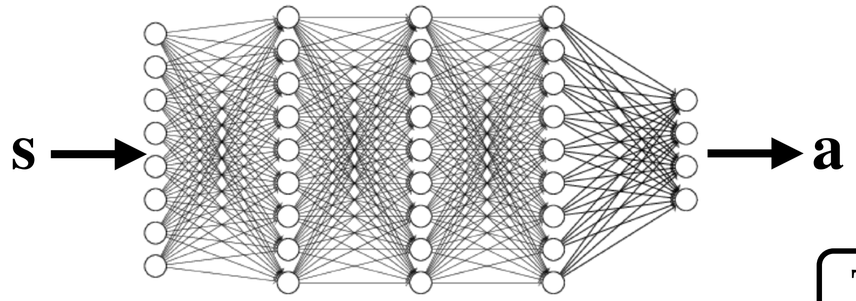
Policy Gradient Methods

The problem

Find a policy $\pi_\theta(\mathbf{a}\,|\,\mathbf{s})$ such that the expected long-term reward, $E_{\tau\sim(p,\pi_\theta)}\left[\sum_t r(\mathbf{s}_t,\mathbf{a}_t)\right]$ is maximized, where π_θ is a probability distribution conditioned on $\mathbf{s}$ and parameterized by θ.

For example, a policy can be represented by a neural network and a Gaussian distribution:



$f_\theta : \mathcal{S} \mapsto \mathcal{A}$

$$\pi_\theta(\mathbf{a}|\mathbf{s}) = \mathcal{N}(f_\theta(\mathbf{s}), \Sigma)$$

The mean is the output of Neural Network

The variance can be from a scalar, a vector, or NN.

$\pi_\theta(\mathbf{a}\,|\,\mathbf{s})$: policy

$\mathbf{s}$: state

$\mathbf{a}$: action

$p(\mathbf{s}'\,|\,\mathbf{s},\mathbf{a})$: dynamic transition

$r(\mathbf{s},\mathbf{a})$: reward function

Policy search

- Directly optimize the parameters of a policy using gradient ascent, without the need of a value function.

$$\max_{\theta} \quad J(\theta) = E_{\tau \sim (p, \pi_\theta)} \Big[\sum_t r(\mathbf{s}_t, \mathbf{a}_t) \Big]$$

Main challenge: computing the gradient!

Solve by gradient ascent iteratively: $\theta^{(k+1)} = \theta^{(k)} + \alpha \nabla_\theta J$

- It allows for a natural integration of expert knowledge, e.g. through both structure and initializations of the policy.

- Basic policy search algorithms do not leverage Bellman equations.

Quiz

- Can you estimate the policy gradient using finite difference?
- How many function evaluations do you need for 1000 parameters?

$$\theta^{(k+1)} = \theta^{(k)} + \alpha \nabla_\theta J$$

Assuming that $R_1(x)$ is sufficiently small, the approximation of the first derivative of "f" is:

$$f'(a) \approx \frac{f(a+h) - f(a)}{h}.$$

Policy gradient derivation

$$J(\theta) = E_\tau\Big[R(\tau)\Big] = \int_\tau p_\theta(\tau)R(\tau)d\tau, \text{ where } \quad R(\tau) = \sum_t^T r(\mathbf{s}_t, \mathbf{a}_t)$$

$$\nabla_\theta J = \int_\tau \nabla_\theta p_\theta(\tau)R(\tau)d\tau$$

Does $R(\tau)$ depend on θ ?

No, θ only changes the probability of τ, not its reward.

$$\nabla_\theta J = \int_\tau p_\theta(\tau)\nabla_\theta \log p_\theta(\tau)R(\tau)d\tau = E_\tau\Big[\nabla_\theta \log p_\theta(\tau)R(\tau)\Big]$$

$$\nabla_\theta \log p_\theta(\tau) = \nabla_\theta \log \big(p_1(\mathbf{s}_1)\prod_t^{T-1} \pi(\mathbf{a}_t|\mathbf{s}_t)p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)\big)$$

$$= \nabla_\theta \big(\cancel{\log p_1(\mathbf{s}_1)} + \sum_t^{T-1} \log \pi(\mathbf{a}_t|\mathbf{s}_t) + \sum_t^{T-1} \cancel{\log p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)}\big)$$

$$= \sum_t^{T-1} \nabla_\theta \log \pi(\mathbf{a}_t|\mathbf{s}_t)$$

Which terms are independent from θ?

Likelihood-ratio identity:

$$\nabla_x \log f(x) = \frac{\nabla_x f(x)}{f(x)}$$

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

replace f with p and x with θ

$$\nabla_\theta p_\theta = p_\theta \nabla_\theta \log p_\theta$$

Policy gradient derivation

$$\nabla_\theta J = E_\tau \Big[\nabla_\theta \log p_\theta(\tau) R(\tau) \Big]$$

$$= E_\tau \Big[\sum_t^{T-1} \nabla_\theta \log \pi_\theta(\mathbf{a}_t | \mathbf{s}_t) \sum_t^{T} r(\mathbf{s}_t, \mathbf{a}_t) \Big]$$

gradient does not depend on dynamics!

$$= \frac{1}{N} \sum_i^N \big(\sum_t^{T-1} \nabla_\theta \log \pi_\theta(\mathbf{a}_t^i | \mathbf{s}_t^i) \sum_t^{T} r(\mathbf{s}_t^i, \mathbf{a}_t^i) \big)$$

- Sample N rollouts by executing the current policy.
- Evaluate rewards of each rollout and sum them up: $\sum_t^T r(\mathbf{s}_t^i, \mathbf{a}_t^i)$
- Evaluate $\nabla_\theta \log \pi_\theta$ at each step in the rollout and sum them up: $\sum_t^{T-1} \nabla_\theta \log \pi_\theta(\mathbf{a}_t^i | \mathbf{s}_t^i)$

Quiz

- Given a policy represented as a Gaussian distribution, whose mean is the output of a neural network and whose covariance is constant:

neural network

$$\pi_\theta(\mathbf{a}|\mathbf{s}) = \mathcal{N}(\bar{\mathbf{a}}, \Sigma), \text{ where } \bar{\mathbf{a}} = f_\theta(\mathbf{s})$$

Compute $\nabla_\theta \log \pi_\theta(\mathbf{a}|\mathbf{s})$

Normal distribution:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2}$$

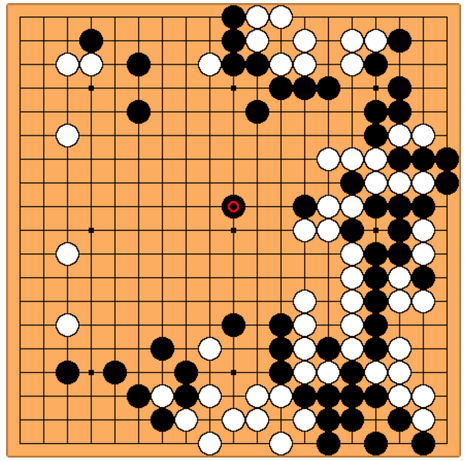
Answer:

$$\log \pi_\theta(\mathbf{a}\,|\,\mathbf{s}) = -\frac{1}{2}\|f(s) - a\|^2_\Sigma + \text{const}$$

$$\nabla \log \pi_\theta(\mathbf{a}\,|\,\mathbf{s}) = -\Sigma^{-1}(f(s) - a)\frac{\partial f}{\partial \theta}$$

Quiz

- Can you guess what do we need to change for partially observable MDPs?



Partially observable policy

- Partial observation doesn't change the calculation of policy gradient; substituting **s** with **o**, everything will still work.

$$\nabla_\theta J \approx \frac{1}{N} \sum_i^N \Big(\sum_t^{T-1} \nabla_\theta \log \pi_\theta(\mathbf{a}_t^i | \mathbf{o}_t^i) \Big) \Big(\sum_t^T r(\mathbf{o}_t^i, \mathbf{a}_t^i) \Big)$$

- But the policy might not perform as well because POMDP is in general a harder problem than MDP.

- We can include the history of state-action pairs to make the problem ``more MDP", or we can use a recurrent neural network to represent the policy.

REINFORCE

while not converge

sample N rollouts by executing π_θ

evaluate gradient using rollouts: $\nabla_\theta J = \frac{1}{N} \sum_i^N \left(\sum_t^{T-1} \nabla_\theta \log \pi_\theta(\mathbf{a}_t^i | \mathbf{s}_t^i) \sum_t^T r(\mathbf{s}_t^i, \mathbf{a}_t^i) \right)$

update policy: $\theta^{(k+1)} = \theta^{(k)} + \alpha \nabla_\theta J$

direction that makes rollout i more likely

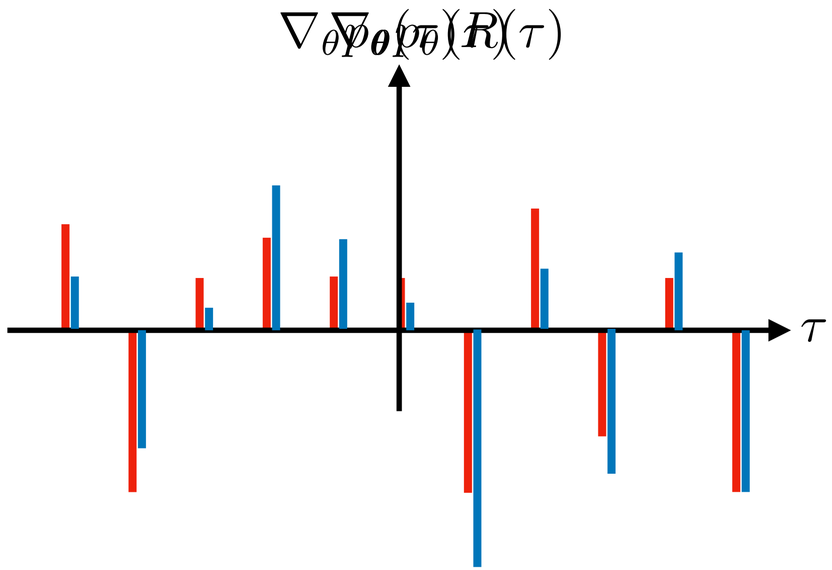
weight for rollout i

Policy update strategy:
- Increase the chance of a rollout that performs better
- Decrease the change of a rollout that performs poorly

Variance

- Reward scales the gradient of the log probability.

$$\nabla_\theta J \approx \frac{1}{N} \sum_i^N \Big(\sum_t^{T-1} \nabla_\theta \log \pi_\theta(\mathbf{a}_t^i | \mathbf{o}_t^i) \Big) \Big(\sum_t^T r(\mathbf{o}_t^i, \mathbf{a}_t^i) \Big)$$



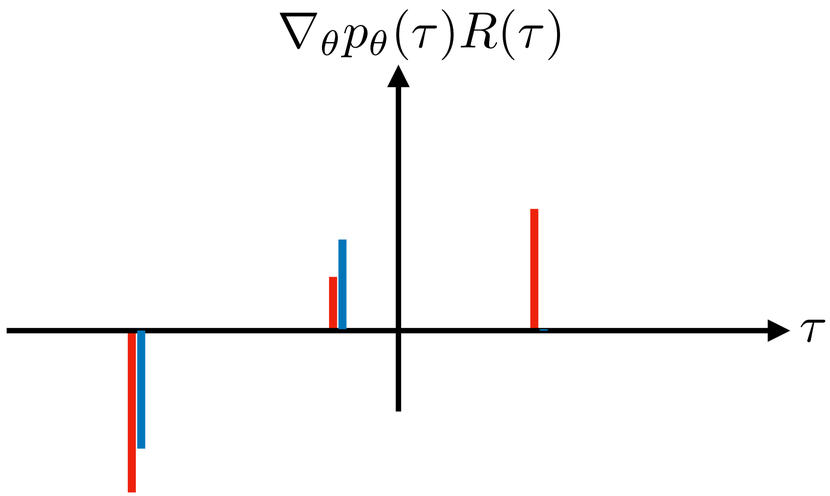
each red bar is scaled by its R

Quiz

- Let's assume we have three trajectories and the maximum reward of three episodes is zero. Why is it bad?

$$\nabla_\theta J \approx \frac{1}{N} \sum_i^N \Big(\sum_t^{T-1} \nabla_\theta \log \pi_\theta(\mathbf{a}_t^i | \mathbf{o}_t^i) \Big) \Big(\sum_t^T r(\mathbf{o}_t^i, \mathbf{a}_t^i) \Big)$$

Ex) 0.7, 1.5, 0.0



We lost the most important trajectory in the gradient!

Reduce variance: Baseline

$$\nabla_\theta J = E_\tau\Big[\nabla_\theta \log p_\theta(\tau) R(\tau)\Big]$$

expectation varies greatly based on the sampled rollouts

Baseline technique

Claim: adding a constant term to the total reward of a rollout does not change the gradient

$$\nabla_\theta J = E_\tau\Big[\nabla_\theta \log p_\theta(\tau) R(\tau)\Big] = E_\tau\Big[\nabla_\theta \log p_\theta(\tau)\big(R(\tau) - b\big)\Big]$$

Proof: consider only the second term

likelihood-ratio identity

$$E_\tau\Big[\nabla_\theta \log p_\theta(\tau) b\Big] = \int_\tau p_\theta(\tau) \nabla_\theta \log p_\theta(\tau) b \; d\tau = b \int_\tau \nabla_\theta p_\theta(\tau) d\tau$$

$$= b \nabla_\theta \int_\tau p_\theta(\tau) d\tau = 0$$

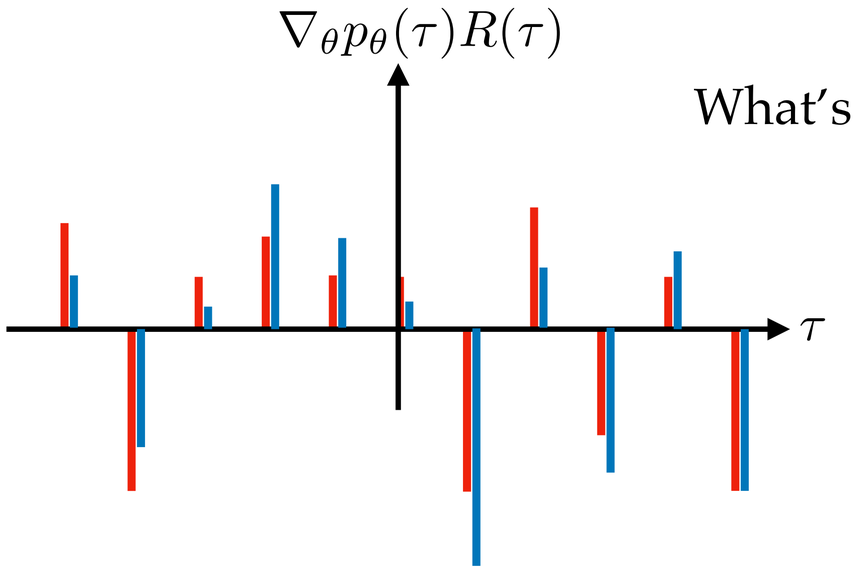
$$\int_\tau p_\theta(\tau) d\tau = 1$$

Reduce variance: Baseline

Now consider the first term of $\nabla_\theta J = E_\tau\Big[\nabla_\theta \log p_\theta(\tau)\big(R(\tau) - b\big)\Big]$

$$E_\tau\Big[\nabla_\theta \log p_\theta(\tau)\big(R(\tau)\big)\Big] = \int_\tau \nabla_\theta p_\theta(\tau) R(\tau) d\tau$$

Facts:
- red bars must sum up to zero
- each red bar is scaled by its R
- b modify the scaling of each bar



What's the optimal b such that the variance of blue bars is minimized?

We can compute the optimal baseline by computing the variance w.r.t. b and solving for the optimality condition.

A simpler baseline that works reasonably well:

$$b = \frac{1}{N}\sum_i^N R(\tau^i)$$

Reduce variance: Causality

$$\nabla_\theta J = \frac{1}{N} \sum_i^N \Big(\sum_t^{T-1} \nabla_\theta \log \pi_\theta(\mathbf{a}_t^i | \mathbf{s}_t^i) \sum_t^T r(\mathbf{s}_t^i, \mathbf{a}_t^i) \Big)$$

The variance can grow with the length of rollouts

Causality: actions can only affect the current and future rewards

$$\nabla_\theta J = \frac{1}{N} \sum_i^N \sum_t^{T-1} \nabla_\theta \log \pi_\theta(\mathbf{a}_t^i | \mathbf{s}_t^i) \Big(\sum_{t'=1}^T r(\mathbf{s}_{t'}^i, \mathbf{a}_{t'}^i) \Big)$$

We can start with t′ = t due to causality.

$$\nabla_\theta J = \frac{1}{N} \sum_i^N \sum_t^{T-1} \nabla_\theta \log \pi_\theta(\mathbf{a}_t^i | \mathbf{s}_t^i) \Big(\sum_{t'=t}^T r(\mathbf{s}_{t'}^i, \mathbf{a}_{t'}^i) \Big)$$

Reward to go, that is, the Q function

$$= \frac{1}{N} \sum_i^N \sum_t^{T-1} \nabla_\theta \log \pi_\theta(\mathbf{a}_t^i | \mathbf{s}_t^i) Q(\mathbf{s}_t^i, \mathbf{a}_t^i)$$

Quiz

- Is the policy gradient on-policy or off-policy?

Only use the samples from the current policy

Use the samples from any policy

Natural policy gradients

- Small changes in θ can induce large changes in trajectory distribution.
- Reducing the magnitude of change , $\delta\theta^T\delta\theta$, is ineffective because it assumes all parameter dimensions have similar effect on the resulting trajectory distribution.
- To improve learning stability, we need to regularize the change in trajectory distribution directly:

$$\max_{\theta} J(\theta) \quad \text{subject to} \quad D_{KL}(\pi_\theta \| \pi_{\theta_{old}}) \leq \delta$$

Approximated solution

$$J(\theta) \approx J(\theta_{old}) + \mathbf{g}^T(\theta - \theta_{old}), \text{ where } \mathbf{g} = \nabla_\theta J_{|\theta=\theta_{old}}$$

$$D_{KL}(\pi_\theta \| \pi_{\theta_{old}}) \approx \frac{1}{2}(\theta - \theta_{old})^T H(\theta - \theta_{old})$$

$$\text{where } H = E_{(\mathbf{s},\mathbf{a})\sim p_{\theta_{old}}}\left[\nabla_\theta \log \pi_{\theta_{old}}(\mathbf{a}|\mathbf{s}) \nabla_\theta \log \pi_{\theta_{old}}(\mathbf{a}|\mathbf{s})^T\right]$$

known as Fisher information matrix

Problem:

$$\max_\theta \mathbf{g}^T(\theta - \theta_k)$$

$$\text{subject to } \frac{1}{2}(\theta - \theta_k)^T H(\theta - \theta_k) \leq \delta$$

Solution:

$$\theta_{k+1} = \theta_k + \sqrt{\frac{2\delta}{\mathbf{g}^T H^{-1} \mathbf{g}}} H^{-1}\mathbf{g}$$

Trust region policy optimization (TRPO)

- Built on the concept of natural gradients, TRPO is an on-policy policy gradient method that guarantees monotonic improvement.

- Policy update: $$\max_{\boldsymbol{\theta}} \mathbb{E}_{\mathbf{s}\sim\rho_{\boldsymbol{\theta}_{old}},\mathbf{a}\sim\pi_{\boldsymbol{\theta}_{old}}}[\frac{\pi_{\boldsymbol{\theta}}(\mathbf{a}|\mathbf{s})}{\pi_{\boldsymbol{\theta}_{old}}(\mathbf{a}|\mathbf{s})}A_{old}(\mathbf{s},\mathbf{a})]$$

$$\text{s.t.} \quad \mathbb{E}_{\mathbf{s}\sim\rho_{\boldsymbol{\theta}_{old}}}[D_{KL}(\pi_{\boldsymbol{\theta}_{old}}||\pi_{\boldsymbol{\theta}})] \leq \sigma$$

The actual implementation: 1. Compute $\mathbf{d} = \mathbf{H}^{-1}\mathbf{g}$, where $\mathbf{g} = \frac{1}{N}\sum_i \nabla_\theta \log \pi_{\boldsymbol{\theta}}(\mathbf{a}_i|\mathbf{s}_i)A_{old}(\mathbf{s}_i,\mathbf{a}_i)$

2. Compute $\Delta\mathbf{d} = \sqrt{\frac{2\varepsilon}{\mathbf{g}^T\mathbf{H}^{-1}\mathbf{g}}}\mathbf{d}$

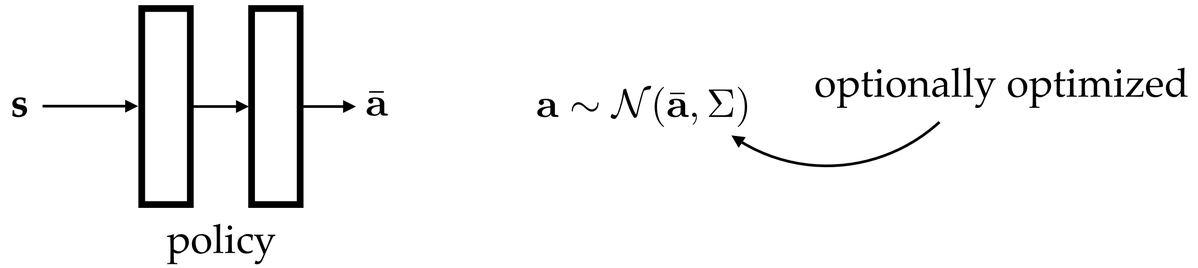
$\frac{\pi_{\boldsymbol{\theta}}(\mathbf{a}|\mathbf{s})}{\pi_{\boldsymbol{\theta}_{old}}(\mathbf{a}|\mathbf{s})} = 1$ when $\mathbf{g}$ is evaluated

3. Line-search such that $\boldsymbol{\theta} = \boldsymbol{\theta} + \beta\Delta\mathbf{d}$ reduces loss and satisfies KL divergence

TRPO

- Remark: On-policy because the policy update uses the current policy to generate samples to estimate the current gradient. The importance sampling in the objective function is only needed for objective function evaluation during the line-search procedure. Since the gradient is computed once at the beginning of each iteration, the importance sampling ratio will simply be 1 during **gradient computation**. However, the ratio will be different for **objective function evaluation** during line-search.

- Architecture:



Generalized advantage estimator (GAE)

- A general technique to balance the variance and bias in estimating advantage function
- Can be applied to any policy optimization method that requires advantage estimation
- Policy update:

$$\boldsymbol{\theta} = \boldsymbol{\theta} + \alpha \mathbb{E}[\sum_t \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\mathbf{a}_t|\mathbf{s}_t)\hat{A}_t]$$

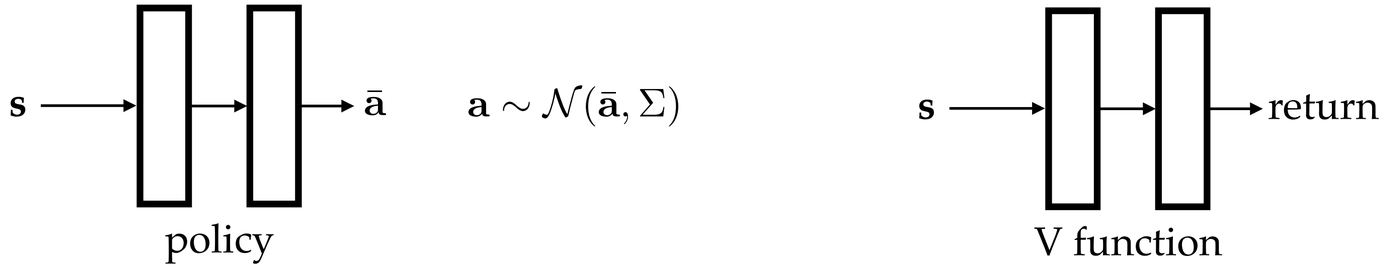
$$\hat{A}_t = \sum_l (\gamma\lambda)^l (r_{t+l} + \gamma V(\mathbf{s}_{t+l+1}) - V(\mathbf{s}_{t+l}))$$

- Value function update: $\min_{\boldsymbol{\phi}} \sum_i^N \|V_{\boldsymbol{\phi}}(\mathbf{s}_i) - \hat{V}_i\|^2$

GAE

- GAE uses V for two purposes: approximating Q and reducing variance. The former introduces bias while the latter does not.

- Architecture:



Proximal policy optimization (PPO)

- Stochastic, on-policy, continuous actions, learn policy and value function
- Exploration: stochastic on-policy $\pi(\mathbf{a}|\mathbf{s})$
- Policy update: $\boldsymbol{\theta} = \boldsymbol{\theta} + \alpha \nabla_{\boldsymbol{\theta}} L$

$$L = \mathbb{E}[\sum_t \min(r\hat{A}_t, \text{clip}(1-\epsilon, 1+\epsilon, r)\hat{A}_t)]$$

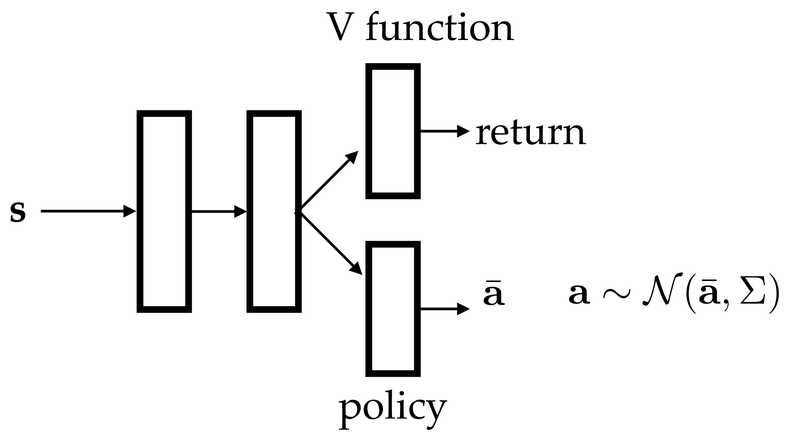
$$r = \frac{\pi_{\boldsymbol{\theta}}(\mathbf{a}_t|\mathbf{s}_t)}{\pi_{\boldsymbol{\theta_{old}}}(\mathbf{a}_t|\mathbf{s}_t)} \qquad \hat{A}_t = -V(\mathbf{s}_t) + \sum_{l=0}^{T-t-1} \gamma^l r_{t+l} + \gamma^{T-t} V(\mathbf{s}_T)$$

- Value function update: If value function and policy don't share parameters, update value function like GAE. Otherwise, change L above to include $(V_{\boldsymbol{\theta}}(\mathbf{s}_t) - \hat{V}_t)^2$

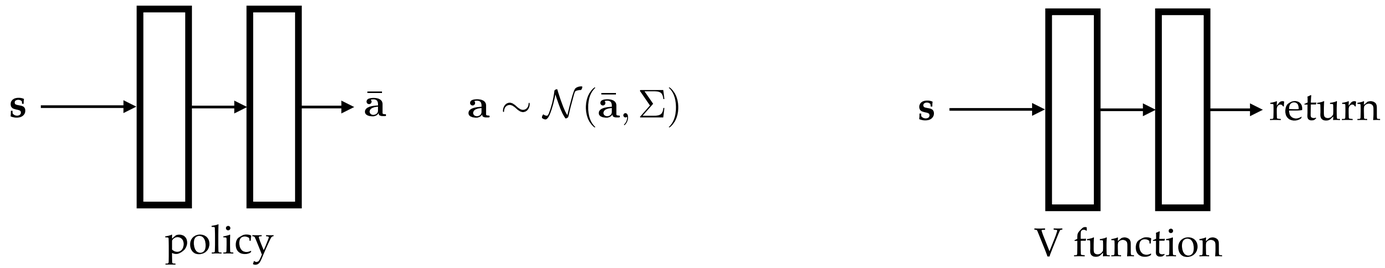
PPO

- Remark: Removing KL constraint makes policy update a lot easier. It also makes possible the parameter sharing between policy and value function.

- Architecture:



or



Reading List

Policy Gradient with monotonic improvement

- Paper #1: Schulman, J., Levine, S., Abbeel, P., Jordan, M. and Moritz, P., 2015, June. Trust region policy optimization. In International conference on machine learning (pp. 1889-1897). PMLR.

Extension with GAE

- Paper #2: Schulman, J., Moritz, P., Levine, S., Jordan, M. and Abbeel, P., 2015. High-dimensional continuous control using generalized advantage estimation. arXiv preprint arXiv:1506.02438.

Simple trick for faster convergence

- Paper #3: Schulman, J., Wolski, F., Dhariwal, P., Radford, A. and Klimov, O., 2017. Proximal policy optimization algorithms.

A bit different thinking about policy gradient

- Paper #4: Mania, Horia, Aurelia Guy, and Benjamin Recht. "Simple random search provides a competitive approach to reinforcement learning." arXiv preprint arXiv:1803.07055 (2018).

Classical paper of REINFORCE

- Runner-up: Williams, R.J., 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. Machine learning, 8(3), pp.229-256.

Summary

- Policy gradient is an effective method for solving MDPs (if you have enough samples).
- Variance reduction is the key.
- Works well on many high-dimensional continuous problems.