



Reinforcement Learning

Lecture 19: Policy Gradient

28 Oct, 2025

Instructor: Raunak P. Bhattacharyya



ScAI

YARDI SCHOOL OF ARTIFICIAL INTELLIGENCE
INDIAN INSTITUTE OF TECHNOLOGY DELHI

Recap

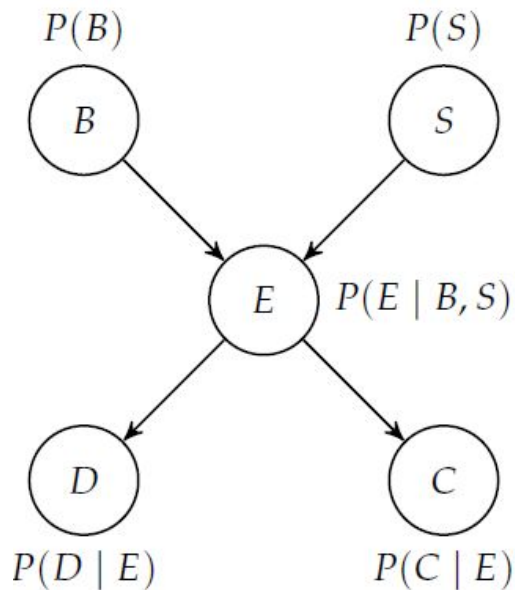
- Motivating PG
- Aliased grid world
- Derivative-Free opt
- Finite difference

Plan

- Likelihood weighting and score function
- PG intuition and computation
- Variance reduction

Why is it called 'Likelihood Ratio' PG?

Satellite Failure Example: Bayes Net

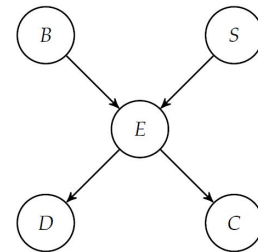


- 5 binary vars: battery failure, solar panel failure, electrical failure, deviation from planned path, comms loss
- C and D observed: evidence vars

How do we compute the probability that nothing is wrong?

$$P(b^0, s^0, e^0, d^0, c^0)$$

Number of parameters



$$P(b^0, s^0, e^0, d^0, c^0)$$

If we want to specify the full joint distribution, how many independent params would we need?

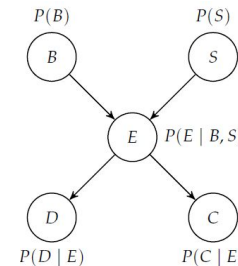
$$2^5 - 1 = 31$$

$$P(b^0, s^0, e^0, d^0, c^0) = P(b^0)P(s^0)P(e^0 \mid b^0, s^0)P(d^0 \mid e^0)P(c^0 \mid e^0)$$

How many params do we need now?

$$1 + 1 + 4 + 2 + 2 = 10$$

Inference Task



- Compute the probability that battery failure has occurred given that we observe both trajectory deviation and loss of comms with our satellite

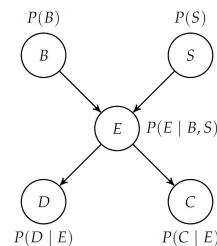
$$P(b^1 \mid d^1, c^1) = \frac{P(b^1, d^1, c^1)}{P(d^1, c^1)}$$

$$P(b^1, d^1, c^1) = \sum_s \sum_e P(b^1, s, e, d^1, c^1)$$

$$P(b^1, s, e, d^1, c^1) = P(b^1)P(s)P(e \mid b^1, s)P(d^1 \mid e)P(c^1 \mid e)$$

Exact inference in Bayes Nets is NP-hard. Thus, approximate inference

Direct Sampling



$$P(b^1 \mid d^1, c^1)$$

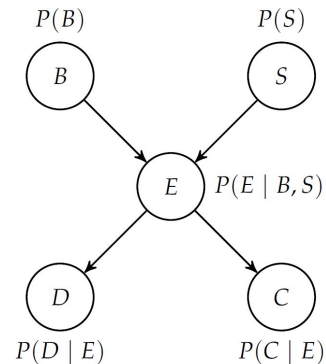
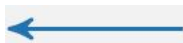
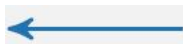
$$(b^{(i)}, s^{(i)}, e^{(i)}, d^{(i)}, c^{(i)})$$

$$P(b^1 \mid d^1, c^1) \approx \frac{\sum_i (b^{(i)} = 1 \wedge d^{(i)} = 1 \wedge c^{(i)} = 1)}{\sum_i (d^{(i)} = 1 \wedge c^{(i)} = 1)}$$

- Start from a topological sort of the vars
i.e., arrange parent nodes before child
nodes
- Sample from the joint distribution in the
order of the topological sort

Direct Sampling

B	S	E	D	C
0	0	1	1	0
0	0	0	0	0
1	0	1	0	0
1	0	1	1	1
0	0	0	0	0
0	0	0	1	0
0	0	0	0	1
0	1	1	1	1
0	0	0	0	0
0	0	0	1	0



$$P(b^1 \mid d^1, c^1)$$

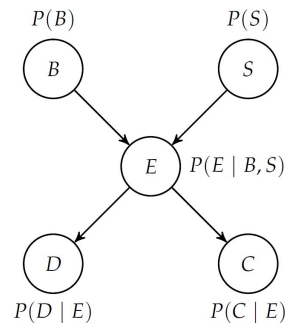
What is our inference based on the 10 samples obtained?

What is the issue with direct sampling?

Will need to discard many samples inconsistent with evidence

Likelihood Weighted Sampling

Generate samples consistent with evidence.
Weight by how likely the evidence variable values are



$$P(b^1 \mid d^1, c^1)$$

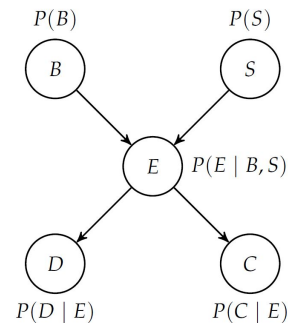
- Sample from $P(B)$
- Sample from $P(S)$
- Sample from $P(E|B,S)$
- For D and C , assign $D=1, C=1$

- Weight sample based on E

$$P(d^1 \mid e^1)P(c^1 \mid e^1)$$

$$P(d^1 \mid e^0)P(c^1 \mid e^0)$$

Likelihood Weighted Sampling



$$P(b^1 \mid d^1, c^1)$$

$$P(b^1 \mid d^1, c^1) \approx \frac{\sum_i w_i(b^{(i)} = 1 \wedge d^{(i)} = 1 \wedge c^{(i)} = 1)}{\sum_i w_i(d^{(i)} = 1 \wedge c^{(i)} = 1)}$$

$$= \frac{\sum_i w_i(b^{(i)} = 1)}{\sum_i w_i}$$

B	S	E	D	C	Weight
1	0	1	1	1	$P(d^1 \mid e^1)P(c^1 \mid e^1)$
0	1	1	1	1	$P(d^1 \mid e^1)P(c^1 \mid e^1)$
0	0	0	1	1	$P(d^1 \mid e^0)P(c^1 \mid e^0)$
0	0	0	1	1	$P(d^1 \mid e^0)P(c^1 \mid e^0)$
0	0	1	1	1	$P(d^1 \mid e^1)P(c^1 \mid e^1)$

$$P(d^1 \mid e^1)P(c^1 \mid e^1) = 0.95$$

$$P(d^1 \mid e^0)P(c^1 \mid e^0) = 0.01$$

$$P(b^1 \mid d^1, c^1) = \frac{0.95}{0.95 + 0.95 + 0.01 + 0.01 + 0.95} \approx 0.331$$

Likelihood Ratio PG

$$\nabla_{\theta} J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \left(\sum_{t=1}^T \nabla_{\theta} \log \pi_{\theta}(a_{i,t} \mid s_{i,t}) \right) \left(\sum_{t=1}^T r(s_{i,t}, a_{i,t}) \right)$$

Score Function Examples

$$\pi_{\theta}(s, a) \propto e^{\phi(s, a)^{\top} \theta}$$

$$\nabla_{\theta} \log \pi_{\theta}(s, a) = \phi(s, a) - \mathbb{E}_{\pi_{\theta}} [\phi(s, \cdot)]$$

$$\mu(s) = \phi(s)^{\top} \theta$$

$$a \sim \mathcal{N}(\mu(s), \sigma^2)$$

$$\nabla_{\theta} \log \pi_{\theta}(s, a) = \frac{(a - \mu(s))\phi(s)}{\sigma^2}$$

Example: LR gradient computation

Consider the single-step, single-state problem from example 11.1. Suppose we have a stochastic policy π_{θ} that samples its action according to a Gaussian distribution $\mathcal{N}(\theta_1, \theta_2^2)$, where θ_2^2 is the variance.

$$\begin{aligned}\log \pi_{\theta}(a | s) &= \log \left(\frac{1}{\sqrt{2\pi\theta_2^2}} \exp \left(-\frac{(a - \theta_1)^2}{2\theta_2^2} \right) \right) \\ &= -\frac{(a - \theta_1)^2}{2\theta_2^2} - \frac{1}{2} \log(2\pi\theta_2^2)\end{aligned}$$

The gradient of the log policy likelihood is

$$\begin{aligned}\frac{\partial}{\partial \theta_1} \log \pi_{\theta}(a | s) &= \frac{a - \theta_1}{\theta_2^2} \\ \frac{\partial}{\partial \theta_2} \log \pi_{\theta}(a | s) &= \frac{(a - \theta_1)^2 - \theta_2^2}{\theta_2^3}\end{aligned}$$

Suppose we run three rollouts with $\theta = [0, 1]$, taking actions $\{0.5, -1, 0.7\}$ and receiving the same rewards ($R(s, a) = a$). The estimated policy gradient is

$$\begin{aligned}\nabla U(\theta) &\approx \frac{1}{m} \sum_{i=1}^m \nabla_{\theta} \log p_{\theta}(\tau^{(i)}) R(\tau^{(i)}) \\ &= \frac{1}{3} \left(\begin{bmatrix} 0.5 \\ -0.75 \end{bmatrix} 0.5 + \begin{bmatrix} -1.0 \\ 0.0 \end{bmatrix} (-1) + \begin{bmatrix} 0.7 \\ -0.51 \end{bmatrix} 0.7 \right) \\ &= [0.58, -0.244]\end{aligned}$$

Gradient Ascent: Constrained Optimisation

The Reinforce Algorithm

$$\nabla_{\theta} J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \left(\sum_{t=1}^T \nabla_{\theta} \log \pi_{\theta}(a_{i,t} | s_{i,t}) \right) \left(\sum_{t=1}^T r(s_{i,t}, a_{i,t}) \right)$$

REINFORCE: Monte-Carlo Policy-Gradient Control (episodic) for π_*

Input: a differentiable policy parameterization $\pi(a|s, \theta)$

Algorithm parameter: step size $\alpha > 0$

Initialize policy parameter $\theta \in \mathbb{R}^{d'}$ (e.g., to $\mathbf{0}$)

Loop forever (for each episode):

 Generate an episode $S_0, A_0, R_1, \dots, S_{T-1}, A_{T-1}, R_T$, following $\pi(\cdot|\cdot, \theta)$

 Loop for each step of the episode $t = 0, 1, \dots, T - 1$:

$$G \leftarrow \sum_{k=t+1}^T \gamma^{k-t-1} R_k \tag{G_t}$$

$$\theta \leftarrow \theta + \alpha \gamma^t G \nabla \ln \pi(A_t | S_t, \theta)$$

Trivia

234

R.J. WILLIAMS

$$\Delta w_{ij} = \alpha_{ij}(r - b_{ij})e_{ij},$$

where α_{ij} is a *learning rate factor*, b_{ij} is a *reinforcement baseline*, and $e_{ij} = \partial \ln g_i / \partial w_{ij}$ is called the *characteristic eligibility* of w_{ij} . Suppose further that the reinforcement baseline b_{ij} is conditionally independent of y_i , given $\mathbf{W}$ and $\mathbf{x}^i$, and the rate factor α_{ij} is nonnegative and depends at most on $\mathbf{w}^i$ and t . (Typically, α_{ij} will be taken to be a constant.) Any learning algorithm having this particular form will be called a *REINFORCE* algorithm. The name is an acronym for “*RE*ward *I*ncrement = *N*onnegative *F*actor $\times$ *O*ffset *R*einforcement $\times$ *C*haracteristic *E*ligibility,” which describes the form of the algorithm.

2-step Example: Why does PG work?

PG using AutoDiff

$$\nabla_{\theta} J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \left(\sum_{t=1}^T \nabla_{\theta} \log \pi_{\theta}(a_{i,t} \mid s_{i,t}) \right) \left(\sum_{t=1}^T r(s_{i,t}, a_{i,t}) \right)$$

Implementing PG