EECS 545: Machine Learning

Lecture 24. Reinforcement Learning 3

Honglak Lee and Michał Dereziński

04/06/2022

Outline: Policy-based RL Methods

- Introduction
- REINFORCE
- Actor-Critic methods
  - Advantage Actor Critic (A2C & A3C)
- Advanced policy gradient methods
  - Natural policy gradient (NPG)
  - Proximal policy optimization (PPO)

Policy gradients

Formally, let's define a class of parametrized policies: $\Pi = \{\pi_\theta, \theta \in \mathbb{R}^m\}$

For each policy, define its value:

$$J(\theta) = \mathbb{E}\left[\sum_{t \geq 0} \gamma^t r_t | \pi_\theta\right]$$

We want to find the optimal policy $\theta^* = \arg\max_\theta J(\theta)$

How can we do this?

Gradient ascent on policy parameters!

Slide credit: Fei fei Li & Justin Johnson & Serena Yeung

Intuition

Gradient estimator: $\nabla_\theta J(\theta) \approx \sum_{t \geq 0} r(\tau) \nabla_\theta \log \pi_\theta(a_t|s_t)$

Interpretation:

- If r(τ) is high, push up the probabilities of the actions seen
- If r(τ) is low, push down the probabilities of the actions seen

Might seem simplistic to say that if a trajectory is good then all its actions were good. But in expectation, it averages out!

However, this also suffers from high variance because credit assignment is really hard. Can we help the estimator?

Slide credit: Fei fei Li & Justin Johnson & Serena Yeung

Variance reduction

Gradient estimator: $\nabla_\theta J(\theta) \approx \sum_{t \geq 0} r(\tau) \nabla_\theta \log \pi_\theta(a_t|s_t)$

First idea: Push up probabilities of an action seen, only by the cumulative future reward from that state

$$\nabla_\theta J(\theta) \approx \sum_{t \geq 0} \left(\sum_{t' \geq t} r_{t'} \right) \nabla_\theta \log \pi_\theta(a_t|s_t)$$

Second idea: Use discount factor γ to ignore delayed effects

$$\nabla_\theta J(\theta) \approx \sum_{t \geq 0} \left(\sum_{t' \geq t} \gamma^{t'-t} r_{t'} \right) \nabla_\theta \log \pi_\theta(a_t|s_t)$$

Slide credit: Fei fei Li & Justin Johnson & Serena Yeung

Variance reduction: Baseline

Problem: The raw value of a trajectory isn't necessarily meaningful. For example, if rewards are all positive, you keep pushing up probabilities of actions.

What is important then? Whether a reward is better or worse than what you expect to get

Idea: Introduce a baseline function dependent on the state.
Concretely, estimator is now:

$$\nabla_\theta J(\theta) \approx \sum_{t \geq 0} \left(\sum_{t' \geq t} \gamma^{t'-t} r_{t'} - b(s_t) \right) \nabla_\theta \log \pi_\theta(a_t|s_t)$$

Slide credit: Fei fei Li & Justin Johnson & Serena Yeung

How to choose the baseline?

$$\nabla_\theta J(\theta) \approx \sum_{t \geq 0} \left(\sum_{t' \geq t} \gamma^{t'-t} r_{t'} - b(s_t) \right) \nabla_\theta \log \pi_\theta(a_t|s_t)$$

A simple baseline: constant moving average of rewards experienced so far from all trajectories

Slide credit: Fei fei Li & Justin Johnson & Serena Yeung

How to choose the baseline?

$$\nabla_\theta J(\theta) \approx \sum_{t \geq 0} \left(\sum_{t' \geq t} \gamma^{t'-t} r_{t'} - b(s_t) \right) \nabla_\theta \log \pi_\theta(a_t | s_t)$$

A simple baseline: constant moving average of rewards experienced so far from all trajectories

A better baseline: Want to push up the probability of an action from a state, if this action was better than the **expected value of what we should get from that state**.

Q: What does this remind you of?

Slide credit: Fei fei Li & Justin Johnson & Serena Yeung

How to choose the baseline?

$$\nabla_\theta J(\theta) \approx \sum_{t \geq 0} \left(\sum_{t' \geq t} \gamma^{t'-t} r_{t'} - b(s_t) \right) \nabla_\theta \log \pi_\theta(a_t|s_t)$$

A simple baseline: constant moving average of rewards experienced so far from all trajectories

A better baseline: Want to push up the probability of an action from a state, if this action was better than the **expected value of what we should get from that state**.

Q: What does this remind you of?

A: Q-function and value function!

Slide credit: Fei fei Li & Justin Johnson & Serena Yeung

How to choose the baseline?

$$\nabla_\theta J(\theta) \approx \sum_{t \geq 0} \left(\sum_{t' \geq t} \gamma^{t'-t} r_{t'} - b(s_t) \right) \nabla_\theta \log \pi_\theta(a_t|s_t)$$

Option 1: Instead of moving average of cumulative rewards, use a value function estimate $V_\phi(s_t) \approx V^{\pi_\theta}(s_t)$

$$\nabla_\theta J(\theta) \approx \sum_{t \geq 0} \left(\sum_{t' \geq t} \gamma^{t'-t} r_{t'} - V_\phi(s_t) \right) \nabla_\theta \log \pi_\theta(a_t|s_t)$$

where

$$V_\phi(s_t) \approx V^{\pi_\theta}(s_t) = \mathbb{E}_{a_t \sim \pi_\theta(a|s_t)} \left[\sum_{t' \geq t} \gamma^{t'-t} r_{t'} | s_t \right]$$

← minimize L2 error (with SGD)

Actor-Critic Algorithm

Initialize policy parameters θ, critic parameters ϕ

For iteration $= 1, 2 \ldots$ **do**

Sample m trajectories under the current policy

$\Delta\theta \leftarrow 0$

For i $= 1, \ldots,$ m **do**

For t $= 1, \ldots,$ T **do**

$$A_t^i = \sum_{t' \geq t} \gamma^{t'-t} r_t^i - V_\phi(s_t^i)$$

$$\Delta\theta \leftarrow \Delta\theta + A_t^i \Delta_\theta \log \pi_\theta(a_t^i | s_t^i)$$

$$\Delta\phi \leftarrow \sum_i \sum_t \nabla_\phi ||A_t^i||^2$$

$$\theta \leftarrow \theta + \alpha\Delta\theta$$

$$\phi \leftarrow \phi - \beta\Delta\phi$$

Slide credit: Fei fei Li & Justin Johnson & Serena Yeung

How to choose the baseline?

A better baseline: Want to push up the probability of an action from a state, if this action was better than the **expected value of what we should get from that state**.

Q: What does this remind you of?

A: Q-function and value function! (Option 2: Use both!)

Intuitively, we are happy with an action a_t in a state s_t if $Q^\pi(s_t, a_t) - V^\pi(s_t)$ is large. On the contrary, we are unhappy with an action if it's small.

Slide credit: Fei fei Li & Justin Johnson & Serena Yeung

How to choose the baseline?

A better baseline: Want to push up the probability of an action from a state, if this action was better than the **expected value of what we should get from that state**.

Q: What does this remind you of?

A: Q-function and value function! (Option 2: Use both!)

Intuitively, we are happy with an action a_t in a state s_t if $Q^{\pi}(s_t, a_t) - V^{\pi}(s_t)$ is large. On the contrary, we are unhappy with an action if it's small.

Using this, we get the estimator: $\nabla_\theta J(\theta) \approx \sum_{t \geq 0} (Q^{\pi_\theta}(s_t, a_t) - V^{\pi_\theta}(s_t)) \nabla_\theta \log \pi_\theta(a_t | s_t)$

Slide credit: Fei fei Li & Justin Johnson & Serena Yeung

Advantage Actor-Critic algorithm (A2C)

$$Q^{\pi}(\mathbf{s}_t, \mathbf{a}_t) = \mathbb{E}_{\pi_\theta}[\sum_{t' \geq t} \gamma^{t'-t} r(\mathbf{s}_{t'}, \mathbf{a}_{t'}) | \mathbf{s}_t, \mathbf{a}_t]$$ Total reward from taking a_t in s_t

$$V^{\pi}(\mathbf{s}_t) = E_{\mathbf{a}_t \sim \pi_\theta(\mathbf{a}_t|\mathbf{s}_t)}[Q^{\pi}(\mathbf{s}_t, \mathbf{a}_t)]$$ Total reward from s_t

$$A^{\pi}(\mathbf{s}_t, \mathbf{a}_t) = Q^{\pi}(\mathbf{s}_t, \mathbf{a}_t) - V^{\pi}(\mathbf{s}_t)$$ How much better a_t is

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t^i | s_t^i) A^{\pi}(s_t^i, a_t^i)$$

For further simplification, we use the following properties:

$$Q^{\pi}(s_t, a_t) = r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1} \sim p(s_{t+1}|s_t, a_t)}[V^{\pi}(s_{t+1})] \approx r(s_t, a_t) + \gamma V^{\pi}(s_{t+1})$$

$$A^{\pi}(s_t, a_t) \approx r(s_t, a_t) + \gamma V^{\pi}(s_{t+1}) - V^{\pi}(s_t)$$

We can learn $V^{\pi}(\mathbf{s})$ by minimizing $\mathcal{L}(\phi) = ||V_\phi(s_t) - y||^2$ where $y = r(s_t, a_t) + \gamma V_\phi(s_{t+1})$

Advantage Actor-Critic algorithm (A2C)

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t^i | s_t^i) A^\pi(s_t^i, a_t^i)$$

- Advantage estimate $A^\pi(s_t, a_t) \approx r(s_t, a_t) + \gamma V^\pi(s_{t+1}) - V^\pi(s_t)$
- It is also common to take n-step rewards into account

$$A^\pi(s_t, a_t) \approx \sum_{t'=t}^{t+n-1} \gamma^{t'-t} r(s_{t'}, a_{t'}) + \gamma^n V^\pi(s_{t+n}) - V^\pi(s_t)$$

- Variance of the estimator increases with n
- Can be tuned as a hyperparameter

Asynchronous Advantage Actor-Critic algorithm (A3C)

Algorithm S3 Asynchronous advantage actor-critic - pseudocode for each actor-learner thread.

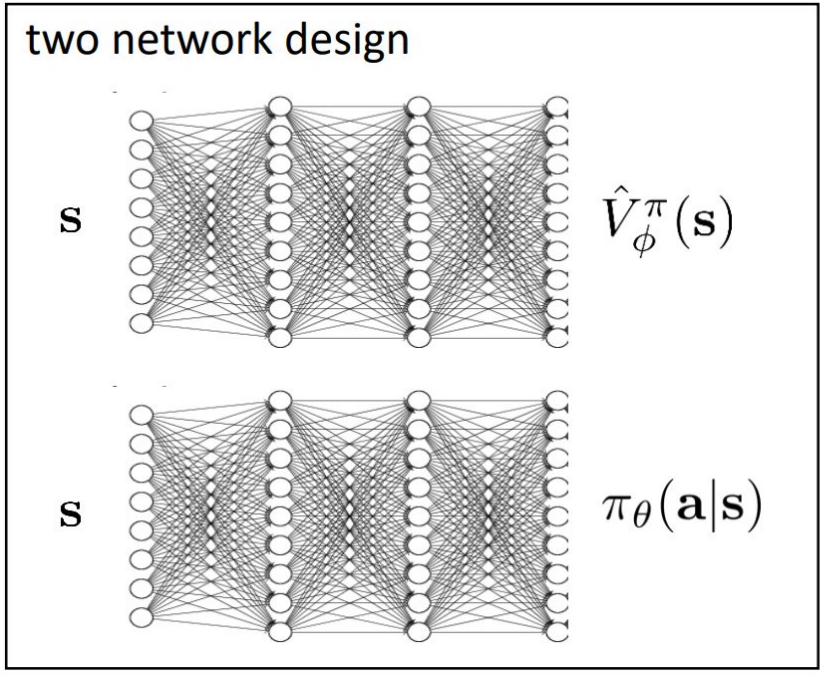
// Assume global shared parameter vectors θ *and* θ_v *and global shared counter* $T = 0$
// Assume thread-specific parameter vectors θ' *and* θ'_v
Initialize thread step counter $t \leftarrow 1$
repeat
  Reset gradients: $d\theta \leftarrow 0$ and $d\theta_v \leftarrow 0$.
  Synchronize thread-specific parameters $\theta' = \theta$ and $\theta'_v = \theta_v$
  $t_{start} = t$
  Get state s_t
  repeat
    Perform a_t according to policy $\pi(a_t|s_t;\theta')$
    Receive reward r_t and new state s_{t+1}
    $t \leftarrow t + 1$
    $T \leftarrow T + 1$
  until terminal s_t **or** $t - t_{start} == t_{max}$
  $R = \begin{cases} 0 & \text{for terminal } s_t \\ V(s_t, \theta'_v) & \text{for non-terminal } s_t \text{// Bootstrap from last state} \end{cases}$
  for $i \in \{t-1, \ldots, t_{start}\}$ **do**
    $R \leftarrow r_i + \gamma R$
    Accumulate gradients wrt θ': $d\theta \leftarrow d\theta + \nabla_{\theta'} \log \pi(a_i|s_i;\theta')(R - V(s_i;\theta'_v))$
    Accumulate gradients wrt θ'_v: $d\theta_v \leftarrow d\theta_v + \partial \left(R - V(s_i;\theta'_v)\right)^2 / \partial \theta'_v$
  end for
  Perform asynchronous update of θ using $d\theta$ and of θ_v using $d\theta_v$.
until $T > T_{max}$

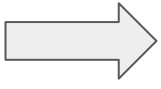
Mnih et al., ICML 2016

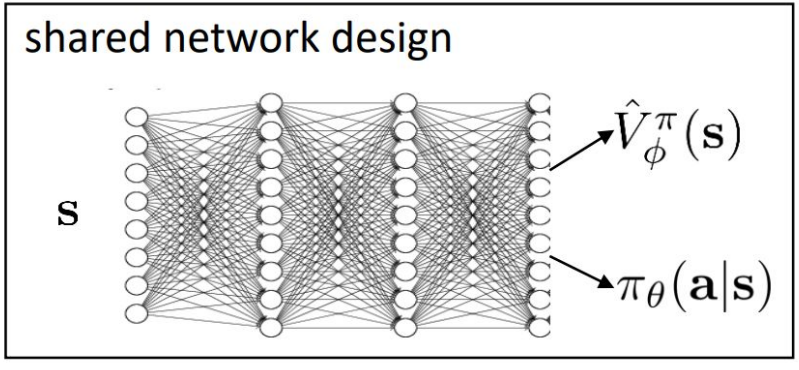
Network Design

+ simple & stable

- no shared features between actor & critic

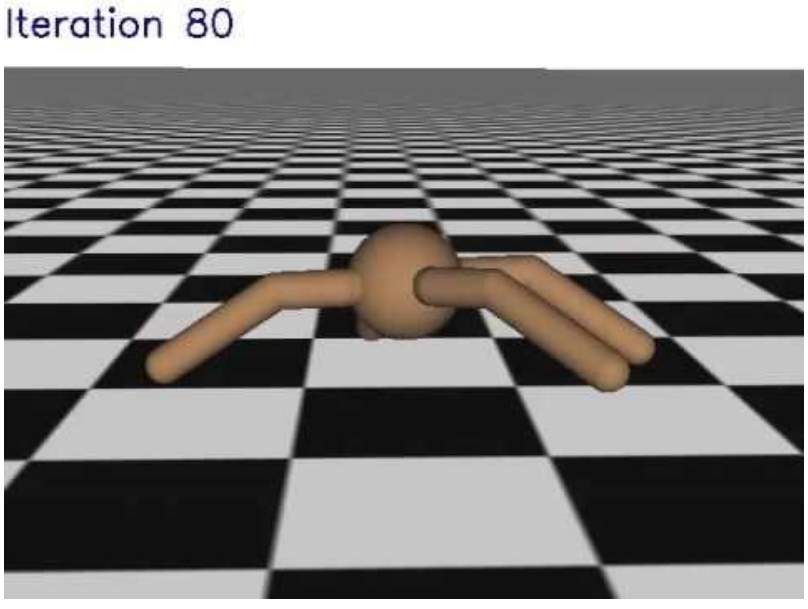






Actor-critic examples

- Batch-mode actor-critic
- Blends Monte Carlo and function approximator estimators (GAE)



https://www.youtube.com/watch?v=SHLuf2ZBQSw

Schulman, J., Moritz, P., Levine, S., Jordan, M., & Abbeel, P. (2015). High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*.

Slide credit: Sergey Levine

Actor-critic examples

- Online actor-critic, parallelized batch
- N-step returns with N = 4
- Single network for actor and critic

https://youtu.be/nMR5mjCFZCw

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D. & Kavukcuoglu, K. (2016, June). Asynchronous methods for deep reinforcement learning. In *International conference on machine learning* (pp. 1928-1937).

Slide credit: Sergey Levine

Outline: Policy-based RL Methods

- Introduction
- REINFORCE
- Actor-Critic methods
  - Advantage Actor Critic (A2C & A3C)
- Advanced policy gradient methods
  - Natural policy gradient (NPG)
  - Proximal policy optimization (PPO)

Relative performance of two policies

In a policy optimization algorithm, we want an update step that

- Uses trajectories based on the most recent policy as efficiently as possible.
- Takes steps that respect **distance in policy space** as opposed to distance in parameter space.

To figure out the right update rule, we need to exploit relationships between the performance of two policies.

Relative policy performance identity: for any policies π, π′

$$J(\pi') - J(\pi) = \underset{\tau \sim \pi'}{\mathrm{E}} \left[\sum_{t=0}^{\infty} \gamma^t A^{\pi}(s_t, a_t) \right]$$

Slide credit: Joshua Achiam

Relative performance of two policies

Proof:

$$
\begin{aligned}
\mathop{\mathrm{E}}_{\tau\sim\pi'}\left[\sum_{t=0}^{\infty}\gamma^t A^{\pi}(s_t,a_t)\right] &= \mathop{\mathrm{E}}_{\tau\sim\pi'}\left[\sum_{t=0}^{\infty}\gamma^t\left(R(s_t,a_t,s_{t+1})+\gamma V^{\pi}(s_{t+1})-V^{\pi}(s_t)\right)\right] \\
&= J(\pi') + \mathop{\mathrm{E}}_{\tau\sim\pi'}\left[\sum_{t=0}^{\infty}\gamma^{t+1}V^{\pi}(s_{t+1})-\sum_{t=0}^{\infty}\gamma^t V^{\pi}(s_t)\right] \\
&= J(\pi') + \mathop{\mathrm{E}}_{\tau\sim\pi'}\left[\sum_{t=1}^{\infty}\gamma^t V^{\pi}(s_t)-\sum_{t=0}^{\infty}\gamma^t V^{\pi}(s_t)\right] \\
&= J(\pi') - \mathop{\mathrm{E}}_{\tau\sim\pi'}\left[V^{\pi}(s_0)\right] \\
&= J(\pi') - J(\pi)
\end{aligned}
$$

Slide credit: Joshua Achiam

Relative performance of two policies

Can we use this for policy improvement, where π′ represents the new policy and π represents the old one?

$$\max_{\pi'} J(\pi') = \max_{\pi'} J(\pi') - J(\pi)$$

$$= \max_{\pi'} \mathop{\mathrm{E}}_{\tau \sim \pi'} \left[\sum_{t=0}^{\infty} \gamma^t A^{\pi}(s_t, a_t) \right]$$

This is suggestive but not useful yet.

Nice feature of this optimization problem: defines the performance of π′ in terms of the advantages of π!

But problematic feature: still requires trajectories sampled from π′….

Slide credit: Joshua Achiam

Relative performance of two policies

In terms of the **discounted future state distribution** d^{π}, defined by

$$d^{\pi}(s) = (1-\gamma)\sum_{t=0}^{\infty}\gamma^{t}P(s_t = s|\pi),$$

we can rewrite the relative policy performance identity:

$$\begin{aligned} J(\pi') - J(\pi) &= \mathop{\mathrm{E}}_{\tau\sim\pi'}\left[\sum_{t=0}^{\infty}\gamma^{t}A^{\pi}(s_t, a_t)\right] \\ &= \frac{1}{1-\gamma}\mathop{\mathrm{E}}_{\substack{s\sim d^{\pi'} \\ a\sim\pi'}}\left[A^{\pi}(s,a)\right] \\ &= \frac{1}{1-\gamma}\mathop{\mathrm{E}}_{\substack{s\sim d^{\pi'} \\ a\sim\pi}}\left[\frac{\pi'(a|s)}{\pi(a|s)}A^{\pi}(s,a)\right] \end{aligned}$$

… almost there! The only problem is $s\sim d^{\pi'}$

Slide credit: Joshua Achiam

Relative performance of two policies

what if we just said $d^{\pi'} \approx d^{\pi}$ and did not worry about it?

$$J(\pi') - J(\pi) \approx \frac{1}{1-\gamma} \mathop{\mathrm{E}}_{\substack{s \sim d^{\pi} \\ a \sim \pi}} \left[\frac{\pi'(a|s)}{\pi(a|s)} A^{\pi}(s,a) \right]$$
$$\doteq \mathcal{L}_{\pi}(\pi')$$

Turns out: this approximation is pretty good when π′ and π are close! But why, and how close do they have to be?

Relative performance bounds:

$$\left| J(\pi') - \left(J(\pi) + \mathcal{L}_{\pi}(\pi') \right) \right| \leq C \sqrt{\mathop{\mathrm{E}}_{s \sim d^{\pi}} \left[D_{KL}(\pi' || \pi)[s] \right]}$$

If policies are close in KL-divergence — the approximation is good!

Slide credit: Joshua Achiam

Relative performance of two policies

What did we learn from making that approximation?

$$J(\pi') - J(\pi) \approx \mathcal{L}_{\pi}(\pi') = \frac{1}{1-\gamma} \mathop{\mathrm{E}}_{\substack{s\sim d^{\pi} \\ a\sim\pi}} \left[\frac{\pi'(a|s)}{\pi(a|s)} A^{\pi}(s,a) \right]$$

$$= \mathop{\mathrm{E}}_{\tau\sim\pi} \left[\sum_{t=0}^{\infty} \gamma^t \frac{\pi'(a_t|s_t)}{\pi(a_t|s_t)} A^{\pi}(s_t,a_t) \right]$$

- This is something that we can optimize using trajectories sampled from the old policy π!
- Similar to using importance sampling, but because weights only depend on current timestep (and not preceding history), they don't vanish or explode.

Something else cool—the approximation matches $J(\pi_\theta) - J(\pi_{\theta_k})$ to first order in policy parameters! That is, $\nabla_\theta \mathcal{L}_{\theta_k}(\theta)|_{\theta_k}$ is equal to policy gradient:

$$\nabla_\theta \mathcal{L}_{\theta_k}(\theta)|_{\theta_k} = \mathop{\mathrm{E}}_{\tau\sim\pi_{\theta_k}} \left[\sum_{t=0}^{\infty} \gamma^t \frac{\nabla_\theta \pi_\theta(a_t|s_t)|_{\theta_k}}{\pi_{\theta_k}(a_t|s_t)} A^{\pi_{\theta_k}}(s_t,a_t) \right]$$

$$= \mathop{\mathrm{E}}_{\tau\sim\pi_{\theta_k}} \left[\sum_{t=0}^{\infty} \gamma^t \nabla_\theta \log \pi_\theta(a_t|s_t)|_{\theta_k} A^{\pi_{\theta_k}}(s_t,a_t) \right]$$

Slide credit: Joshua Achiam

Natural policy gradients

So we have this nice optimization problem: $\pi_{k+1} = \arg\max_{\pi'} \mathcal{L}_{\pi_k}(\pi')$

$$\text{s.t. } \bar{D}_{KL}(\pi' || \pi_k) \leq \delta$$

but how do we solve it? Solution: Taylor expansion!

$$\mathcal{L}_{\theta_k}(\theta) \approx \mathcal{L}_{\theta_k}(\theta_k) + g^T(\theta - \theta_k) \qquad g \doteq \nabla_\theta \mathcal{L}_{\theta_k}(\theta)\,|_{\theta_k}$$

$$\bar{D}_{KL}(\theta||\theta_k) \approx \frac{1}{2}(\theta - \theta_k)^T H(\theta - \theta_k) \qquad H \doteq \nabla^2_\theta \bar{D}_{KL}(\theta||\theta_k)\,|_{\theta_k}$$

Note: zeroth and first-order terms for $\bar{D}_{KL}$ are zero at θ_k .

$$\theta_{k+1} = \arg\max_{\theta} g^T(\theta - \theta_k)$$

$$\text{s.t. } \frac{1}{2}(\theta - \theta_k)^T H(\theta - \theta_k) \leq \delta$$

Solution to approximate problem: $\theta_{k+1} = \theta_k + \sqrt{\frac{2\delta}{g^T H^{-1} g}} H^{-1} g$

Slide credit: Joshua Achiam

Natural policy gradients

- Recall that $\nabla_\theta \mathcal{L}_{\theta_k}(\theta)|_{\theta_k}$ is equal to the policy gradient—so this update gives a policy gradient algorithm where we pre-multiply by H^{-1}.

- The KL-divergence Hessian H is equal to a special matrix called the **Fisher information matrix**, which comes up in a few other places:

$$H = \mathop{\mathrm{E}}_{s,a\sim\theta^k}\left[\nabla_\theta \log \pi_\theta(a|s)|_{\theta_k} \; \nabla_\theta \log \pi_\theta(a|s)|_{\theta_k}^T\right]$$

- The NPG direction $H^{-1}g$ is **covariant**—that is, **it points in the same direction regardless of the parametrization used to compute it**.

Slide credit: Joshua Achiam

Natural policy gradients

Algorithm 1 Natural Policy Gradient

Input: initial policy parameters θ_0
for $k = 0, 1, 2, \ldots$ **do**
Collect set of trajectories $\mathcal{D}_k$ on policy $\pi_k = \pi(\theta_k)$
Estimate advantages $\hat{A}_t^{\pi_k}$ using any advantage estimation algorithm
Form sample estimates for

- policy gradient $\hat{g}_k$ (using advantage estimates)
- and KL-divergence Hessian / Fisher Information Matrix $\hat{H}_k$

Compute Natural Policy Gradient update:

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

end for

Slide credit: Joshua Achiam

Outline: Policy-based RL Methods

- Introduction
- REINFORCE
- Actor-Critic methods
  - Advantage Actor Critic (A2C & A3C)
- Advanced policy gradient methods
  - Natural policy gradient (NPG)
  - Proximal policy optimization (PPO)

Proximal Policy Optimization (PPO)

Proximal Policy Optimization (PPO) is a family of methods that approximately enforce the KL constraint **without computing natural gradients**. Two variants:

- Adaptive KL penalty:
  - Policy update solves the unconstrained optimization problem:

$$\theta_{k+1} = \arg\max_{\theta} \mathcal{L}_{\theta_k}(\theta) - \beta_k \bar{D}_{KL}(\theta||\theta_k)$$

  - Penalty coefficient β_k changes between iterations to approximately enforce KL-divergence constraint.

- Clipped objective: let $r_t(\theta) = \pi_\theta(a_t|s_t)/\pi_{\theta_k}(a_t|s_t)$. Then
  - New objective function:

$$\mathcal{L}^{CLIP}_{\theta_k}(\theta) = \mathop{\mathrm{E}}_{\tau\sim\pi_k}\left[\sum_{t=0}^{T}\left[\min(r_t(\theta)\hat{A}^{\pi_k}_t, \text{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}^{\pi_k}_t)\right]\right]$$

    where ϵ is a hyperparameter

  - Policy update is $\theta_{k+1} = \arg\max_\theta \mathcal{L}^{CLIP}_{\theta_k}(\theta)$

Slide credit: Joshua Achiam

Proximal Policy Optimization with adaptive KL Penalty

Algorithm 4 PPO with Adaptive KL Penalty

Input: initial policy parameters θ_0, initial KL penalty β_0, target KL-divergence δ
for $k = 0, 1, 2, \ldots$ **do**
  Collect set of partial trajectories $\mathcal{D}_k$ on policy $\pi_k = \pi(\theta_k)$
  Estimate advantages $\hat{A}_t^{\pi_k}$ using any advantage estimation algorithm
  Compute policy update

$$\theta_{k+1} = \arg\max_{\theta} \mathcal{L}_{\theta_k}(\theta) - \beta_k \bar{D}_{KL}(\theta || \theta_k)$$

  by taking K steps of minibatch SGD (via Adam)
  if $\bar{D}_{KL}(\theta_{k+1} || \theta_k) \geq 1.5\delta$ **then**
    $\beta_{k+1} = 2\beta_k$
  else if $\bar{D}_{KL}(\theta_{k+1} || \theta_k) \leq \delta / 1.5$ **then**
    $\beta_{k+1} = \beta_k / 2$
  end if
end for

- Initial KL penalty not that important—it adapts quickly
- Some iterations may violate KL constraint, but most do not

Slide credit: Joshua Achiam

Proximal Policy Optimization with clipped Objective

Algorithm 5 PPO with Clipped Objective

Input: initial policy parameters θ_0, clipping threshold ϵ
for $k = 0, 1, 2, \ldots$ **do**
  Collect set of partial trajectories $\mathcal{D}_k$ on policy $\pi_k = \pi(\theta_k)$
  Estimate advantages $\hat{A}_t^{\pi_k}$ using any advantage estimation algorithm
  Compute policy update

$$\theta_{k+1} = \arg\max_{\theta} \mathcal{L}_{\theta_k}^{CLIP}(\theta)$$

  by taking K steps of minibatch SGD (via Adam), where

$$\mathcal{L}_{\theta_k}^{CLIP}(\theta) = \mathop{\mathrm{E}}_{\tau \sim \pi_k} \left[\sum_{t=0}^{T} \left[\min(r_t(\theta)\hat{A}_t^{\pi_k}, \text{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_t^{\pi_k}) \right] \right]$$

end for

- Clipping prevents policy from having incentive to go far away from θ_{k+1}
- **Clipping seems to work at least as well as PPO with KL penalty, but is simpler to implement**

Slide credit: Joshua Achiam

Proximal Policy Optimization with clipped Objective

But how does clipping keep policy close? **By making objective as pessimistic as possible about performance far away from θ_k:**

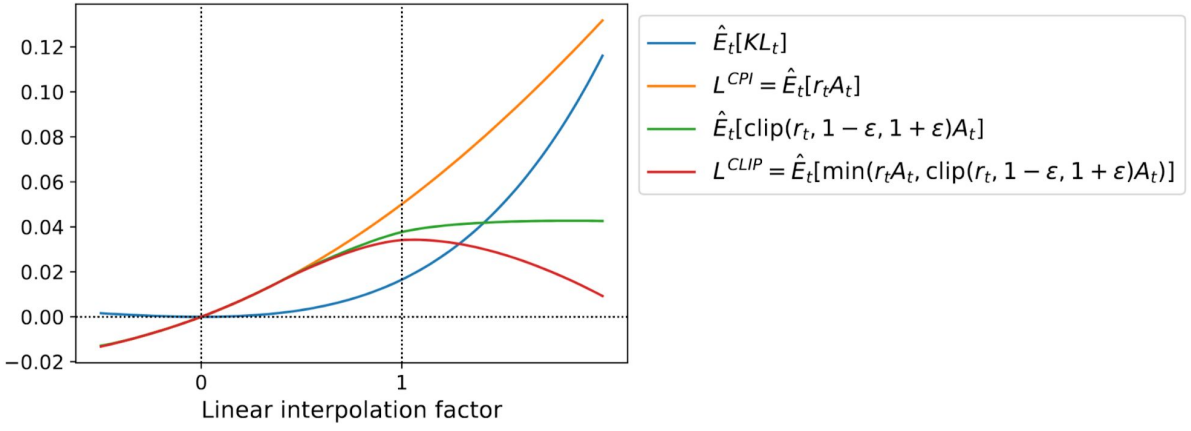


Figure: Various objectives as a function of interpolation factor α between θ_{k+1} and θ_k after one update of PPO-Clip [9]

Slide credit: Joshua Achiam

Summary: NPG → PPO

- Natural policy gradient (NPG)
  - Policy gradient + natural gradient

$$\pi_{k+1} = \arg\max_{\pi'} \mathcal{L}_{\pi_k}(\pi') \quad \text{s.t. } \bar{D}_{KL}(\pi' || \pi_k) \leq \delta$$

$$\Rightarrow \quad \theta_{k+1} = \theta_k + \sqrt{\frac{2\delta}{\hat{g}_k^T \hat{H}_k^{-1} \hat{g}_k}} \hat{H}_k^{-1} \hat{g}_k$$

- Proximal policy optimization (PPO)
  - Penalty version (instead of constrained optimization)
  - Clipping without explicit KL calculation

Empirical performance of PPO

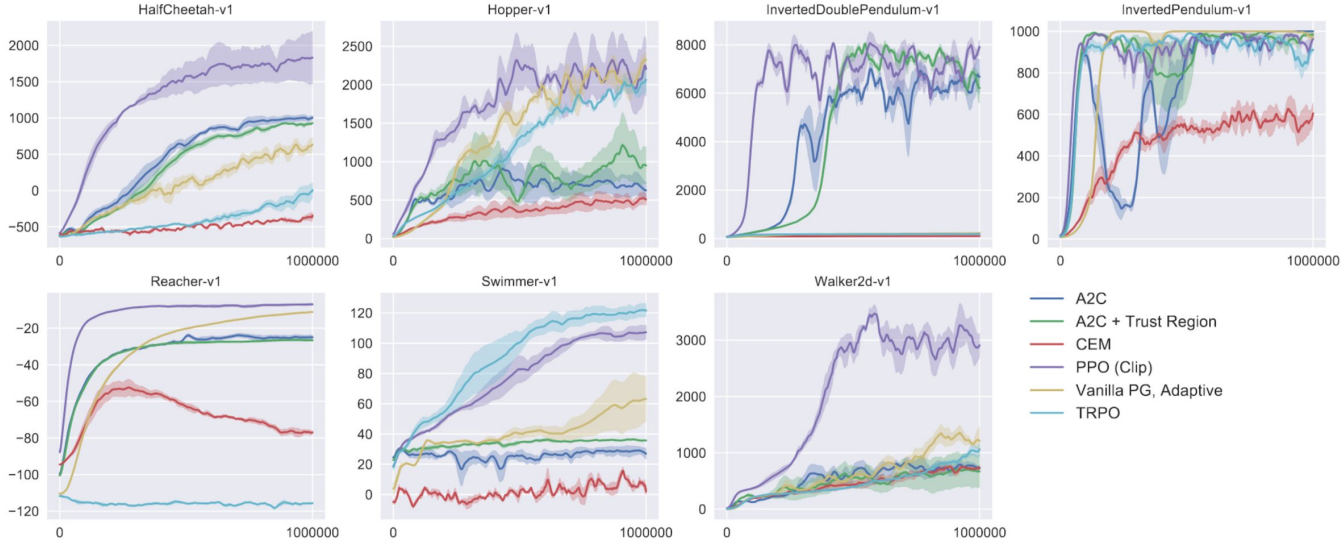


Figure: Performance comparison between PPO with clipped objective and various other deep RL methods on a slate of MuJoCo tasks. [10]

Slide credit: Joshua Achiam

Learning continuous control with PPO

https://youtu.be/Tg0Dyu3iQek

https://youtu.be/irkXnpZP89s`

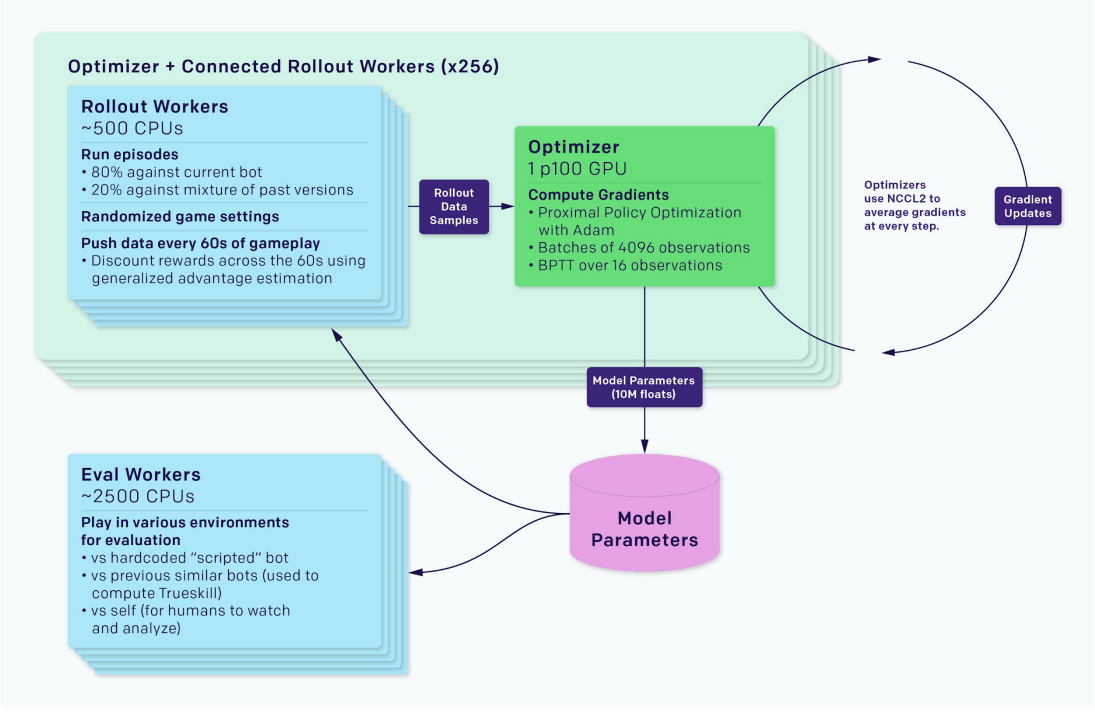
Webpage: https://learningai.io/projects/2017/07/28/ai-gym-workout.html

SOTA application of PPO: OpenAI Five

- **Self-play**:
  - Plays 180 years worth of games against itself every day.
- **Scaling up**:
  - Trains using a scaled-up version of PPO running on 256 GPUs and 128,000 CPU cores.
- **LSTM policy for partial observability and long-term dependencies**:
  - Using a separate LSTM for each hero and no human data, it learns recognizable strategies.
  - Dota games run at 30 frames per second for an average of 45 minutes, resulting in 80,000 ticks per game. OpenAI Five observes every fourth frame, yielding 20,000 moves.

https://openai.com/blog/openai-five/

SOTA application of PPO: OpenAI Five



https://openai.com/blog/openai-five/

Quiz

https://forms.gle/LrYce8k5GExVAaaA8