Policy Gradient Theorem (Infinite Setting)

The policy gradient theorem generalizes the likelihood ratio approach

$$\nabla_\theta J(\pi_\theta) = \sum_{h=0}^{\infty} \gamma^h \mathbb{E}_{s,a\sim\mathbb{P}_h^{\pi_\theta}} \nabla_\theta \ln \pi_\theta(a \mid s) \cdot Q^{\pi_\theta}(s,a)$$

$$= \frac{1}{1-\gamma} \mathbb{E}_{s,a\sim d_\mu^{\pi_\theta}} \left[\nabla_\theta \ln \pi_\theta(a \mid s) \cdot \boxed{Q^{\pi_\theta}(s,a)]} \right.$$

Policy Evaluation!

Policy Gradient Theorem (Infinite Setting)

The policy gradient theorem generalizes the likelihood ratio approach

$$
\begin{aligned}
\nabla_\theta J(\pi_\theta) &= \sum_{h=0}^{\infty} \gamma^h \mathbb{E}_{s,a\sim\mathbb{P}_h^{\pi_\theta}} \nabla_\theta \ln \pi_\theta(a \mid s) \cdot Q^{\pi_\theta}(s,a) \\
&= \frac{1}{1-\gamma} \mathbb{E}_{s,a\sim d_\mu^{\pi_\theta}} \left[\nabla_\theta \ln \pi_\theta(a \mid s) \cdot \boxed{Q^{\pi_\theta}(s,a)} \right]
\end{aligned}
$$

We can use the return G as an unbiased estimate of Q (MC)

REINFORCE

Initialize policy parameters θ arbitrarily
for each episode $\{s_1, a_1, r_2, \cdots, s_{T-1}, a_{T-1}, r_T\} \sim \pi_\theta$ **do**
  for $t = 1$ to $T - 1$ **do**
    $\theta \leftarrow \theta + \alpha \nabla_\theta \log \pi_\theta(s_t, a_t) G_t$
  endfor
endfor
return θ

REINFORCE

Initialize policy parameters θ arbitrarily

for each episode $\{s_1, a_1, r_2, \cdots, s_{T-1}, a_{T-1}, r_T\} \sim \pi_\theta$ **do**

  for $t = 1$ to $T - 1$ **do**

    $\theta \leftarrow \theta + \alpha \nabla_\theta \log \pi_\theta(s_t, a_t) G_t$

  endfor

endfor

return θ

Unbiased, but remember the issue?

REINFORCE

Initialize policy parameters θ arbitrarily

for each episode $\{s_1, a_1, r_2, \cdots, s_{T-1}, a_{T-1}, r_T\} \sim \pi_\theta$ **do**

  for $t = 1$ to $T - 1$ **do**

    $\theta \leftarrow \theta + \alpha \nabla_\theta \log \pi_\theta(s_t, a_t) G_t$

  endfor

endfor

return θ

VARIANCE!

Baseline

To reduce the variance we can introduce baselines (function of state)

Baseline

To reduce the variance we can introduce baselines (function of state)

$$\nabla_\theta J(\pi_\theta) = \frac{1}{1-\gamma}\mathbb{E}_{s,a\sim d_\mu^{\pi_\theta}}\left[\nabla_\theta \ln \pi_\theta(a\,|\,s)\cdot\left(Q^{\pi_\theta}(s,a)-\boxed{b(s)}\right)\right]$$

Baseline

To reduce the variance we can introduce baselines (function of state)

$$\nabla_\theta J(\pi_\theta) = \frac{1}{1-\gamma} \boxed{\mathbb{E}_{s,a\sim d_\mu^{\pi_\theta}} \Big[\nabla_\theta \ln \pi_\theta(a\,|\,s) \cdot} \Big(Q^{\pi_\theta}(s,a) - \boxed{b(s)}\Big) \Big]$$

Is this term introducing a bias?

Baseline

Baselines do not introduce bias

$$\mathbb{E}_{a\sim\pi_\theta(\cdot|s)}\boxed{\nabla_\theta \ln \pi_\theta(a\,|\,s)}\,b(s)$$

Let's expand this term

Baseline

Baselines do not introduce bias

$$\mathbb{E}_{a\sim\pi_\theta(\cdot|s)} \nabla_\theta \ln \pi_\theta(a|s) b(s)$$

$$= \sum_a \pi_\theta(a|s) \boxed{\frac{\nabla \pi_\theta(a|s)}{\pi_\theta(a|s)} b(s)}$$

And fully write down the gradient of ln

Baseline

Baselines do not introduce bias

$$\mathbb{E}_{a\sim\pi_\theta(\cdot|s)} \nabla_\theta \ln \pi_\theta(a\,|\,s) b(s)$$

$$= \sum_a \pi_\theta(a\,|\,s) \frac{\nabla \pi_\theta(a\,|\,s)}{\pi_\theta(a\,|\,s)} b(s)$$

And fully write down the gradient of ln as well as the expectation

Baseline

Baselines do not introduce bias

$$\mathbb{E}_{a\sim\pi_\theta(\cdot|s)} \nabla_\theta \ln \pi_\theta(a\,|\,s) b(s)$$

$$= \sum_a \pi_\theta(a\,|\,s) \frac{\nabla \pi_\theta(a\,|\,s)}{\pi_\theta(a\,|\,s)} b(s) \quad = b(s) \sum_a \nabla \pi_\theta(a\,|\,s)$$

We can get simplify and highlight b,
as it does not depend on a (the sum)

Baseline

Baselines do not introduce bias

$$\mathbb{E}_{a\sim\pi_\theta(\cdot|s)} \nabla_\theta \ln \pi_\theta(a\,|\,s) b(s)$$

$$= \sum_a \pi_\theta(a\,|\,s) \frac{\nabla \pi_\theta(a\,|\,s)}{\pi_\theta(a\,|\,s)} b(s) \;= b(s) \sum_a \nabla \pi_\theta(a\,|\,s) = b(s) \nabla \left[\sum_a \pi_\theta(a\,|\,s) \right] \;= \boxed{b(s) \nabla 1 = 0}$$

This holds for any baseline as long as
it is action-independent

Value Function as Baseline

As baselines have to be action-independent, a common choice for a baseline is

$$b(s) = V^{\pi_\theta}(s)$$

Value Function as Baseline

As baselines have to be action-independent, a common choice for a baseline is

$$b(s) = V^{\pi_\theta}(s)$$

$$\nabla_\theta J(\theta) = \frac{1}{1-\gamma}\mathbb{E}_{s,a\sim d^{\pi_\theta}}\left[\nabla_\theta \ln \pi_\theta(a \mid s)\big(Q^{\pi_\theta}(s,a) - V^{\pi_\theta}(s)\big)\right]$$

Value Function as Baseline

As baselines have to be action-independent, a common choice for a baseline is

$$b(s) = V^{\pi_\theta}(s)$$

$$\nabla_\theta J(\theta) = \frac{1}{1-\gamma}\mathbb{E}_{s,a\sim d^{\pi_\theta}}\left[\nabla_\theta \ln \pi_\theta(a\,|\,s)\boxed{\left(Q^{\pi_\theta}(s,a) - V^{\pi_\theta}(s)\right)}\right]$$

Called Advantage Function

$$\nabla_\theta J(\theta_t) = \frac{1}{1-\gamma}\mathbb{E}_{s,a\sim d^{\pi_{\theta_t}}}\left[\nabla_\theta \ln \pi_{\theta_t}(a\,|\,s)\boxed{\left(A^{\pi_{\theta_t}}(s,a)\right)}\right]$$

Advantage Function

Intuition: the advantage function tells us how good an action is compared to the average value of the state

Value of an action in the state

$$Q^{\pi_\theta}(s, a) - V^{\pi_\theta}(s)$$

Average value of the state

Advantage Function

Intuition: the advantage function tells us how good an action is compared to the average value of the state

$$Q^{\pi_\theta}(s, a) - V^{\pi_\theta}(s)$$

Remember in REINFORCE we estimate our Q by the return

REINFORCE with Baseline

Initialize policy parameter θ, baseline b

for iteration$=1, 2, \cdots$ **do**

Collect a set of trajectories by executing the current policy

At each timestep t in each trajectory τ^i, compute

Return $G_t^i = \sum_{t'=t}^{T-1} r_{t'}^i$, and

Advantage estimate $\hat{A}_t^i = G_t^i - b(s_t)$.

Re-fit the baseline, by minimizing $\sum_i \sum_t ||b(s_t) - G_t^i||^2$,

Update the policy, using a policy gradient estimate $\hat{g}$,

Which is a sum of terms $\nabla_\theta \log \pi(a_t|s_t, \theta)\hat{A}_t$.

(Plug $\hat{g}$ into SGD or ADAM)

endfor

Credits: Emma Brunskill

REINFORCE with Baseline

Initialize policy parameter θ, baseline b

for iteration$=1, 2, \cdots$ **do**

Collect a set of trajectories by executing the current policy

At each timestep t in each trajectory τ^i, compute

Return $\boxed{G_t^i = \sum_{t'=t}^{T-1} r_{t'}^i}$, and

Advantage estimate $\hat{A}_t^i = G_t^i - b(s_t)$.

Re-fit the baseline, by minimizing $\sum_i \sum_t ||b(s_t) - G_t^i||^2$,

Update the policy, using a policy gradient estimate $\hat{g}$,

Which is a sum of terms $\nabla_\theta \log \pi(a_t|s_t, \theta)\hat{A}_t$.

(Plug $\hat{g}$ into SGD or ADAM)

endfor

We're still using the return and collecting MC samples

Credits: Emma Brunskill

SAPIENZA UNIVERSITÀ DI ROMA

Advantage Function

$$Q^{\pi_\theta}(s,a) - V^{\pi_\theta}(s)$$

If we can access the true value function, the TD error is an unbiased estimate of the advantage function

$$\boxed{\delta^{\pi_\theta} = r + \gamma V^{\pi_\theta}(s') - V^{\pi_\theta}(s)}$$

$$\begin{aligned}
\mathbb{E}_{\pi_\theta}\left[\delta^{\pi_\theta} | s, a\right] &= \mathbb{E}_{\pi_\theta}\left[r + \gamma V^{\pi_\theta}(s') | s, a\right] - V^{\pi_\theta}(s) \\
&= Q^{\pi_\theta}(s,a) - V^{\pi_\theta}(s) \\
&= A^{\pi_\theta}(s,a)
\end{aligned}$$

Credits: David Silver

Advantage Function

$$Q^{\pi_\theta}(s,a) - V^{\pi_\theta}(s)$$

So we can use the TD error to compute the policy gradient

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s,a)\ \delta^{\pi_\theta}\right]$$

$$\delta^{\pi_\theta} = r + \gamma V^{\pi_\theta}(s') - V^{\pi_\theta}(s)$$

Credits: David Silver

SAPIENZA UNIVERSITÀ DI ROMA

Advantage Function

$$Q^{\pi_\theta}(s, a) - V^{\pi_\theta}(s)$$

So we can use the TD error to compute the policy gradient

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s, a)\ \delta^{\pi_\theta}\right]$$

$$\delta^{\pi_\theta} = r + \gamma \boxed{V^{\pi_\theta}(s')} - \boxed{V^{\pi_\theta}(s)}$$

Can be approximated!

Advantage Function

$$Q^{\pi_\theta}(s, a) - V^{\pi_\theta}(s)$$

So we can use the TD error to compute the policy gradient

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s, a)\ \delta^{\pi_\theta}\right]$$

$$\delta^{\pi_\theta} = r + \gamma \boxed{V^{\pi_\theta}(s')} - \boxed{V^{\pi_\theta}(s)}$$

Introduces bias, but it's fine as we know from TD

Advantage Function

$$Q^{\pi_\theta}(s,a) - V^{\pi_\theta}(s)$$

So we can use the TD error to compute the policy gradient

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s,a)\, \boxed{\delta^{\pi_\theta}}\right]$$

This leads us to the notion of critic

Reducing Variance with Critic

Motivation: Monte-Carlo policy gradient still has high variance!

Reducing Variance with Critic

Motivation: Monte-Carlo policy gradient still has high variance!

We can estimate V/Q by using a *critic*

Such critic is also parameterized

$$Q_w(s, a) \approx Q^{\pi_\theta}(s, a)$$

Reducing Variance with Critic

We can estimate V/Q by using a *critic*

Such critic is also parameterized

$$Q_w(s, a) \approx Q^{\pi_\theta}(s, a)$$

$$\nabla_\theta J(\theta) \approx \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s, a)\ Q_w(s, a)\right]$$

$$\Delta\theta = \alpha \nabla_\theta \log \pi_\theta(s, a)\ Q_w(s, a)$$

We can select any blend between TD and MC estimators for Q_w

Reducing Variance with Critic

We can estimate V/Q by using a *critic*

Such critic is also parameterized

$$Q_w(s, a) \approx Q^{\pi_\theta}(s, a)$$

$$\nabla_\theta J(\theta) \approx \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s, a)\ Q_w(s, a)\right]$$

$$\Delta\theta = \alpha \nabla_\theta \log \pi_\theta(s, a)\ Q_w(s, a)$$

Actor-Critic Algorithms use 1-step TD

MC vs TD Policy Gradient

- In MC policy gradient, the target is the return G

$$\Delta\theta = \alpha(G_t - V_v(s_t))\nabla_\theta \log \pi_\theta(s_t, a_t)$$

- In Actor-Critic the target is a TD target and relies on bootstrapping
  - Multiple timescales are possible (not only 1-step)
  - Also TD-lambda with forward/backward view

$$\Delta\theta = \alpha(r + \gamma V_v(s_{t+1}) - V_v(s_t))\nabla_\theta \log \pi_\theta(s_t, a_t)$$

Credits: David Silver

Actor-Critic VS Baseline

REINFORCE with Baseline (episodic), for estimating $\pi_{\theta} \approx \pi_*$

Input: a differentiable policy parameterization $\pi(a|s, \boldsymbol{\theta})$
Input: a differentiable state-value function parameterization $\hat{v}(s,\mathbf{w})$
Algorithm 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):
  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 $t = 0, 1, \ldots, T-1$:
    $\boxed{G \leftarrow \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k}$ (G_t)
    $\delta \leftarrow G - \hat{v}(S_t,\mathbf{w})$
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}} \delta \nabla \hat{v}(S_t,\mathbf{w})$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} \gamma^t \delta \nabla \ln \pi(A_t|S_t, \boldsymbol{\theta})$

One-step Actor–Critic (episodic), for estimating $\pi_{\theta} \approx \pi_*$

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

Actor-Critic with LFA

Critic $Q_w(s,a) = \phi(s,a)^T w$ updates weights w by linear TD(0)
Actor updates weights by policy gradient

function QAC
  Initialise s, θ
  Sample $a \sim \pi_\theta$
  for each step **do**
    Sample reward $r = \mathcal{R}_s^a$; sample transition $s' \sim \mathcal{P}_{s,\cdot}^a$
    Sample action $a' \sim \pi_\theta(s', a')$
    $\delta = r + \gamma Q_w(s', a') - Q_w(s, a)$
    $\theta = \theta + \alpha \nabla_\theta \log \pi_\theta(s, a) Q_w(s, a)$
    $w \leftarrow w + \beta \delta \phi(s, a)$
    $a \leftarrow a', s \leftarrow s'$
  end for
end function

Credits: David Silver

Eligibility Traces in Policy Gradient

Integrating eligibility traces in policy gradient (actor has multiple timescales)

$$
\begin{aligned}
\delta &= r_{t+1} + \gamma V_v(s_{t+1}) - V_v(s_t) \\
e_{t+1} &= \lambda e_t + \boxed{\nabla_\theta \log \pi_\theta(s, a)} \\
\Delta\theta &= \alpha \delta e_t
\end{aligned}
$$

Policy Gradient Summary

$$\begin{aligned}
\nabla_\theta J(\theta) &= \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s,a) G_t\right] && \text{REINFORCE} \\
&= \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s,a)\; Q^w(s,a)\right] && \text{Q Actor-Critic} \\
&= \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s,a)\; A^w(s,a)\right] && \text{Advantage Actor-Critic} \\
&= \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s,a)\; \delta\right] && \text{TD Actor-Critic} \\
&= \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s,a)\; \delta e\right] && \text{TD}(\lambda)\text{ Actor-Critic}
\end{aligned}$$

Critic does policy evaluation to estimate Q, V or A using bootstrapping (*if it uses MC we do not call it a critic*)

Credits: David Silver

SAPIENZA UNIVERSITÀ DI ROMA