Reinforcement Learning

Naeemullah Khan
naeemullah.khan@kaust.edu.sa

KAUST Academy
King Abdullah University of Science and Technology

Recap

- It can be hard to learn exact Q-value for every (state, action) pair for high dimensional states and actions
- But, we can just learn a policy that maximizes the reward
- We can use gradient ascent on policy parameters
- But this can suffer from high variance. Various strategies to tackle this.
- Actor-Critic methods combine Policy Gradients and Q-learning by training both an actor (the policy) and a critic (the Q-network)
- The actor decides which action to take, and the critic tells the actor how good its action was and how it should adjust

Types of Policies

- Policy π determines how the agent chooses actions
- Deterministic Policy:

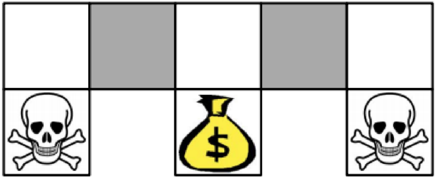
$$\pi(s) = a$$

- Stochastic Policy:

$$\pi(a|s) = Pr(a_t = a|s_t = s)$$

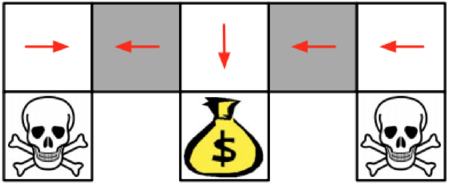
- So far have focused on deterministic policies or ϵ-greedy policies
- ϵ-greedy policies are also near deterministic as we decrease the value of epsilon with training
- Is deterministic policy optimal?

Example: Aliased Grid world



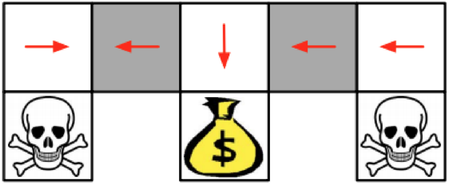
- Consider a Grid World as in the image
- The agent can move in four direction (N, E, W, S) if valid
- The agent **cannot** differentiate the grey states

Example: Aliased Grid world



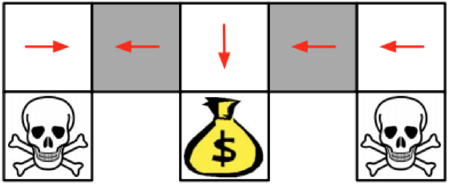
- Under aliasing, an optimal deterministic policy will either
  - move W in both grey states (shown by red arrows)
  - move E in both grey states

Example: Aliased Grid world



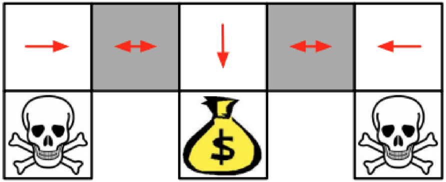
- Under aliasing, an optimal deterministic policy will either
  - move W in both grey states (shown by red arrows)
  - move E in both grey states
- Either way, it can get stuck and never reach the money

Example: Aliased Grid world



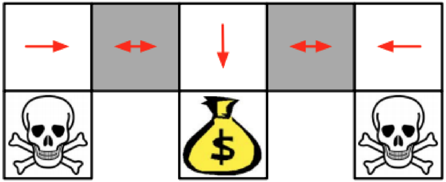
- Under aliasing, an optimal deterministic policy will either
  - move W in both grey states (shown by red arrows)
  - move E in both grey states
- Either way, it can get stuck and never reach the money
- Similarly, Value-based RL learns a near-deterministic policy (e.g., ϵ-greedy)
- As a result, it will traverse the corridor for a long time depending on the value of Epsilon

Example: Aliased Grid world



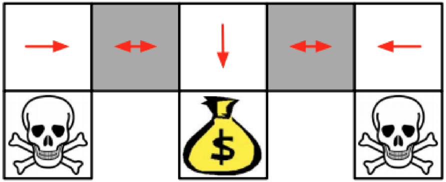
- An optimal stochastic policy will randomly move E or W in grey states
  - π_θ(wall to N and S, move E) $= 0.5$
  - π_θ(wall to N and S, move W) $= 0.5$

Example: Aliased Grid world



- An optimal stochastic policy will randomly move E or W in grey states
  - $\pi_\theta(\text{wall to N and S, move E}) = 0.5$
  - $\pi_\theta(\text{wall to N and S, move W}) = 0.5$
- It will reach the goal state in a few steps with high probability

Example: Aliased Grid world



- An optimal stochastic policy will randomly move E or W in grey states
  - $\pi_\theta(\text{wall to N and S, move E}) = 0.5$
  - $\pi_\theta(\text{wall to N and S, move W}) = 0.5$
- It will reach the goal state in a few steps with high probability
- Policy-based RL can learn the optimal stochastic policy

Limitations of policy gradients

- Sample efficiency is poor
  - We throw out each batch of data immediately after just one gradient step
  - Why? PG is an on-policy expectation.

Limitations of policy gradients

- Sample efficiency is poor
  - We throw out each batch of data immediately after just one gradient step
  - Why? PG is an on-policy expectation.
  - Recycling old data to estimate policy gradients is hard
  - Potential Solution: Use trajectories from other policies with importance sampling.

Limitations of policy gradients

- Sample efficiency is poor
  - We throw out each batch of data immediately after just one gradient step
  - Why? PG is an on-policy expectation.
  - Recycling old data to estimate policy gradients is hard
  - Potential Solution: Use trajectories from other policies with importance sampling.
- Distance in parameter space $\neq$ distance in policy space!

Limitations of policy gradients

- Sample efficiency is poor
  - We throw out each batch of data immediately after just one gradient step
  - Why? PG is an on-policy expectation.
  - Recycling old data to estimate policy gradients is hard
  - Potential Solution: Use trajectories from other policies with importance sampling.
- Distance in parameter space $\neq$ distance in policy space!
  - What is policy space? For tabular case, set of matrices

$$\Pi = \left\{\pi : \pi \in \mathbb{R}^{|S| \times |A|}, \pi_{sa} \geq 0\right\}$$

Limitations of policy gradients

- Sample efficiency is poor
  - We throw out each batch of data immediately after just one gradient step
  - Why? PG is an on-policy expectation.
  - Recycling old data to estimate policy gradients is hard
  - Potential Solution: Use trajectories from other policies with importance sampling.
- Distance in parameter space $\neq$ distance in policy space!
  - What is policy space? For tabular case, set of matrices

$$\Pi = \left\{ \pi : \pi \in \mathbb{R}^{|S| \times |A|}, \pi_{sa} \geq 0 \right\}$$

  - Policy gradients take steps in parameter space
  - Step size is hard to get right as a result

Importance Sampling

- Importance sampling is a technique for estimating expectations using samples drawn from a different distribution.

$$
\begin{aligned}
E_{x\sim P}[f(x)] &= \int P(x)f(x)dx \\
&= \int P(x)\frac{Q(x)}{Q(x)}f(x)dx \\
&= \int Q(x)\frac{P(x)}{Q(x)}f(x)dx \\
&= E_{x\sim Q}\left[\frac{P(x)}{Q(x)}f(x)\right]
\end{aligned}
$$

Importance Sampling

$$\therefore \quad E_{x\sim P}[f(x)] = E_{x\sim Q}\left[\frac{P(x)}{Q(x)}f(x)\right] \approx \frac{1}{|D|}\sum_{x\in D}\frac{P(x)}{Q(x)}f(x)$$

- The ratio P(x)/Q(x) is the importance sampling weight for x

$$\therefore \quad E_{x\sim P}[f(x)] = E_{x\sim Q}\left[\frac{P(x)}{Q(x)}f(x)\right] \approx \frac{1}{|D|}\sum_{x\in D}\frac{P(x)}{Q(x)}f(x)$$

- The ratio P(x)/Q(x) is the importance sampling weight for x
- What is the variance of an importance sampling estimator?

$$\begin{aligned} var(\hat{\mu}_Q) &= \frac{1}{N} var\left(\frac{P(x)}{Q(x)}f(x)\right) \\ &= \frac{1}{N}\left(E_{x\sim Q}\left[\left(\frac{P(x)}{Q(x)}f(x)\right)^2\right] - E_{x\sim Q}\left[\frac{P(x)}{Q(x)}f(x)\right]^2\right) \\ &= \frac{1}{N}\left(E_{x\sim P}\left[\frac{P(x)}{Q(x)}f(x)^2\right] - E_{x\sim P}\left[f(x)\right]^2\right) \end{aligned}$$

Importance Sampling

$$\therefore \quad E_{x\sim P}[f(x)] = E_{x\sim Q}\left[\frac{P(x)}{Q(x)}f(x)\right] \approx \frac{1}{|D|}\sum_{x\in D}\frac{P(x)}{Q(x)}f(x)$$

- The ratio P(x)/Q(x) is the importance sampling weight for x
- What is the variance of an importance sampling estimator?

$$\begin{aligned} var(\hat{\mu}_Q) &= \frac{1}{N} var\left(\frac{P(x)}{Q(x)}f(x)\right) \\ &= \frac{1}{N}\left(E_{x\sim Q}\left[\left(\frac{P(x)}{Q(x)}f(x)\right)^2\right] - E_{x\sim Q}\left[\frac{P(x)}{Q(x)}f(x)\right]^2\right) \\ &= \frac{1}{N}\left(E_{x\sim P}\left[\frac{P(x)}{Q(x)}f(x)^2\right] - E_{x\sim P}\left[f(x)\right]^2\right) \end{aligned}$$

- The term in red is problematic - if $\frac{P(x)}{Q(x)}$ is large in the wrong places, the variance of the estimator explodes.

- Now, let's put this in policy gradient. $\pi_{\theta'}$ represents new policy.

$$\nabla_{\theta'} \mathcal{J}(\theta') = E_{\tau \sim \pi_{\theta'}} \left[\sum_{t \geq 0} \gamma^t \nabla_{\theta'} \log \pi_{\theta'}(a_t | s_t) A^{\pi_{\theta'}}(s_t, a_t) \right]$$

$$= E_{\tau \sim \pi_\theta} \left[\sum_{t \geq 0} \frac{P(\tau_t | \pi_{\theta'})}{P(\tau_t | \pi_\theta)} \gamma^t \nabla_{\theta'} \log \pi_{\theta'}(a_t | s_t) A^{\pi_{\theta'}}(s_t, a_t) \right]$$

- Now, let's put this in policy gradient. $\pi_{\theta'}$ represents new policy.

$$\nabla_{\theta'}\mathcal{J}(\theta') = E_{\tau\sim\pi_{\theta'}}\left[\sum_{t\geq 0}\gamma^t \nabla_{\theta'}\log\,\pi_{\theta'}(a_t|s_t)A^{\pi_{\theta'}}(s_t,a_t)\right]$$

$$= E_{\tau\sim\pi_{\theta}}\left[\sum_{t\geq 0}\frac{P(\tau_t|\pi_{\theta'})}{P(\tau_t|\pi_{\theta})}\gamma^t \nabla_{\theta'}\log\,\pi_{\theta'}(a_t|s_t)A^{\pi_{\theta'}}(s_t,a_t)\right]$$

$$\frac{P(\tau_t|\pi_{\theta'})}{P(\tau_t|\pi_{\theta})} = \frac{\mu(s_0)\prod_{t'=0}^{t}P(s_{t'+1}|s_{t'},a_{t'})\pi_{\theta}(s_{t'},a_{t'})}{\mu(s_0)\prod_{t'=0}^{t}P(s_{t'+1}|s_{t'},a_{t'})\pi_{\theta'}(s_{t'},a_{t'})} = \prod_{t'=0}^{t}\frac{\pi_{\theta'}(s_{t'},a_{t'})}{\pi_{\theta}(s_{t'},a_{t'})}$$

Importance Sampling for Policy Gradients

- Now, let's put this in policy gradient. $\pi_{\theta'}$ represents new policy.

$$\nabla_{\theta'} \mathcal{J}(\theta') = E_{\tau \sim \pi_{\theta'}} \left[\sum_{t \geq 0} \gamma^t \nabla_{\theta'} \log \pi_{\theta'}(a_t|s_t) A^{\pi_{\theta'}}(s_t, a_t) \right]$$

$$= E_{\tau \sim \pi_\theta} \left[\sum_{t \geq 0} \frac{P(\tau_t|\pi_{\theta'})}{P(\tau_t|\pi_\theta)} \gamma^t \nabla_{\theta'} \log \pi_{\theta'}(a_t|s_t) A^{\pi_{\theta'}}(s_t, a_t) \right]$$

$$\frac{P(\tau_t|\pi_{\theta'})}{P(\tau_t|\pi_\theta)} = \frac{\mu(s_0) \prod_{t'=0}^{t} P(s_{t'+1}|s_{t'}, a_{t'}) \pi_\theta(s_{t'}, a_{t'})}{\mu(s_0) \prod_{t'=0}^{t} P(s_{t'+1}|s_{t'}, a_{t'}) \pi_{\theta'}(s_{t'}, a_{t'})} = \prod_{t'=0}^{t} \frac{\pi_{\theta'}(s_{t'}, a_{t'})}{\pi_\theta(s_{t'}, a_{t'})}$$

- Looks useful - what's the issue?

Importance Sampling for Policy Gradients

- Now, let's put this in policy gradient. $\pi_{\theta'}$ represents new policy.

$$\nabla_{\theta'} \mathcal{J}(\theta') = E_{\tau \sim \pi_{\theta'}} \left[\sum_{t \geq 0} \gamma^t \nabla_{\theta'} \log \pi_{\theta'}(a_t|s_t) A^{\pi_{\theta'}}(s_t, a_t) \right]$$

$$= E_{\tau \sim \pi_\theta} \left[\sum_{t \geq 0} \frac{P(\tau_t|\pi_{\theta'})}{P(\tau_t|\pi_\theta)} \gamma^t \nabla_{\theta'} \log \pi_{\theta'}(a_t|s_t) A^{\pi_{\theta'}}(s_t, a_t) \right]$$

$$\frac{P(\tau_t|\pi_{\theta'})}{P(\tau_t|\pi_\theta)} = \frac{\mu(s_0) \prod_{t'=0}^{t} P(s_{t'+1}|s_{t'}, a_{t'}) \pi_\theta(s_{t'}, a_{t'})}{\mu(s_0) \prod_{t'=0}^{t} P(s_{t'+1}|s_{t'}, a_{t'}) \pi_{\theta'}(s_{t'}, a_{t'})} = \prod_{t'=0}^{t} \frac{\pi_{\theta'}(s_{t'}, a_{t'})}{\pi_\theta(s_{t'}, a_{t'})}$$

- Looks useful - what's the issue?
- Exploding or vanishing importance sampling weights. Even for policies only slightly different from each other, many small differences multiply to become a big difference.

Importance Sampling for Policy Gradients

- Solution?

- Solution?
- Stay close to the previous policy!
- We can use KL divergence for that.
- What is KL-divergence between policies?

$$D_{KL}(\pi'||\pi)[s] = \sum_{a \in A} \pi'(a|s) \log \frac{\pi'(a|s)}{\pi(a|s)}$$

Importance Sampling for Policy Gradients

- Solution?
- Stay close to the previous policy!
- We can use KL divergence for that.
- What is KL-divergence between policies?

$$D_{KL}(\pi'||\pi)[s] = \sum_{a \in A} \pi'(a|s) \log \frac{\pi'(a|s)}{\pi(a|s)}$$

- Now, we have

$$\nabla_{\theta'} \mathcal{J}(\theta') \;\text{ s.t. }\; D_{KL}(\pi'||\pi) \leq \epsilon$$

Relative Policy Performance Identity

- But, recall that for $\nabla_{\theta'} \mathcal{J}(\theta')$ we will still have to compute $\log \pi_{\theta'}(a_t|s_t) A^{\pi_{\theta'}}(s_t, a_t)$ based on current policy.
- This is not desirable.

- But, recall that for $\nabla_{\theta'} \mathcal{J}(\theta')$ we will still have to compute $\log \pi_{\theta'}(a_t|s_t) A^{\pi_{\theta'}}(s_t, a_t)$ based on current policy.
- This is not desirable.
- So, we make use of Relative Policy Performance Identity. This states that for two policies, $\pi_{\theta'}$ and π_θ

$$\mathcal{J}(\pi_{\theta'}) - \mathcal{J}(\pi_\theta) = E_{\tau \sim \pi_{\theta'}} \left[\sum_{t=0}^{T} \gamma^t A^{\pi_\theta}(s_t, a_t) \right]$$

- But, recall that for $\nabla_{\theta'} \mathcal{J}(\theta')$ we will still have to compute $\log \pi_{\theta'}(a_t|s_t) A^{\pi_{\theta'}}(s_t, a_t)$ based on current policy.
- This is not desirable.
- So, we make use of Relative Policy Performance Identity. This states that for two policies, $\pi_{\theta'}$ and π_θ

$$\mathcal{J}(\pi_{\theta'}) - \mathcal{J}(\pi_\theta) = E_{\tau \sim \pi_{\theta'}} \left[\sum_{t=0}^{T} \gamma^t A^{\pi_\theta}(s_t, a_t) \right]$$

- Using importance sampling, we get

$$\mathcal{J}(\pi_{\theta'}) - \mathcal{J}(\pi_\theta) = E_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{T} \frac{\pi_{\theta'}(s_t, a_t)}{\pi_\theta(s_t, a_t)} \gamma^t A^{\pi_\theta}(s_t, a_t) \right]$$

Relative Policy Performance Identity

- Can we use this for policy improvement?

- Can we use this for policy improvement?
- Recall that our objective is to have policy with maximum return.
- Basically, we want to

$$\max_{\theta'} \mathcal{J}(\pi_{\theta'})$$

Relative Policy Performance Identity

- Can we use this for policy improvement?
- Recall that our objective is to have policy with maximum return.
- Basically, we want to

$$\max_{\theta'} \mathcal{J}(\pi_{\theta'})$$

- But, this is essentially the same as

$$\max_{\theta'} (\mathcal{J}(\pi_{\theta'}) - \mathcal{J}(\pi_{\theta}))$$

Relative Policy Performance Identity

- Can we use this for policy improvement?
- Recall that our objective is to have policy with maximum return.
- Basically, we want to

$$\max_{\theta'} \mathcal{J}(\pi_{\theta'})$$

- But, this is essentially the same as

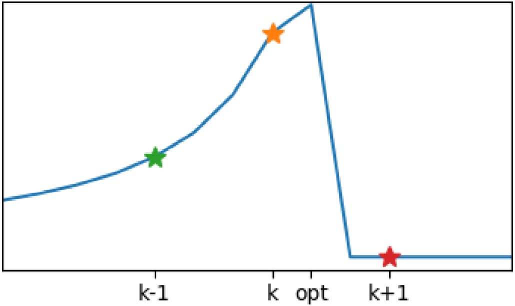
$$\max_{\theta'} (\mathcal{J}(\pi_{\theta'}) - \mathcal{J}(\pi_{\theta}))$$

- Therefore, we can use this as our loss function

$$\mathcal{L}_{\theta'}(\pi_{\theta'}) = \mathcal{J}(\pi_{\theta'}) - \mathcal{J}(\pi_{\theta})$$

Choosing a Step Size for Policy Gradients

- Policy gradient algorithms are stochastic gradient ascent
- If the step is too large, performance collapse is possible
- If the step is too small, progress is unacceptably slow
- "Right" step size changes based on θ



Choosing a Step Size for Policy Gradients

- But, the problem is more than step size
- Distance in parameter space $\neq$ distance in policy space!
- Small changes in the policy parameters can unexpectedly lead to big changes in the policy.

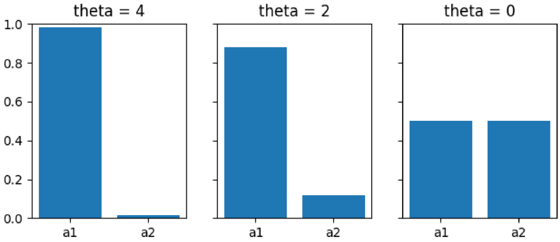
Choosing a Step Size for Policy Gradients

- But, the problem is more than step size
- Distance in parameter space $\neq$ distance in policy space!
- Small changes in the policy parameters can unexpectedly lead to big changes in the policy.
- Consider a family of policies with parametrization:

$$\pi_\theta(a) = \begin{cases} \sigma(\theta) & a = 1 \\ 1 - \sigma(\theta) & a = 2 \end{cases}$$

- But, the problem is more than step size
- Distance in parameter space $\neq$ distance in policy space!
- Small changes in the policy parameters can unexpectedly lead to big changes in the policy.
- Consider a family of policies with parametrization:

$$\pi_\theta(a) = \begin{cases} \sigma(\theta) & a = 1 \\ 1 - \sigma(\theta) & a = 2 \end{cases}$$



Trust Region Policy Optimization (TRPO)

- TRPO updates policies by taking the largest step possible to improve performance, while satisfying a special constraint on how close the new and old policies are allowed to be.
- The constraint is expressed in terms of KL-Divergence
- This is different from normal policy gradient, which keeps new and old policies close in parameter space
- TRPO nicely avoids this kind of collapse, and tends to quickly and monotonically improve performance
- TRPO uses conjugate gradients for computing the hessian matrix for KL divergence derivative

[0] https://spinningup.openai.com/en/latest/algorithms/trpo.html

KAUST ACADEMY

- TRPO uses backtracking line search with exponential decay (decay coeff $\alpha \in (0,1)$, budget L) to make appropriate step sizes

---

Algorithm 2 Line Search for TRPO

Compute proposed policy step $\Delta_k = \sqrt{\frac{2\delta}{\hat{g}_k^T \hat{H}_k^{-1} \hat{g}_k}} \hat{H}_k^{-1} \hat{g}_k$
for $j = 0, 1, 2, ..., L$ **do**
  Compute proposed update $\theta = \theta_k + \alpha^j \Delta_k$
  if $\mathcal{L}_{\theta_k}(\theta) \geq 0$ and $\bar{D}_{KL}(\theta || \theta_k) \leq \delta$ **then**
    accept the update and set $\theta_{k+1} = \theta_k + \alpha^j \Delta_k$
    break
  end if
end for

Algorithm 3 Trust Region Policy Optimization

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-vector product function $f(v) = \hat{H}_k v$

Use CG with n_{cg} iterations to obtain $x_k \approx \hat{H}_k^{-1} \hat{g}_k$

Estimate proposed step $\Delta_k \approx \sqrt{\frac{2\delta}{x_k^T \hat{H}_k x_k}} x_k$

Perform backtracking line search with exponential decay to obtain final update

$$\theta_{k+1} = \theta_k + \alpha^j \Delta_k$$

end for

Proximal Policy Optimization (PPO)

- PPO is motivated by the same question as TRPO: how can we take the biggest possible improvement step on a policy using the data we currently have, without stepping so far that we accidentally cause performance collapse?
- Where TRPO tries to solve this problem with a complex second-order method, PPO is a family of first-order methods that use a few other tricks to keep new policies close to old.
- It approximately enforce KL constraint without computing natural gradients.
- PPO methods are significantly simpler to implement, and empirically seem to perform at least as well as TRPO.
- There are two primary variants of PPO: PPO-Penalty and PPO-Clip.

[0] https://spinningup.openai.com/en/latest/algorithms/ppo.html

Proximal Policy Optimization (PPO)

- Adaptive KL Penalty
  - Policy update solves unconstrained optimization problem

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

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

- Adaptive KL Penalty
  - Policy update solves unconstrained optimization problem

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

  - Penalty coefficient β_k changes between iterations to approximately enforce KL-divergence constraint
- Clipped Objective
  - New objective function: let $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)}$. Then,

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

  - ϵ is a hyperparameter (e.g., $\epsilon = 0.2$)
  - Policy update is

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

Proximal Policy Optimization (PPO)

Algorithm 4 PPO with Adaptive KL Penalty

Input: initial policy parameters θ_0, initial KL penalty β_0, target KL-divergence δ
for $k = 0, 1, 2, \dots$ **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

- PPO clip is more widely used as it seems to work at least as well as PPO with KL penalty, but is simpler to implement

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

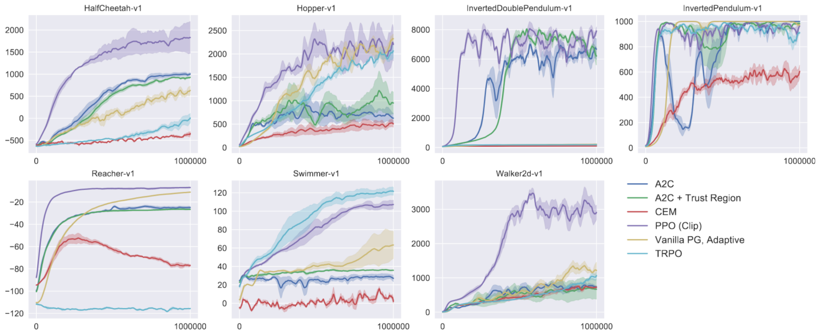
$$\theta_{k+1} = \arg\max_{\theta} \mathcal{L}_{\theta_k}^{CLIP}(\theta)$$

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

Empircal Performance of PPO



[0] Schulman, Wolski, Dhariwal, Radford, Klimov, 2017

Summary

- Sometimes, it is better to learn a stochastic policy than a deterministic policy
- Policy gradient methods suffer from poor sample efficiency
- Importance sampling can help alleviate this problem
- However, we need to make sure that current policy is not far from policy with which we have collected trajectories
- Small changes in the policy parameters can unexpectedly lead to big changes in the policy.
- TRPO uses importance sampling to take multiple gradient steps and constrains the optimization objective in the policy space
- PPO does the same but approximately enforces KL constraint without computing natural gradients.

References

These slides have been adapted from

- Chelsea Finn & Karol Hausman, Stanford CS224R: Deep Reinforcement Learning
- Emma Brunskill, Stanford CS234: Reinforcement Learning
- Joshua Achiam, Berkeley CS294 Deep Reinforcement Learning
- Sergey Levine, Berkeley CS285 Deep Reinforcement Learning