A Cubic-regularized Policy Newton Algorithm for Reinforcement Learning

Prashanth L.A.[†]
Joint work with Mizhaan Maniyar[†], Akash Mondal[♯] and Shalabh Bhatnagar[♯]

† IIT Madras

♯ Indian Institute of Science

(To appear in AISTATS 2024)

1-slide summary

Cubic-regularized policy Newton (CR-PN)

- *Gradient and Hessian estimates* + bias/variance bounds
- *SOSP convergence* $O(n^{(2-\alpha)/2})$, $\alpha \in (0,1)$ is Hölder exponent of w
- REINFORCE is known to converge to an FOSP

Approximate CR-PN

- Use gradient descent to solve a sub-problem in CR-PN + *Hessian-vector products enough*
- *SOSP convergence* $O(d\sqrt{n}\ \text{polylog}(n))$.

Simulation experiments

- *Cart-pole:* CR-PN performs better than REINFORCE with *linear features*
- *Mujoco:* same conclusion with *neural net features*

Background

RL 101: finite-horizon MDP, policy gradient framework

Stochastic non-convex optimization: first and second-order stationary points, stochastic gradient and Newton algorithms

Background

RL 101: finite-horizon MDP, policy gradient framework

Stochastic non-convex optimization: first and second-order stationary points, stochastic gradient and Newton algorithms

---

Our work

Cubic-regularized policy Newton (CR-PN): gradient and Hessian estimation, estimation error bounds, algorithm

Theoretical results: SOSP convergence, sample complexity

Approximate CR-PN: Computational efficiency, SOSP guarantee

Simulation experiments: Cart pole, Mujoco

Introduction

Markov Decision Processes (MDPs)

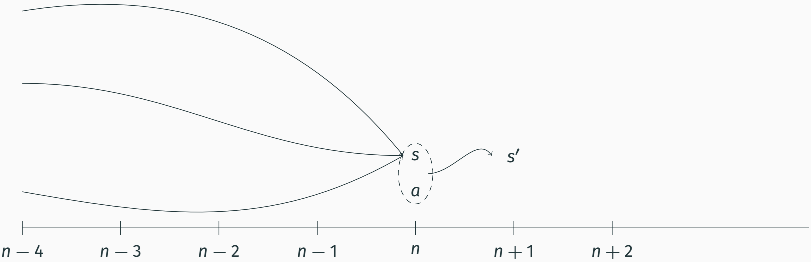
Basic Elements: Set of States $\mathcal{S}$, Set of Actions $\mathcal{A}$, Costs $c(x, a)$

Transition Probabilities:

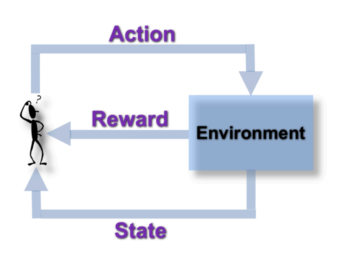
$$P(s'|s, a)$$

Markov Property $\forall i_0, i_1, \ldots, s, s', b_0, b_1 \ldots$,a,

$$P(s_{n+1} = s' \mid s_n = s, a_n = a, \ldots, s_0 = i_0, a_0 = b_0) = P(s'|s, a)$$



Reinforcement Learning (RL)



- RL: A class of learning problems in which an agent interacts with a dynamic, stochastic, and incompletely known environment
- Goal: Learn an action-selection strategy, or ***policy***, to optimize some performance measure
- Interaction: Modeled as a Markov Decision Process (MDP)

Finite-horizon MDP

Policy: $\pi(a|s) \rightarrow$ probability of choosing action a in state s

Trajectory: $\tau := (s_0, a_0, \ldots, a_{H-1}, s_H)$ has probability:

$$\boxed{p(\tau;\pi) := \left(\prod_{h=0}^{H-1} P(s_{h+1}|s_h, a_h)\pi(a_h|s_h)\right)\rho(s_0)}$$

Finite-horizon MDP

Policy: $\pi(a|s) \rightarrow$ probability of choosing action a in state s

Trajectory: $\tau := (s_0, a_0, \dots, a_{H-1}, s_H)$ has probability:

$$\boxed{p(\tau;\pi) := \left(\prod_{h=0}^{H-1} P(s_{h+1}|s_h, a_h)\pi(a_h|s_h)\right)\rho(s_0)}$$

$$\pi^* = \arg\min_{\pi} \left\{ J(\pi) = \mathbb{E}_{\tau\sim p(\tau;\pi)}\left[\sum_{h=0}^{H-1} \gamma^{h-1}\, c(s_h, a_h) \;\middle|\; \pi\right]\right\}$$

Finite-horizon MDP

Policy: $\pi(a|s) \rightarrow$ probability of choosing action a in state s

Trajectory: $\tau := (s_0, a_0, \ldots, a_{H-1}, s_H)$ has probability:

$$\boxed{p(\tau;\pi) := \left(\prod_{h=0}^{H-1} P(s_{h+1}|s_h, a_h)\pi(a_h|s_h)\right)\rho(s_0)}$$

Optimal policy

$$\pi^* = \arg\min_{\pi} \left\{ J(\pi) = \mathbb{E}_{\tau\sim p(\tau;\pi)}\left[\sum_{h=0}^{H-1} \gamma^{h-1}\, c(s_h, a_h) \;\middle|\; \pi\right]\right\}$$

Value function

Cost

Policy

Policy gradient framework

Policy Gradient Setting

A class of parameterized stochastic (randomized) policies

$$\boxed{\left\{\pi(\cdot|s;\theta), s \in \mathcal{S}, \theta \in \mathbb{R}^d\right\}}$$

$\pi(|s_h;\theta)$: probability distribution (parameterized by θ) over action space rather than unique action for each state

Example: Boltzmann policies

$$\boxed{\pi(a|s;\theta) = \frac{\exp\left(\psi(s,a)^T\theta\right)}{\sum_{b\in\mathcal{A}} \exp\left(\psi(s,b)^T\theta\right)}, \ \forall s \in \mathcal{S} \ , \ \forall a \in \mathcal{A}}$$

Lot of interest in analyzing policy gradient algorithms, cf. (Agarwal et al. 2020; Sutton et al. 1999; Papini et al. 2018; Vijayan and Prashanth 2021; Zhang et al. 2020; Kumar, Koppel, and Ribeiro 2023)

Policy gradient

Setting: (θ policy parameter)

- Aim: minimize $J(\theta)$

$$\boxed{\min_\theta J(\theta) = \mathbb{E}_{\tau\sim p(\tau;\pi)}\left[\sum_{h=0}^{H-1}\gamma^{h-1}c(s_h, a_h)\right]}$$

- PG update: $\boxed{\theta_{k+1} = \theta_k - \eta\widehat{\nabla}J(\theta_k)}$

Policy gradient

Setting: (θ policy parameter)

- Aim: minimize $J(\theta)$

$$\boxed{\min_\theta J(\theta) = \mathbb{E}_{\tau\sim p(\tau;\pi)}\left[\sum_{h=0}^{H-1} \gamma^{h-1} c(s_h, a_h)\right]}$$

- PG update: $\boxed{\theta_{k+1} = \theta_k - \eta \widehat{\nabla} J(\theta_k)}$

Policy gradient estimation:

Q1) How to form $\widehat{\nabla} J(\theta_n)$ in a finite horizon MDP?

Q2) What is the bias and variance of such an estimate?

Policy gradient: variants

Policy gradient:

$$\boxed{\theta_{k+1} = \theta_k - \eta \widehat{\nabla} J(\theta_k)}$$

Pro: Easy to implement, Con: Slow convergence near optima

Policy Newton *(using gradient and Hessian estimates)*:

$$\boxed{\theta_{k+1} = \theta_k - \eta \left(\widehat{\nabla}^2 J(\theta_k) \right)^{-1} \widehat{\nabla} J(\theta_k)}$$

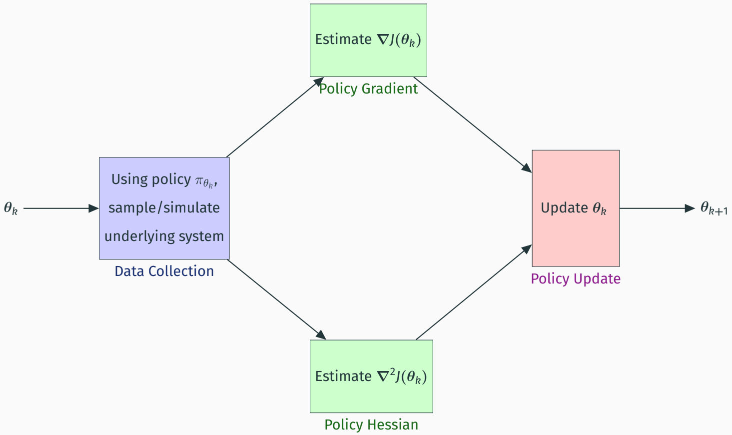
Pro: Faster convergence near optima, Con: Computational burden

---

Best of both: Perform a steepest-descent step for large gradients; else take a step in a negative-curvature direction for $\nabla^2 f$.

Cubic-regularized policy Newton $\rightarrow$ does both to escape saddle points (more details later)

Policy Newton algorithm



Likelihood ratio method

(Stochastic) Gradient Estimation

Huge literature; here will focus on likelihood ratio (LR) method, aka score function (SF) method (other: perturbation analysis)

general setting: parameter appears in input distribution, e.g., distribution over actions in randomized policy

Simple single r.v. X example: (p_θ p.m.f. of X)

$$\boxed{\mathbb{E}[X] = \sum_x x\mathbb{P}_\theta(X = x) = \sum_x x p_\theta(x),}$$

Differentiating w.r.t. θ (assuming exchange),

$$\boxed{\frac{d\mathbb{E}[X]}{d\theta} = \sum_x x \frac{d\mathbb{P}_\theta(X = x)}{d\theta} = \sum_x x \frac{d \ln p_\theta(x)}{d\theta} p_\theta(x) = \mathbb{E}\left[X \frac{d \ln p_\theta(X)}{d\theta}\right],}$$

so LR derivative estimator

$$X \frac{d \ln p_\theta(X)}{d\theta}.$$

(Stochastic) Gradient Estimation: Markov Chains

Markov chain $\{X_n\}$ with a single recurrent state 0, transient states $1, \ldots, r$, and transition probability matrix $\boxed{P(\theta) := [p_{ij}(\theta)]_{i,j=0}^{r}}$

$\tau \to$ first passage time to state 0.

Unbiased single-run sample path LR gradient estimator:

$$\boxed{\widehat{\nabla} h(\theta) = \hat{h}(X) \nabla \ln p_{X_0 X_1 \cdots X_\tau}(\theta) = \hat{h}(X) \sum_{m=0}^{\tau-1} \frac{\nabla p_{X_m X_{m+1}}(\theta)}{p_{X_m X_{m+1}}(\theta)}.}$$

Likelihood ratios for gradient estimation

Markov chain. $\{X_n\}$

States. 0 recurrent, other states transient

Transition probability matrix. $P(\theta) := [[p_{X_i X_j}(\theta)]]_{i,j=0}^{r}$

Performance measure. $F(\theta) = \mathbb{E}[f(X)]$

Likelihood ratios for gradient estimation

Markov chain. $\{X_n\}$

States. 0 recurrent, other states transient

Transition probability matrix. $P(\theta) := [[p_{X_i X_j}(\theta)]]_{i,j=0}^{r}$

Performance measure. $F(\theta) = \mathbb{E}[f(X)]$

Simulate (using $P(\theta)$) and obtain $X := (X_0, \ldots, X_{\tau-1})^T$

$$\nabla_\theta F(\theta) = \mathbb{E}\left[f(X) \sum_{m=0}^{\tau-1} \frac{\nabla_\theta p_{X_m X_{m+1}}(\theta)}{p_{X_m X_{m+1}}(\theta)} \right]$$

Policy gradient and Hessian theorem

Assumptions

(A1) Bounded costs:

$$c(s,a)| \leq K, \qquad \forall (s,a) \in \mathcal{S} \times \mathcal{A},$$

(A2) Parameterization regularity:

$$\|\nabla \log \pi(a|s;\theta)\| \leq G \quad \text{and} \quad \left\|\nabla^2 \log \pi(a|s;\theta)\right\| \leq L_1, \, \forall \theta$$

(A3) Lipschitz Hessian:

$$\left\|\nabla^2 \log \pi(a|s;\theta_1) - \nabla^2 \log \pi(a|s;\theta_2)\right\| \leq L_2 \left\|\theta_1 - \theta_2\right\|, \, \forall \theta_1, \theta_2$$

Policy gradient and Hessian expressions

Total discounted cost:

$$\Psi_i(\tau) := \sum_{h=i}^{H-1} \gamma^{h-1} c(s_h, a_h) \text{ and } \Phi(\theta;\tau) := \sum_{i=0}^{H-1} \Psi_i(\tau) \log \pi(a_i|s_i;\theta)$$

Policy gradient:

$$\boxed{\nabla J(\theta) = \mathbb{E}_{\tau\sim p(\tau;\theta)} \left(\nabla \Phi(\theta;\tau)\right)}$$

Policy Hessian:

$$\boxed{\nabla^2 J(\theta) = \mathbb{E}_{\tau\sim p(\tau;\theta)} \left(\nabla \Phi(\theta;\tau) \nabla^\top \log p(\tau;\theta) + \nabla^2 \Phi(\theta;\tau)\right)}$$

See Shen et al. 2019 for the proof

Policy gradient and Hessian: Smoothness results

Under (A1)-(A3), for any θ_1, θ_2, we have

Lipschitz function: $|J(\theta_1) - J(\theta_2)| \leq M_{\mathcal{H}} \|\theta_1 - \theta_2\|$

Lipschitz gradient: $\boxed{\|\nabla J(\theta_1) - \nabla J(\theta_2)\| \leq G_{\mathcal{H}} \|\theta_1 - \theta_2\|}$

Lipschitz Hessian: $\boxed{\left\|\nabla^2 J(\theta_1) - \nabla^2 J(\theta_2)\right\| \leq L_{\mathcal{H}} \|\theta_1 - \theta_2\|}$

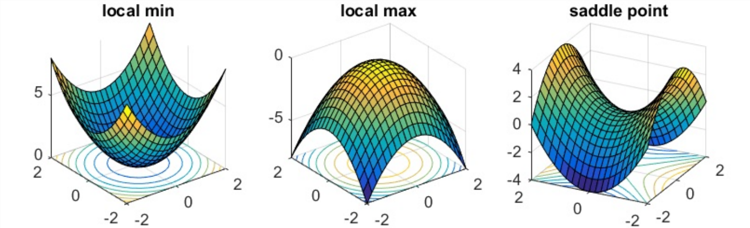
Last condition implies: $J(\theta + \Delta) \leq J(\theta) + \nabla J(\theta)^{\top} \Delta + \frac{1}{2} \Delta^{\top} \nabla^2 J(\theta) \Delta + \frac{1}{6} L_{\mathcal{H}} \|\Delta\|^3$

See Maniyar et al. 2023 for the proof


I run my own startup and drive a Porsche, but I feel like something's missing ...
Whoa, what an amazing day! The bear only devoured one of my arms!
Perspectives on happiness - now and then

Stationary points: First, second, . . .

Stationary points: First and second



Type	Condition
FOSP θ	$\nabla J(\theta) = 0$
ϵ-FOSP θ	$\|\nabla J(\theta)\| \leq \epsilon$
SOSP θ	$\nabla J(\theta) = 0$ and $\nabla^2 J(\theta) \succeq 0$
SOSP θ	$\|\nabla J(\theta)\| \leq \epsilon$ and $\nabla^2 J(\theta) \succeq -\sqrt{\rho\epsilon}\,\mathbb{I}$

FOSP: First-order stationary point, SOSP: Second-order stationary point; Image src: offconvex.org

More on SOSPs

For a non-convex J, finding an FOSP ain't enough.

e.g. $\boxed{J(\theta_1, \theta_2) = \theta_1^2 - \theta_2^2}$ $\nabla J(0, 0) = 0$. Is it a local minimum?

More on SOSPs

For a non-convex J, finding an FOSP ain't enough.

e.g. $\boxed{J(\theta_1, \theta_2) = \theta_1^2 - \theta_2^2}$ $\nabla J(0,0) = 0$. Is it a local minimum?

$J(0, \epsilon) < J(0,0)$ Compute $\nabla^2 J(\theta)$

$\boxed{\nabla J(\theta) = 0 \text{ and } \nabla^2 J(\theta) \succ 0 \Rightarrow \theta \text{ is a local minimum}}$

If $\nabla J(\theta) = 0$ and $\nabla^2 J(\theta) = 0$, then try a TOSP, and so on.

Bad news: It is NP-hard to find a local minimum

Not so bad if saddle points are strict, as polynomial time algorithms can find a local minimum.

Strict saddle: $\boxed{\nabla J(\theta) = 0 \text{ and } \lambda_{\min}(\nabla^2 J(\theta)) < 0}$

Approximate SOSP

Definition
Algorithm outputs a random θ_R. Then, for some $\rho > 0$, θ_R is an ϵ-SOSP if

$$\boxed{\max\left\{\sqrt{\mathbb{E}\|\nabla J(\theta_R)\|}, -\frac{1}{\sqrt{\rho}}\mathbb{E}\lambda_{\min}\left(\nabla^2 J(\theta_R)\right)\right\} \le \sqrt{\epsilon}}$$

w.h.p. variant: For any $\delta \in (0,1)$, w.p. $(1-\delta)$, we have

$$\boxed{\max\left\{\sqrt{\|\nabla J(\theta_R)\|}, \frac{-1}{\sqrt{\rho}}\lambda_{\min}\left(\nabla^2 J(\theta_R)\right)\right\} \le \sqrt{\epsilon}}$$

Summarizing..

- $\boxed{\text{FOSPs}}$ aren't necessarily local optima owing to non-convexity

Summarizing..

- FOSPs aren't necessarily local optima owing to non-convexity
- SOSPs: local minima if saddle points are strict;

Summarizing..

- $\boxed{\text{FOSPs}}$ aren't necessarily local optima owing to non-convexity
- $\boxed{\text{SOSPs}}$: local minima if saddle points are strict;
- $\boxed{\text{Policy gradient RL}}$: Find an ϵ-SOSP using ideas from stochastic non-convex optmization

Getting to SOSPs: Typical Approaches

- Perturbed gradient descent (Ge et al. 2015; Jin et al. 2021): Add isotropic noise in the update decrement to escape saddle points $\boxed{\theta_{k+1} = \theta_k - \eta \nabla J(\theta_k) + \eta_k, \quad \eta_k \sim \mathcal{N}(0, \sigma^2 \mathbb{I})}$
- Cubic-regularized Newton (Nesterov and Polyak 2006): Use second-order information

$\boxed{\text{Policy gradient + perturbed GD:}}$ not an easy combination for getting to SOSPs (Why?)

Cubic-regularized policy Newton (CR-PN)

Main message #1:

Cubic-regularized policy Newton finds an ϵ-SOSP with a $O(1/\epsilon^{3.5})$ bound on the sample complexity[1]

Algorithm	Sample complexity	ϵ-FOSP	ϵ-SOSP
REINFORCE	$\mathcal{O}\left(\frac{1}{\epsilon^4}\right)$	✓	✗
(Shen et al. 2019)	$\mathcal{O}\left(\frac{1}{\epsilon^3}\right)$	✓	✗
(Yang, Zheng, and Pan 2021)	$\mathcal{O}\left(\frac{1}{\epsilon^{4.5}}\right)$	✓	✓
Our work	$\mathcal{O}\left(\frac{1}{\epsilon^{3.5}}\right)$	✓	✓

[1] Mizhaan Prajit Maniyar, Prashanth L.A., Akash Mondal, Shalabh Bhatnagar, A Cubic-regularized Policy Newton Algorithm for Reinforcement Learning, AISTATS, 2024 (Accepted).

Motivation for cubic-regularization

- The standard Newton step is given by:

$$\boxed{\theta_{k+1} = \theta_k - \nabla^2 J(\theta_k)^{-1} \nabla J(\theta_k)}$$

- This is equivalent to finding a θ that minimizes

$$\boxed{\langle \nabla J(\theta_k), \theta - \theta_k \rangle + \frac{1}{2} \left\langle \nabla^2 J(\theta_k)(\theta - \theta_k), \theta - \theta_k \right\rangle}$$

- The issues that arise which such an update, is that the Hessian can be degenerate or non-negative definite.
- Alternative: Add a cubic term to the quadratic approximation:

$$\boxed{\langle \nabla J(\theta_k), \theta - \theta_k \rangle + \frac{1}{2} \left\langle \nabla^2 J(\theta_k)(\theta - \theta_k), \theta - \theta_k \right\rangle + \frac{\alpha}{6} \|\theta - \theta_k\|^3 .}$$

Recall: Policy gradient and Hessian expressions

Total discounted cost:

$$\Psi_i(\tau) := \sum_{h=i}^{H-1} \gamma^{h-1} c(s_h, a_h) \text{ and } \Phi(\theta; \tau) := \sum_{i=0}^{H-1} \Psi_i(\tau) \log \pi(a_i | s_i; \theta)$$

Policy gradient: $\boxed{\nabla J(\theta) = \mathbb{E}_{\tau \sim p(\tau;\theta)} \left(\nabla \Phi(\theta; \tau) \right)}$

Policy Hessian:

$$\boxed{\nabla^2 J(\theta) = \mathbb{E}_{\tau \sim p(\tau;\theta)} \left(\nabla \Phi(\theta; \tau) \nabla^\top \log p(\tau; \theta) + \nabla^2 \Phi(\theta; \tau) \right)}$$

Cubic-regularized policy Newton

Three-step solution:

Step 1: Obtain multiple trajectories for the MDP using π_{θ_k};

Step 2: Estimate $\nabla J(\theta)$ and $\nabla^2 J(\theta)$ using these trajectories

Step 3: Solve cubic subproblem and then update θ_k

$$\boxed{\theta_k = \operatorname*{arg\,min}_{\theta \in \mathbb{R}^d} \left\{ \tilde{J}^k(\theta) \equiv \tilde{J}(\theta, \theta_{k-1}, \bar{\mathcal{H}}_k, \bar{g}_k, \alpha_k) \right\}, \text{ where}}$$

$$\tilde{J}(\theta, \bar{\theta}, \mathcal{H}, g, \alpha) =$$
$$\left\langle g, \theta - \bar{\theta} \right\rangle + \frac{1}{2} \left\langle \mathcal{H}(\theta - \bar{\theta}), \theta - \bar{\theta} \right\rangle + \frac{\alpha}{6} \left\| \theta - \bar{\theta} \right\|^3 .$$

Estimating the gradient and Hessian

Estimates from a single trajectory τ under policy θ:

$$\boxed{g(\theta;\tau) := \nabla\Phi(\theta;\tau), \mathcal{H}(\theta;\tau) := \nabla\Phi(\theta;\tau)\nabla^\top \log p(\tau;\theta) + \nabla^2\Phi(\theta;\tau)}$$

Sample average approximations:

Gradient estimate with m_k trajectories:

$$\bar{g}_k = \frac{1}{m_k}\sum_{\tau\in\mathcal{T}_m}\sum_{h=0}^{H-1}\Psi_h(\tau)\nabla\log\pi(a_h|s_h;\theta_{k-1})$$

Hessian estimate with m_k trajectories:

$$\bar{\mathcal{H}}_k = \frac{1}{b_k}\sum_{\tau\in\mathcal{T}_b}\left(\sum_{h=0}^{H-1}\Psi_h(\tau)\nabla\log\pi(a_h|s_h;\theta_{k-1})\sum_{h'=0}^{H-1}\nabla^\top\log\pi(a_{h'}|s_{h'};\theta_{k-1})\right)$$
$$+\frac{1}{b_k}\sum_{\tau\in\mathcal{T}_b}\sum_{h=0}^{H-1}\Psi_h(\tau)\nabla^2\log\pi(a_h|s_h;\theta_{k-1})$$

ϵ-SOSP convergence

Main result: Let θ_N be computed by CR-PN Algorithm with the following parameters:

$$\alpha_k = 3L_{\mathcal{H}}, N = \frac{12\sqrt{L_{\mathcal{H}}}\left(J^* - J(\theta_0)\right)}{\epsilon^{\frac{3}{2}}},$$
$$m_k = \frac{25G_g^2}{4\epsilon^2}, b_k = \frac{36\sqrt[3]{30(1+2\log 2d)}d^{\frac{2}{3}}G_{\mathcal{H}}^2}{\epsilon}$$

Let θ_R be picked uniformly at random from $\{\theta_1, \ldots, \theta_N\}$. Then,

$$\boxed{5\sqrt{\epsilon} \geq \max\left\{\sqrt{\mathbb{E}\|\nabla J(\theta_R)\|}, -\frac{5}{6\sqrt{L_{\mathcal{H}}}}\mathbb{E}\lambda_{\min}\left(\nabla^2 J(\theta_R)\right)\right\}}$$

A similar bound holds with high probability.

Remarks

- To find an ϵ-SOSP, # trajectories to compute the gradient and the Hessian are $\boxed{O\left(\frac{1}{\epsilon^{\frac{7}{2}}}\right) \text{ and } O\left(\frac{1}{\epsilon^{\frac{5}{2}}}\right)}$
- Shen et al. 2019 need $O\left(\frac{1}{\epsilon^3}\right)$ # trajectories, but find an FOSP
- Yang, Zheng, and Pan 2021 need $O\left(\frac{1}{\epsilon^{\frac{9}{2}}}\right)$, while Zhang et al. 2020 require $O\left(\frac{1}{\epsilon^9}\right)$

Approximate cubic-regularized policy Newton (ACRPN)

Approximately solving the cubic problem

- Cubic sub-problem in each iteration of CR-PN is

$$\theta_k = \underset{\theta \in \mathbb{R}^d}{\arg\min} \left\{ \tilde{J}^k(\theta) \equiv \tilde{J}(\theta, \theta_{k-1}, \bar{\mathcal{H}}_k, \bar{g}_k, \alpha_k) \right\}$$

- Approximate solution: perform gradient descent for a reasonable # of steps
- Advantage: GD steps are Hessian-free; need Hessian-vector products.
- This makes implementation in libraries like PyTorch or TensorFlow easier

Solving the cubic sub-problem

- The cubic auxilliary function can be re-written as:

$$F^k(\Delta) := \langle \bar{g}_k, \Delta \rangle + \frac{1}{2} \left\langle \Delta, \bar{\mathcal{H}}_k \Delta \right\rangle + \frac{\alpha}{6} \left\| \Delta \right\|^3$$

- and thus, $\boxed{\theta_k = \theta_{k-1} + \underset{\Delta \in \mathbb{R}^d}{\arg\min} F^k(\Delta)}$

  Perform GD in an inner-loop:
  for t = 1, ..., *T*:

$$\Delta_t = \Delta_{t-1} - \eta \left(\bar{g}_k + \bar{\mathcal{H}}_k \Delta_{t-1} + \frac{\alpha}{2} \left\| \Delta_{t-1} \right\| \Delta_{t-1} \right)$$

- where η, T are hyper-parameters to obtain a "good enough" solution for $\underset{\Delta \in \mathbb{R}^d}{\arg\min} F^k(\Delta)$.

Solving the cubic sub-problem

- The cubic auxilliary function can be re-written as:

$$F^k(\Delta) := \langle \bar{g}_k, \Delta \rangle + \frac{1}{2} \left\langle \Delta, \bar{\mathcal{H}}_k \Delta \right\rangle + \frac{\alpha}{6} \|\Delta\|^3$$

- and thus, $\boxed{\theta_k = \theta_{k-1} + \underset{\Delta \in \mathbb{R}^d}{\arg\min} F^k(\Delta)}$

  Perform GD in an inner-loop:

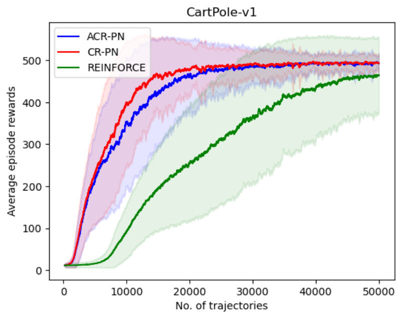
  for t = 1, ..., *T*:

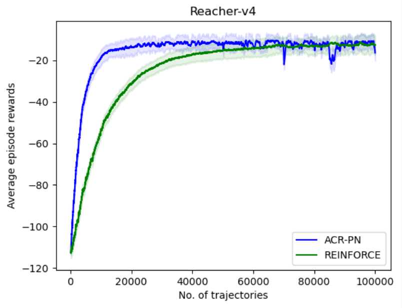
$$\Delta_t = \Delta_{t-1} - \eta \left(\bar{g}_k + \bar{\mathcal{H}}_k \Delta_{t-1} + \frac{\alpha}{2} \|\Delta_{t-1}\| \Delta_{t-1} \right)$$

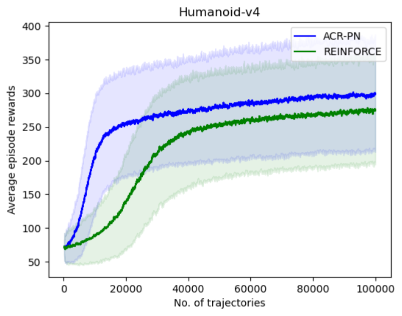
- where η, T are hyper-parameters to obtain a “good enough” solution for $\underset{\Delta \in \mathbb{R}^d}{\arg\min} F^k(\Delta)$.

In a stochastic non-convex opt setting, Carmon and Duchi 2019 suggest a clever GD procedure for solving cubic sub-problem; extended later in Tripuraneni et al. 2018;

Simulation experiments







CR-PN outperforms ACR-PN slightly owing to its higher precision for subproblem solver

ACR-PN can be extended to neural networks as in MuJoCo experiments

Two recent results in risk-sensitive RL

1. Risk Estimation in a Markov Cost Process: Lower and Upper Bounds
Joint work with Gugan Thoppe and Sanjay Bhat

2. Policy Evaluation for Variance in Average Reward Reinforcement Learning
Joint work with Shubhada Agrawal and Siva Theja Maguluri


WALLY, YOUR STATUS REPORT APPEARS TO HAVE BEEN WRITTEN BY A COMMERCIAL-GRADE A.I.
THANK YOU. I APPRECIATE THE COMPLIMENT.
ADD SOME MISTAKES SO IT LOOKS LIKE A HUMAN DID IT.
DOES THE IMAGINARY HUMAN USE SPELL CHECK?

Gugan Thoppe, Prashanth L.A., Sanjay Bhat,
Risk Estimation in a Markov Cost Process,
arxiv preprint 2310.11389

Problem Formulation

- Setup: MCP $M \equiv (\mathcal{S}, P, g, \gamma)$ with the infinite-horizon cumulative discounted cost $\boxed{X_\infty = \sum_{t=0}^{\infty} \gamma^t c(s_t)}$

- Goal: Lower and upper bounds on the samples needed for an ϵ-accurate estimate for VaR, CVaR, and variance of X_∞

- For a random variable X,

$$v_\alpha(X) = \inf\{\xi : \Pr\{X \leq \xi\} \geq \alpha\}$$

$$c_\alpha(X) = \mathbb{E}[X | X \geq v_\alpha(X)]$$

Summary of Key Contributions

Bound type	Risk measure	Sample complexity
Lower bound	Mean, VaR, CVaR, variance	$\Omega\left(\frac{1}{\epsilon^2}\right)$
Upper bound	CVaR	$\widetilde{\mathcal{O}}\left(\frac{1}{\epsilon^2}\right)$
Upper bound	Lipschitz risk measure	$\widetilde{\mathcal{O}}\left(\frac{1}{\epsilon^2}\right)$
Upper bound	Variance	$\widetilde{\mathcal{O}}\left(\frac{1}{\epsilon^2}\right)$

Sample complexity is the # of sample transitions N s.t. $\boxed{\mathbb{E}|\hat{\eta}_n - \eta(D)| < \epsilon}$, where $\hat{\eta}_n \rightarrow$ estimate, $\eta(D) \rightarrow$ risk measure.

Key Proof Ideas - Lower Bounds

- Lower bounds apply to (i) deterministic and (ii) stochastic costs

- For deterministic costs, the hard problem instance involves a 2-state Markov chain with the cost function $2\epsilon \exp(1/\epsilon^2)$

- For stochastic costs, we use a single-state MCP with Gaussian costs. Importantly, the cost mean can be bounded w.r.t. ϵ.

Upper Bounds

- Estimator with truncated trajectories
- Covers variance, CVaR, spectral risk measure, utility-based shortfall risk
- Proof uses concentration bounds for iid case in conjunction with a argument that bounds the error due to truncation

Policy Evaluation for Variance in Average Reward Reinforcement Learning

Shubhada Agrawal, Prashanth L. A. and Siva Theja Maguluri.

Variance in Average-cost MDPs

Average cost

$$J_\mu = \lim_{T\to\infty} \frac{1}{T}\mathbb{E}\left[\sum_{k=0}^{T-1} c(S_k, A_k) | S_0 = s\right]$$

Asymptotic variance

$$\kappa_\mu = \lim_{T\to\infty} \frac{1}{T}\,\mathrm{Var}\left[\sum_{k=0}^{T-1} c(S_k, A_k) \middle| (S_0, A_0) \sim d_\mu\right]$$

Variance in Average-cost MDPs

Average cost

$$J_\mu = \lim_{T\to\infty} \frac{1}{T}\mathbb{E}\left[\sum_{k=0}^{T-1} c(S_k, A_k) | S_0 = s\right]$$

Asymptotic variance

$$\kappa_\mu = \lim_{T\to\infty} \frac{1}{T}\operatorname{Var}\left[\sum_{k=0}^{T-1} c(S_k, A_k)\middle|(S_0, A_0) \sim d_\mu\right]$$

Equivalent expression:

$$\kappa_\mu = \mathbb{E}_{d_\mu}\big[(c(S,A) - J_\mu)^2\big] + 2\lim_{T\to\infty}\sum_{j=1}^{T-1}\mathbb{E}_{d_\mu}\big[(c(S_0,A_0) - J_\mu)\left(c(S_j,A_j) - J_\mu\right)\big]$$

Policy evaluation using TD

Useful expression for designing TD algorithm:

$$\boxed{\kappa_\mu = 2\mathbb{E}_{d_\mu}[(r(S,A) - J_\mu)Q_\mu(S,A)] - \mathbb{E}_{d_\mu}\left[(r(S,A) - J_\mu)^2\right],}$$

where Q is the differential Q-value function.

Policy evaluation using TD

Useful expression for designing TD algorithm:

$$\boxed{\kappa_\mu = 2\mathbb{E}_{d_\mu}[(r(S,A) - J_\mu)Q_\mu(S,A)] - \mathbb{E}_{d_\mu}\left[(r(S,A) - J_\mu)^2\right],}$$

where Q is the differential Q-value function.

Contributions for solving the policy evaluation problem for asymptotic variance.

- TD for both tabular and linear function approximation settings
- Finite sample error bounds with $\tilde{O}(1/k)$ rate of convergence for the mean-squared error

References

- Agarwal, A. et al. (2020). “Optimality and Approximation with Policy Gradient Methods in Markov Decision Processes”. In: *Conference on Learning Theory*. Vol. 125, pp. 64–66.
- Carmon, Yair and John Duchi (2019). “Gradient descent finds the cubic-regularized nonconvex Newton step”. In: *SIAM Journal on Optimization* 29.3, pp. 2146–2178.
- Ge, R. et al. (2015). “Escaping From Saddle Points – Online Stochastic Gradient for Tensor Decomposition”. In: *Conference of Learning Theory*.

Jin, Chi et al. (2021). "On Nonconvex Optimization for Machine Learning: Gradients, Stochasticity, and Saddle Points". In: *J. ACM* 68.2. ISSN: 0004-5411.

Kumar, Harshat, Alec Koppel, and Alejandro Ribeiro (2023). "On the sample complexity of actor-critic method for reinforcement learning with function approximation". In: *Machine Learning*, pp. 1–35.

Maniyar, M. P. et al. (2023). "A Cubic-regularized Policy Newton Algorithm for Reinforcement Learning". In: *arXiv preprint arXiv:2304.10951*.

Nesterov, Yurii and Boris Polyak (Aug. 2006). "Cubic regularization of Newton method and its global performance". In: *Math. Program.* 108, pp. 177–205.

Papini, M. et al. (2018). "Stochastic Variance-Reduced Policy Gradient". In: *ICML*. Vol. 80, pp. 4026–4035.

Shen, Z. et al. (2019). "Hessian aided policy gradient". In: *International Conference on Machine Learning*. PMLR, pp. 5729–5738.

Sutton, R. S. et al. (1999). "Policy gradient methods for reinforcement learning with function approximation.". In: *Advances in Neural Information Processing Systems*. Vol. 99, pp. 1057–1063.

Tripuraneni, N. et al. (2018). "Stochastic Cubic Regularization for Fast Nonconvex Optimization". In: *NeurIPS*. Vol. 31. Curran Associates, Inc.

Vijayan, N. and L. A. Prashanth (2021). "Smoothed functional-based gradient algorithms for off-policy reinforcement learning". In: *Systems & Control Letters* 155, p. 104988.

- Yang, L., Q. Zheng, and G. Pan (2021). "Sample Complexity of Policy Gradient Finding Second-Order Stationary Points.". In: *AAAI* 12, pp. 10630–10638.
- Zhang, K. et al. (2020). "Global Convergence of Policy Gradient Methods to (Almost) Locally Optimal Policies". In: *SIAM Journal on Control and Optimization* 58.6, pp. 3586–3612.

The Way It Is by William Stafford

There’s a thread you follow. It goes among
things that change. But it doesn’t change.
People wonder about what you are pursuing.
You have to explain about the thread.
But it is hard for others to see.
While you hold it you can’t get lost.
Tragedies happen; people get hurt
or die; and you suffer and get old.
Nothing you do can stop time’s unfolding.
You don’t ever let go of the thread.

Excerpt From “The Poetry Pharmacy”