general-purpose algorithms

topic: general-purpose optimization / learning algorithms for control

refs:

Annual Review of Control, Robotics, and Autonomous Systems

A Tour of Reinforcement Learning: The View from Continuous Control

Benjamin Recht

Annu. Rev. Control Robot. Auton. Syst. 2019. 2:253–79

Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning

RONALD J. WILLIAMS

Machine Learning, 8, 229–256 (1992)

Random Gradient-Free Minimization of Convex Functions

Yurii Nesterov & Vladimir Spokoiny

Foundations of Computational Mathematics **17**, 527–566 (2017) | Cite this article

Sampling can be faster than optimization

Yi-An Ma[a], Yuansi Chen[b], Chi Jin[a], Nicolas Flammarion[a], and Michael I. Jordan[a,b,1]

PNAS | October 15, 2019 | vol. 116 | no. 42 | 20881–20885

- consider the unconstrained nonlinear program

$$(\text{NLP}) \quad \min \ c(u) \quad \text{s.t.} \ u \in \mathbb{R}^m$$

"cost"

$c : \mathbb{R}^m \to \mathbb{R}$

$: u \mapsto c(u)$

"decision variable(s)"

* want: u^* s.t. $c(u^*) < c(u)$ $\underbrace{\text{for all } u \neq u^*}_{\text{global minimizer}}$ $\underbrace{(\text{s.t. } \|u - u^*\| < \varepsilon)}_{\text{local minimizer}}$

idea: iteratively choose u^+ such that $c(u^+) < c(u)$

and hope that $u \mapsto u^+ \mapsto u^{++} \mapsto \cdots \mapsto u^*$

* we'll usually assume c has some regularity — continuous, differentiable, etc

$\longrightarrow$ how to choose u^+?

→ how to choose u^+?

* two main strategies:

1°. randomize

$$\underbrace{u^+ \sim p(u)}$$

u^+ is a random variable w/ probability density p

2°. descend

step size / learning rate

$Dc : \mathbb{R}^m \to \mathbb{R}^{m \times 1}$ is the gradient

$$\underbrace{u^+ = u - \gamma \cdot Dc(u)^T}$$

need to have access to gradient, exactly or approximately

ex.:

	1°. randomize	2°. descend
ex.:	∘ genetic/swarm algorithms	∘ gradient descent
	∘ simulated annealing	∘ Newton-Raphson, conjugate-gradient
	∘ Markov Chain Monte Carlo	∘ zero-th order methods
properties:	+ approximate global u^*	− approximate local u^*
	+ easy implementation	+ easy[-ish] implementation
	− sample-inefficient	+ sample-efficient
	* need continuous c	* need differentiable c

∘ let's consider a specific class of randomized algorithms that leverage gradient-like information

* instead of $\min c(u)$ s.t. $u \in \mathbb{R}^m$ consider $\min E[c(u)]$ s.t. $u \sim p$

p is a probability distribution

→ how do minima/minimizers of these two problems relate?

→ how do minima/minimizers of these two problems relate?

- if "δ" distributions are allowed, then minima/minimizers coincide
  (and also note that, since $\int p = 1$, the 2nd problem's cost can't be lower than first)

  * but there are (uncountably) infinite "δ"-distributions

∘ in practice, we go one step further:

p_θ is parameterized by $\theta \in \Theta$.

* instead of $\min E[c(u)]$ s.t. $u \sim p$ consider $\min E[c(u)]$ s.t. $u \sim p_\theta$

⤳ richness of parameterization of p_θ trades off:

tractability vs suboptimality

idea: "log-likelihood trick"

$$
\begin{aligned}
D_\theta \underset{u \sim p_\theta}{E}[c(u)] &= D_\theta \int c(u) \cdot p_\theta(u)\, du && \text{– def. of expectation} \\
&= \int c(u) \cdot D_\theta p_\theta(u)\, du && \text{– assuming } D_\theta \text{ \& } \textstyle\int \text{ commute} \\
&= \int c(u) \cdot D_\theta p_\theta(u) \cdot \frac{p_\theta(u)}{p_\theta(u)}\, du && \text{– assuming } p_\theta(u) \neq 0 \\
&= \int [c(u) \cdot D_\theta \log p_\theta(u)] \cdot p_\theta(u)\, du && \text{– } D_x \log f(x) = \frac{D_x f(x)}{f(x)} \\
&= E[c(u) \cdot D_\theta \log p_\theta(u)] && \text{– def. of expectation}
\end{aligned}
$$

* if we sample $u \sim p_\theta(u)$, compute $D_\theta \log p_\theta(u)$, & evaluate $c(u)$, then by averaging samples we can approximate derivative of $E[c(u)]$ without derivative $D_u c$ √

Then by averaging samples we can approximate derivative of E[C(u)]
<u>without</u> derivative $D_u C$!