High-Dimensional Continuous Control Using Generalized Advantage Estimation.

참고 페이지

Abstract

- 정책 기반 강화 학습의 두 가지 문제점
  - 샘플이 많이 필요하다.
  - the non-stationarity of the incoming data에도 불구 하고 안정적이고 지속적인 개선이 어렵다.
- 제안
  - $TD(\lambda)$ 와 유사한 advantage function의 exponentially-weighted estimator를 사용하여 bias를 희생하더라도 policy gradient estimate의 variance를 줄인다.
  - 정책과 가치 함수에 trust region optimization을 사용 한다.

Introduction

A key source of difficulty in RL

- **credit assignment** (**distal reward problem**)
  - the long time delay between actions and their positive or negative effect on rewards

parameterized stochastic policy

- unbiased
- variance
  - since the effect of an action is confounded with the effects of past and future actions. (?)

soft-actor critic

- bias 이지만 낮은 variance
- 높은 variance를 갖는 다면 더 많은 샘플을 사용하면 되지만 bias는 샘플이 많더라도 수렴하지 못하거나 local optimum 조차도 아닌 a poor solution에 수렴할 수 있다.

이 논문의 contribution

- tolerable 한 bias 수준으로 variance 를 줄이는 policy gradient estimator를 제안 한다. (GAE)
- value function에도 trust region optimization method를 사용한다.
- 위의 두가지를 합쳐서 control task의 neural network policies를 효과적으로 empirically 학습할 수 있는 알고리즘을 찾는다. 이러한 결과는 high-dimensional continuous control에 RL을 사용함으로서 state of art로 확장되었다.

Preliminaries

- initial state s_0 는 distribution ρ_0 에서 샘플링 된 것이다.
- 하나의 trajectory $(s_0, a_0, s_1, a_1, \ldots)$ 는 terminal (absorbing) state에 도달 할 때 까지 policy $a_t \sim \pi(a_t|s_t)$ 에 따른 sampling action과 dynamics $s_{t+1} \sim P(s_{t+1}|s_t, a_t)$ 에 따른 sampling state에 의해 generate 된다.
- reward $r_t = r(s_t, a_t, s_{t+1})$ 는 각 타임 스템프 마다 받는다.
- 목적은 expected total reward $\sum_{t=0}^{\infty} r_t$ 를 최대하는 것인데, 이 것은 모든 policy에 대해 finite 하다고 가정한다.

- 여기서 중요한 점은 γ 를 discount parameter가 아닌 **bias-variance trade off**로 사용한다는 것이다.

- policy gradient 방법은 gradient $g := \bigtriangledown_\theta \mathbb{E}[\sum_{t=0}^{\infty} r_t]$ 를 반복적으로 추정함으로써 expected total reward를 최대화 하는 것이다.
- policy gradient는 여러가지 다른 표현이 있다.

$$g = \mathbb{E}[\sum_{t=0}^{\infty} \Psi \bigtriangledown_\theta log\pi_\theta(a_t|s_t)] \qquad (1)$$

  여기서 Ψ_t는 다음 중 하나 일 수 있다.

1. $\sum_{t=0}^{\infty} r_t$: total reward of the trajectory.
2. $\sum_{t'=t}^{\infty} r'_t$: reward following action a_t
3. $\sum_{t'=t}^{\infty} r'_t - b(s_t)$: baseline version of 2
4. $Q^\pi(s_t, a_t)$: state-action value function
5. $A^\pi(s_t, a_t)$: advantage function
6. $r_t + V^\pi(s_{t+1}) - V^\pi(s_t)$: TD residual

여기서 ,

$V^\pi(s_t) := \mathbb{E}_{s_{t+1}:\infty, a_t:\infty}[\sum_{l=0}^{\infty} r_{t+l}] \qquad Q^\pi(s_t, a_t) := \mathbb{E}_{s_{t+1}:\infty, a_t:\infty}[\sum_{l=0}^{\infty} r_{t+l}] \qquad (2)$

$A^\pi(s_t, a_t) := Q^\pi(s_t, a_t) - V^\pi(s_t) \qquad (3)$ 입니다.

여기서 Ψ_t 에 $A^\pi(s_t, a_t)$를 사용할 때 가장 낮은 variance 가 나온다. (비록, 정확히 알지 못하고 추정해야 하지만.)

parameter γ

- 비록 bias가 있겠지만 delayed effects에 대응하는 reward를 down-weighting 함으로써 variance를 줄인다. (discounted factor와 형태는 일치하지만, 의도는 다르다.)

discounted value function은 다음과 같다.

$V^{\pi}(s_t) := \mathbb{E}_{s_{t+1}:\infty,a_t:\infty}[\sum_{l=0}^{\infty}\gamma^l r_{t+l}] \qquad Q^{\pi}(s_t,a_t) := \mathbb{E}_{s_{t+1}:\infty,a_t:\infty}[\sum_{l=0}^{\infty}\gamma^l r_{t+l}] \qquad (4)$

$A^{\pi}(s_t,a_t) := Q^{\pi}(s_t,a_t) - V^{\pi}(s_t) \qquad (5)$ 이다.

policy gradient에 대한 discounted approximation은 다음과 같다.

$$g^{\gamma} := \mathbb{E}_{\mathrm{s}_0:\infty,\mathrm{a}_0:\infty}[\sum_{t=0}^{\infty} A^{\pi,\gamma}(s_t,a_t)\ \bigtriangledown_{\theta}\ log\pi_{\theta}(a_t|s_t)] \qquad (6)$$

How to obtain biased (but not too biased) estimators for $A^{\pi,\gamma}$

먼저 $\gamma-\text{just if}$ 개념을 소개한다. g^{γ} 를 추정하기 위해 식(6)에서 $A^{\pi,\gamma}$ 대신에 이것을 사용할 때 bias가 생기지 않는 etimator다.

Definition 1 The estimator $\hat{A}_t$는 $\gamma-\text{just if}$

$$\mathbb{E}_{\mathrm{s}_0:\infty,\mathrm{a}_0:\infty}[\sum_{t=0}^{\infty}\hat{A}(s_{0:\infty},a_{0:\infty})\ \bigtriangledown_{\theta}\ log\pi_{\theta}(a_t|s_t)] = \mathbb{E}_{\mathrm{s}_0:\infty,\mathrm{a}_0:\infty}[\sum_{t=0}^{\infty}A^{\pi,\gamma}(s_{0:\infty},a_{0:\infty})\ \bigtriangledown_{\theta}\ log\pi_{\theta}(a_t|s_t)] \qquad (7)$$

이것은 모든 t에 대해서 $\hat{A}_t$가 γ-just하다면

$$\mathbb{E}_{\mathrm{s}_0:\infty,\mathrm{a}_0:\infty}[\sum_{t=0}^{\infty}A^{\pi,\gamma}(s_{0:\infty},a_{0:\infty})\ \bigtriangledown_{\theta}\ log\pi_{\theta}(a_t|s_t)] = g^{\gamma} \qquad (8)$$

가 된다.

$\hat{A}_t$ 가 γ-just가 되기 위한 한가지 충분 조건은 $\hat{A}_t$는 Q_t와 b_t로 나눌 수 있어야 한다는 것이다. 여기서 Q_t는 trajectory variables에 종속적이지만 γ-discounted Q-function에 대해 unbiased estimator를 제공하고 b_t는 a_t 이전에 샘플링 된 state와 action의 arbitrary function이다.

Proposition 1 $\hat{A}_t$ 를 모든 (s_t,a_t)에 대해 $\hat{A}_t(s_{0:\infty},a_{0:\infty}) = Q_t(s_{t:\infty},a_{t:\infty}) - b_t(s_{0:t},a_{0:t-1})$ 로 표현될 수 있다면 $\mathbb{E}_{\mathrm{s}_{t+1}:\infty,\mathrm{a}_{t+1}:\infty|\mathrm{s}_t,\mathrm{a}_t}[Q_t(s_{t:\infty},a_{t:\infty})] = Q^{\pi,\gamma}(s_t,a_t)$.

이 증명은 Appendix B에 있다.

γ-just advantage estimators for $\hat{A}_t$는 다음과 같이 쓸 수 있다.

- $\sum_{l=0}^{\infty} \gamma^l r_{t+l}$
- $A^{\pi,\gamma}(s_t, a_t)$
- $Q^{\pi,\gamma}(s_t, a_t)$
- $r_t + \gamma V^{\pi,\gamma}(s_{t+1}) - V^{\pi,\gamma}(s_t)$

Advantage Function Estimation

이 번 섹션에서는 $\hat{g} = \frac{1}{N} \sum_{n=1}^{N} \sum_{t=0}^{\infty} \hat{A}_t^n \bigtriangledown_\theta log \pi_\theta(a_t^n | s_t^n)$ 과 같은 형태의 policy gradient estimator를 구하기 위해 사용되는 discounted advantage function $A^{\pi,\gamma}(s_t, a_t)$ 를 정확하게 추정하는 $\hat{A}_t$ 를 구하려고 한다.

V 를 approximate value function 이라 하자. 그리고 $\delta_t^V = r_t + \gamma V(s_{t+1}) - V(s_t)$ 즉 discount γ 를 갖는 V 의 TD residual 이다. δ_t^V 는 action a_t 의 advantage 추정값으로 생각할 수 있다.

사실, 우리가 정확한 가치 함수 $V = V^{\pi,\gamma}$ 를 알고 있다면 이것은 γ-just advantage estimator 고 실제로 $A^{\pi,\gamma}$ 의 unbiased estimator 는 다음과 같다.

$$\mathbb{E}_{s_{t+1}}[\delta_t^{V^{\pi,\gamma}}] = \mathbb{E}_{s_{t+1}}[r_t + \gamma V^{\pi,\gamma} - V^{\pi,\gamma}(s_t)] = \mathbb{E}_{s_{t+1}}[Q^{\pi,\gamma}(s_t, a_t) - V^{\pi,\gamma}(s_t)] = A^{\pi,\gamma}(s_t, a_t) \qquad (10)$$

이 estimator는 $V = V^{\pi,\gamma}$ 에 대해 only $\gamma - \text{just}$ 다. 그렇지 않으면 biased policy gradient estimate를 하게 된다.

다음으로 이 δ 의 k 의 sum으로 생각을 해보자 이것을 $\hat{A}_t^{(k)}$ 라 하자.

$$\hat{A}_t^{(1)} := \delta_t^V = -V(s_t) + r_t + \gamma V(s_{t+1}) \qquad (11)$$

$$\hat{A}_t^{(2)} := \delta_t^V + \gamma \delta_{t+1}^V = -V(s_t) + r_t + \gamma r_{t+1} + \gamma^2 V(s_{t+2}) \qquad (12)$$

$$\hat{A}_t^{(3)} := \delta_t^V + \gamma \delta_{t+1}^V + \gamma^2 \delta_{t+2}^V = -V(s_t) + r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 V(s_{t+3}) \qquad (13)$$

$$\hat{A}_t^{(k)} := \sum_{l=0}^{k-1} \gamma^l \delta_{t+l}^V = -V(s_t) + r_t + \gamma r_{t+1} + \cdots + \gamma^{k-1} r_{t+k-1} + \gamma^k V(s_{t+k}) \qquad (14)$$

$k \to \infty$ 할 때 bias는 점점 더 작아진다. 왜냐하면 term $\gamma^k V(s_{t+k})$ 은 더 discounted되고 term $-V(s_t)$ 은 bias에 영향을 주지 않는다.

$$\hat{A}_t^{(\infty)} := \sum_{l=0}^{\infty} \gamma^l \delta_{t+l}^V = -V(s_t) + \sum_{l=0}^{\infty} \gamma^l r_{t+l} \qquad (15)$$

이것은 단지 경험적 return 에서 vlaue function baselie을 뺀 것이다.

$\mathrm{GAE}(\gamma, \lambda)$는 다음과 같이 k-step estimator의 exponentially-weighted average 로 정의할 수 있다.

$$\begin{aligned}
\hat{A}_t^{\mathrm{GAE}(\gamma,\lambda)} &:= (1-\lambda)(\hat{A}_t^{(1)} + \hat{A}_t^{(2)} + \hat{A}_t^{(3)} + \ldots) \\
&= (1-\lambda)(\delta_t^V + \lambda(\delta_t^V + \gamma\delta_{t+1}^V) + \lambda^2(\delta_t^V + \gamma\delta_{t+1}^V + \gamma^2\delta_{t+2}^V) + \ldots) \\
&= (1-\lambda)(\delta_t^V(1 + \lambda + \lambda^2 + \ldots) + \gamma\delta_{t+1}^V(\lambda + \lambda^2 + \lambda^3 + \ldots) + \gamma^2\delta_{t+2}^V(\lambda^2 + \lambda^3 + \lambda^4 + \ldots) \\
&(1-\lambda)(\delta_t^V(\frac{1}{1-\lambda}) + \gamma\delta_{t+1}^V(\frac{\lambda}{1-\lambda}) + \gamma^2\delta_{t+2}^V(\frac{\lambda^2}{1-\lambda})) \\
&= \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+l}^V \qquad (16)
\end{aligned}$$

위의 수식에서 $\lambda = 0$과 $\lambda = 1$에 대해서 특별한 case 가 존재한다.

$$\mathrm{GAE}(\gamma, 0) : \hat{A}_t := \delta_t \quad = r_t + \gamma V(s_{t+1}) - V(s_t) \qquad (17)$$

$$\mathrm{GAE}(\gamma, 1) : \hat{A}_t := \sum_{l=0}^{\infty} \gamma^l \delta_{t+l} \quad = \sum_{l=0}^{\infty} \gamma^l r_{t+l} - V(s_t) \qquad (18)$$

$\mathrm{GAE}(\gamma, 1)$ 는 V 의 정확도에 관계 없이 $\gamma - \mathrm{just}$ 이지만, return의 sum 때문에 high variance이다.
$\mathrm{GAE}(\gamma, 0)$ 는 $V = V^{\pi,\gamma}$ 에 대한 $\gamma - \mathrm{just}$ 이다. bias는 있지만 매우 낮은 variance를 갖는다. GAE에서 λ는 $0 < \lambda < 1$ 에서 bias와 variance 사이를 조절하는 파라메터 이다.

두 가지 별개의 parameter γ and λ 를 가진 advantage estimator는 bias-variance tradeoff에 도움을 준다. 그러나 이 두 가지 parameter는 각각 다른 목적을 가지고 작동한다.

- γ 는 가장 중요하게 value function $V^{\pi,\gamma}$ 의 scale을 결정한다. 또한 γ는 λ에 의존하지 않는다. $\gamma < 1$ 로 정하는 것은 value function의 정확도에 상관없이 policy gradient estimate에서 bias하다.
- 반면에, $\lambda < 1$는 유일하게 value function이 부정확할 때 bias한다. 그리고 경험상, λ의 best value는 γ의 best value보다 훨씬 더 낮다. 왜냐하면 λ가 정확한 value function에 대해 보다 훨씬 덜 bias하기 때문이다.

GAE를 사용할 때 식(6)으로 부터 discounted policy gradient인 g^γ 의 biased estimator는 다음과 같다.

$$g^\gamma \approx \mathbb{E}[\sum_{t=0}^{\infty} \bigtriangledown_\theta log\pi_\theta(a_t|s_t)\hat{A}^{\mathrm{GAE}(\gamma,\lambda)}] = \mathbb{E}[\sum_{t=0}^{\infty} \bigtriangledown_\theta log\pi_\theta(a_t|s_t)(\gamma\lambda)^l\delta_{t+1}^V] \qquad (19)$$

여기서 $\lambda = 1$ 일 때 같다.

Interpretation as Reward Shaping

Value Function Estimation

value function을 추정하는 방법에는 여러가지가 있지만 있다. nonlinear function approximator를 사용할 때 nonlinear regression problem을 푸는 가장 간단한 형태는 다음과 같다.

$$\text{minimize}_{\phi} \sum_{n=1}^{N} \left\| V_{\phi}(s_n) - \hat{V}_n \right\|^2 \qquad (28)$$

여기서 $\hat{V}_t = \sum_{l=0}^{\infty} \gamma^l r_{t+l}$ 은 reward의 discounted sum이다. 그리고 n 은 trajectory의 batch에서 모든 timestamp를 index한 것이다. 이것은 Monte Carlo or TD(1)이라 한다.

이 논문에서는 batch optimization procedure의 각 iteration에서 value function을 최적화 하기 위해 **trust region method**를 사용했다. (가장 최근 데이터에 overfit 되는 것을 막아준다.)

먼저, $\sigma^2 = \frac{1}{N} \sum_{n=1}^{N} \left\| V_{\phi_{\text{old}}}(s_n) - \hat{V}_n \right\|^2$ 을 계산하는데 ϕ_{old} 는 opimization 하기전의 parameter vector 다. 그리고 나서 다음 constrained optimization problem을 푼다.

$$\text{minimize}_{\phi} \sum_{n=1}^{N} \left\| V_{\phi}(s_n) - \hat{V}_n \right\|^2$$

$$\text{subject to} \quad \frac{1}{N} \sum_{n=1}^{N} \frac{\left\| V_{\phi}(s_n) - \hat{V}_{\phi_{\text{old}}} \right\|^2}{2\sigma^2} \leq \epsilon \qquad (29)$$

이 제약식은 이전 value function과 새로운 value function 사이의 평균 KL divergence를 ϵ 보다 작게 제한하는 것과 동일한데, 여기서 value function은 평균 $V_{\phi}(s)$ 과 variace σ^2 를 갖는 coditional Gaussian distribution으로 parameter화 되었다.

여기서, conjugate gradient algorithm을 사용해서 trust region problem에 대한 근사해를 계산했다. 특히, 다음과 같은 quadratic program을 풀게된다.

$$\text{minimize}_{\phi} g^T (\phi - \phi_{\text{old}})$$

$$\text{subject to} \quad \frac{1}{N} \sum_{n=1}^{N} (\phi - \phi_{\text{old}})^T H (\phi - \phi_{\text{old}}) \leq \epsilon \qquad (30)$$

여기서 g는 objective의 gradient이고 $H = \frac{1}{N} \sum_n j_n j_n^T, \text{where } j_n = \bigtriangledown_{\phi} V_{\phi}(s_n)$.

H는 objective의 Hessinan에 대해 "Gauss-Newton" approximation 이다. value function을 coditional probability로 해석한다면 fisher information matrix가 된다.

계산할 때 과정은 TRPO와 동일하다.

Experiments

이 논문에서는 다음과 같은 질문을 조사하기 위해 실험 환경을 설계했다.

1. GAE를 사용하여 episodic total reward를 최적화 할 때 $\lambda \in [0, 1]$ 과 $\gamma \in [0, 1]$ 이 변할때 어떤 경험적 효과가 있는지?
2. TRPO를 policy와 value function optimization에 적용했을 때 GAE를 cotrol 문제를 해결하기 위해 large neural network policy를 최적화 하는데 사용할 수 있는지?

policy optimization algorithm

GAE는 여러가지 다양한 policy gradient method와 함께 사용할 수 있다. 이 논문에서는 TRPO를 사용해서 policy update를 했다. TRPO는 각 iteration마다 다음 constrained optimization을 근사적으로 해결함으로써 policy를 업데이트 한다.

$$\text{minimize}_\theta \; L_{\theta_{\text{old}}}(\theta)$$

$$\text{subject to } \bar{D}_{\text{KL}}^{\theta_{\text{old}}}(\pi_{\theta_{\text{old}}}, \pi_\theta) \leq \epsilon$$

$$\text{where } L_{\theta_{\text{old}}}(\theta) = \frac{1}{N}\sum_{n=1}^{N} \frac{\pi_\theta(s_n|s_n)}{\pi_{\theta_{\text{old}}}(a_n|s_n)}\hat{A}_n$$

$$\bar{D}_{\text{KL}}^{\theta_{\text{old}}}(\pi_{\theta_{\text{old}}}, \pi_\theta) = \frac{1}{N}\sum_{n=1}^{N} D_{\text{KL}}(\pi_{\theta_{\text{old}}}(\cdot|s_n)||\pi_\theta(\cdot|s_n)) \qquad (31)$$

TRPO 논문에서 설명한 것 처럼 objective를 linearizing 하고 constraint를 quadraticizing 함으로써 이 문제를 근사적으로 해결하는데 이것은 $\theta - \theta_{\text{old}} \propto -F^{-}1g$ 방향으로 한 step 이동하는 것이다.

최종 알고리즘은 다음과 같다.

Initialize policy parameter θ_0 and value function parameter ϕ_0.
for $i = 0, 1, 2, \ldots$ **do**
  Simulate current policy π_{θ_i} until N timesteps are obtained.
  Compute δ_t^V at all timesteps $t \in \{1, 2, \ldots, N\}$, using $V = V_{\phi_i}$.
  Compute $\hat{A}_t = \sum_{l=0}^{\infty}(\gamma\lambda)^l\delta_{t+l}^V$ at all timesteps.
  Compute θ_{i+1} with TRPO update, Equation (31).
  Compute ϕ_{i+1} with Equation (30).
end for

여기서 advantage estimation으로 $V_{\phi_{i+1}}$ 가 아니라 V_{ϕ_i} 를 사용하여 $\theta \rightarrow \theta_{i+1}$ 하게 된다. value function을 먼저 업데이트 한다면 추가적인 bias가 생긴다.

극단적으로 생각해서 value function을 overfit한다면 bellman residual $r_t + \gamma V(s_t + 1) - V(s_t)$ 는 모든 타임스템프에서 0가 되고 policy gradient estimate도 zero가 된다.