CS395T: Foundations of Machine Learning for Systems Researchers

Fall 2025

Lecture 9: Baseline Functions in Policy Gradient Methods

Game plan for policy gradient lectures

Deterministic and stochastic/probabilistic control

- Policy network and its parameters θ

Basic *policy gradient* method: REINFORCE

- Monte Carlo sampling to estimate gradient of expected return
- Suffers from high variance $\rightarrow$ requires many samples

Baseline methods for reducing variance

- Based on *control variates* methods from Monte Carlo literature
- Use *advantage* in place of reward
- *Actor-critic* methods

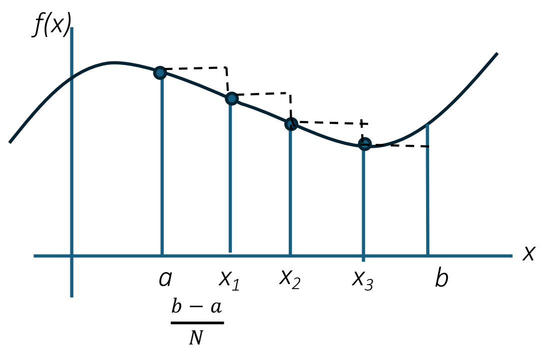
Trust-region methods for improving stability and sample efficiency

- Ensure updates to θ are small
- Trust Region Policy Optimization (TRPO): KL-divergence
- Proximal Policy Optimization (PPO): bounding box around current values

Control variates
in
Monte Carlo Methods

Review: Monte Carlo Integration (I)

$$\hat{A}_{FE}(f;N) = (b-a) * \underbrace{\frac{1}{N}\sum_{i=0}^{N-1} f(x_i)}_{average\ of\ f(x_i)\ values} \quad \text{where } x_i = a + \frac{b-a}{N} * i$$

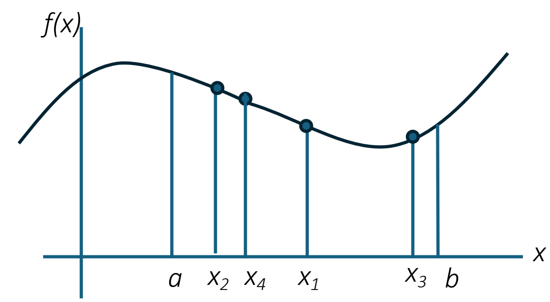


Reinterpret forward-Euler

Generate x_i values using arithmetic progression

Take average of $f(x_i)$ values and multiply by $(b-a)$

$$\hat{A}_{MC}(f;N) = (b-a) * \underbrace{\frac{1}{N}\sum_{i=0}^{N-1} f(X_i)}_{average\ of\ f(X_i)\ values} \quad \text{where } X_i \sim U(a,b)$$



Monte Carlo integration (capital letters for random variables)

Generate X_i values by sampling uniform distribution *U(a,b)*

Take average of $f(X_i)$ values and multiply by $(b-a)$

Review: Monte Carlo Integration (II)

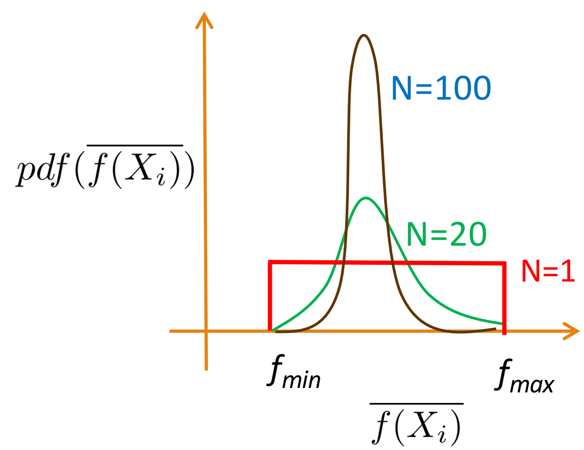
$$\hat{A}_{MC}(f;N) = (b-a) * \underbrace{\frac{1}{N}\sum_{i=0}^{N-1} f(X_i)}_{average\ of\ f(X_i)\ values} \quad \text{where } X_i \sim U(a,b)$$

Unbiased estimator

$$\mathop{\mathbb{E}}_{X_i\sim U(a,b)}\left[\hat{A}_{MC}(f;N)\right] = \frac{b-a}{N}\sum_{i=0}^{N-1}\int_a^b \underbrace{\frac{1}{b-a}}_{\text{pdf of U[a,b]}} f(X_i)dX_i = \int_a^b f(x)dx$$

$$\lim_{N\to\infty} \hat{A}_{MC}(f;N) = \int_a^b f(x)dx \quad \text{(Law of large numbers)}$$

$$\sigma_U^2(\hat{A}_{MC}(f;N)) = \frac{(b-a)^2}{N}\sigma_U^2(f) = O(\frac{\sigma_U^2(f)}{N})$$



Convergence in probability

If f has high variance, need lots of samples to obtain accurate estimate

Baseline Functions for Reducing Variance

$$h(x) = \underbrace{f(x)}_{\text{function of interest}} - \underbrace{g(x)}_{\text{easy to compute}} + \underbrace{\mathbb{E}[g(x)]}_{\text{easy to compute}}$$

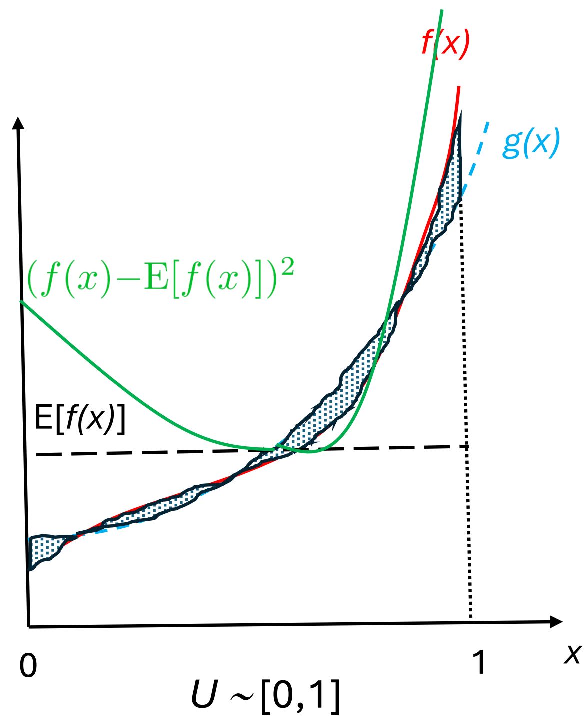
Distribution for expectations and variances: $U \sim [0, 1]$
$\mathbb{E}[g(x)]$ should be a good approximation for $\mathbb{E}[f(x)]$
Special case of *control variates* in Monte Carlo literature

Intuition:
(i) $\mathbb{E}[h(x)] = \mathbb{E}[f(x)]$
(ii) $\mathbb{E}[g(x)]$ is computed analytically
(ii) $\mathbb{E}[f(x) - g(x)]$ provides correction to $\mathbb{E}[g(x)]$ and is estimated by sampling

$$\mathbb{E}[f(x)] = \mathbb{E}[h(x)] \approx \left(\frac{1}{N} \sum_{i=1}^{N} f(x_i) - g(x_i) \right) + \mathbb{E}[g(x)]$$

Variance: $\sigma^2(h(x)) = \sigma^2(f(x)) + \sigma^2(g(x)) - 2 * Cov(f(x), g(x))$
Win if $Cov(f(x), g(x)) > \frac{1}{2}\sigma^2(g(x))$ $(f(x),\ g(x)$ are strongly correlated)
Note: variance can *increase* if *g(x)* is badly chosen!

g(x) is *baseline* for computing $\mathbb{E}[f(x)]$



Optimal Baseline

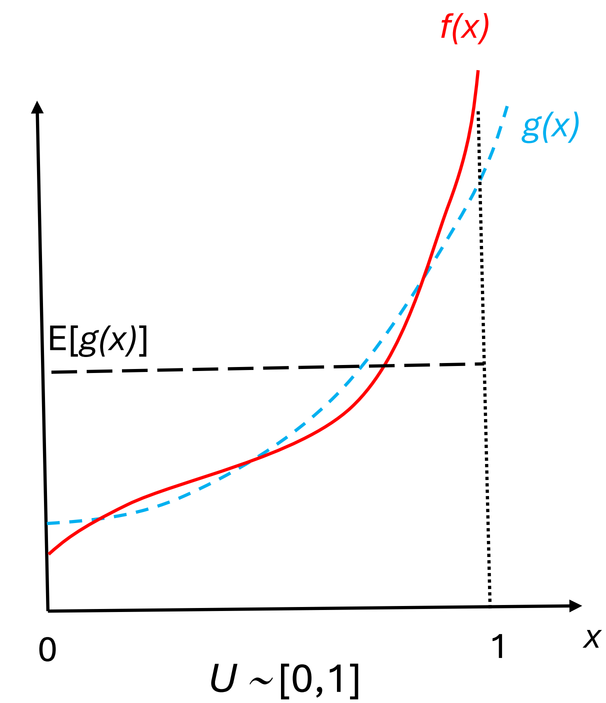
$$h(x) = \underbrace{f(x)}_{\text{function of interest}} - \underbrace{g(x)}_{\text{easy to compute}} + \underbrace{\mathbb{E}[g(x)]}_{\text{easy to compute}}$$

$$\text{Variance: } \sigma^2(h(x)) = \sigma^2(f(x)) + \sigma^2(g(x)) - 2 * Cov(f(x), g(x))$$

Optimal baseline: $g(x)$ that minimizes variance $\sigma^2(h(x))$, is easy to compute, and whose expectation can be computed analytically!

Hard to solve in general but we may be able to find best function in set of parameterized functions $g(x;\phi)$ for use as baseline

Common baseline in policy gradients: *constant*
Paradox: variance of $h(x)$ does not decrease if $g(x)$=*constant*!
Resolution to paradox: see later



g(x) is a baseline for *f(x)*

Baselines
in
Policy Gradients

Key steps

$$\nabla_\theta V^\theta(S0) \approx \frac{1}{|M|}\sum_{e\in M\sim\pi^\theta}\Big(R(e)\nabla_\theta \log \pi^\theta(e)\Big)$$

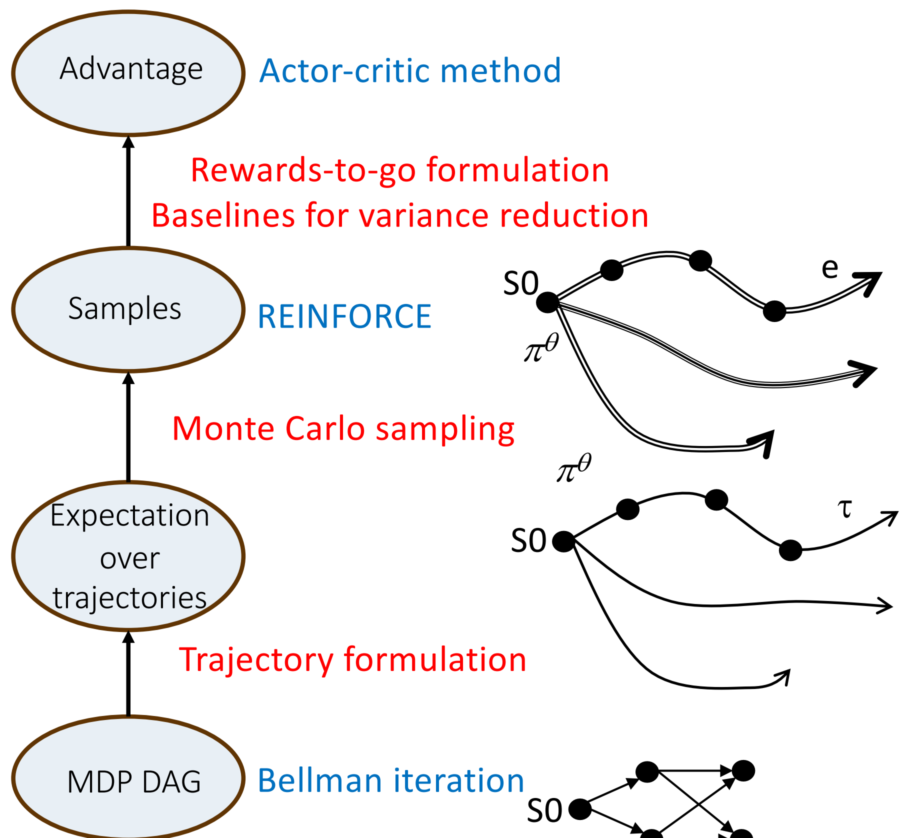
M : multiset of samples (realizations of trajectories)

$$V^\theta(S0) = \mathop{\mathbb{E}}_{\tau\sim\pi^\theta}[R(\tau)]$$

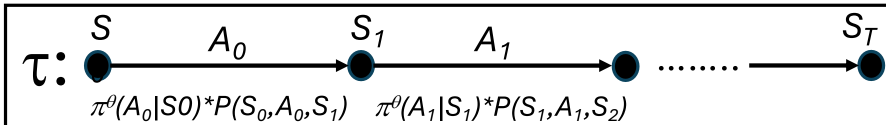
$$\nabla_\theta V^\theta(S0) = \mathop{\mathbb{E}}_{\tau\sim\pi^\theta}[R(\tau)\nabla_\theta \log \pi^\theta(\tau)]$$

$$\theta = \theta + \eta\nabla_\theta V^\theta(S0)$$

$V^\theta(S0)$



Review: policy gradient and REINFORCE



$$\rho(\tau) = \prod_{i=0}^{T-1} \pi^\theta(A_i|S_i) * P(S_i, A_i, S_{i+1}) = \underbrace{\prod_{i=0}^{T-1} \pi^\theta(A_i|S_i)}_{\pi^\theta(\tau)} * \underbrace{\prod_{i=0}^{T-1} P(S_i, A_i, S_{i+1})}_{P(\tau)}$$

$$V^\theta[S0] = \sum_{\tau\sim\pi^\theta} \rho(\tau)R(\tau) \quad \text{(where } R(\tau) = \text{sum of (discounted) rewards on trajectory)}$$

$$\nabla_\theta V^\theta[S0] = \sum_{\tau\sim\pi^\theta} \nabla_\theta \rho(\tau) * R(\tau) = \sum_{\tau\sim\pi^\theta} P(\tau) * \nabla_\theta \pi^\theta(\tau) * R(\tau) = \sum_{\tau\sim\pi^\theta} P(\tau) * \pi^\theta(\tau) * \frac{\nabla_\theta \pi^\theta(\tau)}{\pi^\theta(\tau)} * R(\tau)$$

$$= \sum_{\tau\sim\pi^\theta} \rho(\tau) * \nabla_\theta log(\pi^\theta(\tau)) * R(\tau) = \mathbb{E}_{\tau\sim\pi^\theta}[\nabla_\theta log(\pi^\theta(\tau)) * R(\tau)]$$

REINFORCE algorithm: M is multiset of samples

$$\nabla_\theta V^\theta(S0) \approx \frac{1}{|M|} \sum_{e\in M\sim\pi^\theta} \left(R(e) \nabla_\theta \log \pi^\theta(e) \right)$$

$$\theta = \theta + \eta \nabla_\theta V^\theta(S0)$$

Barto & Sutton REINFORCE:

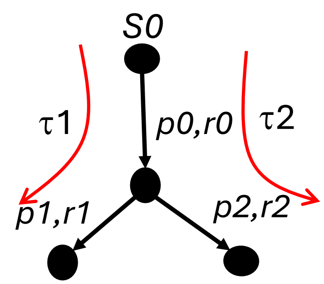
- samples are episodes
- compute state valuations for all states

Review: example

Bellman: $V(S0) = \boxed{p0{*}(r0 + (p1{*}r1 + p2{*}r2))}$

$$= p0{*}r0 + p0{*}p1{*}r1 + p0{*}p2{*}r2$$

Trajectories:



$$V(S0) = \boxed{p0{*}p1{*}(r0{+}r1) + p0{*}p2{*}(r0{+}r2)}$$

$$= p0{*}p1{*}r0 + p0{*}p1{*}r1 + p0{*}p2{*}r0 + p0{*}p2{*}r2$$

$$= p0{*}r0{*}\underbrace{(p1{+}p2)}_{=1} + p0{*}p1{*}r1 + p0{*}p2{*}r2$$

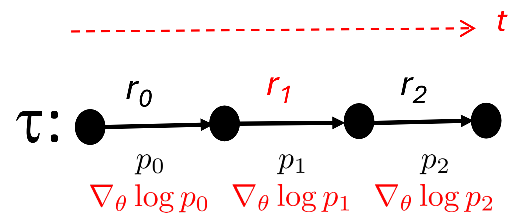
$$= p0{*}r0 + p0{*}p1{*}r1 + p0{*}p2{*}r2$$

$$\nabla_\theta V(S0) = p0{*}p1{*}\underbrace{(\nabla_\theta \log(p0 * p1))}_{\nabla_\theta log\pi^\theta(\tau_1)} * \underbrace{(r0{+}r1)}_{R(\tau_1)} + p0{*}p2{*}\underbrace{(\nabla_\theta \log(p0 * p2))}_{\nabla_\theta log\pi^\theta(\tau_2)} * \underbrace{(r0{+}r2)}_{R(\tau_2)}$$

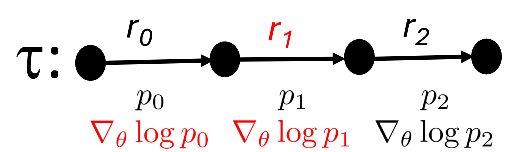
$$\theta \leftarrow \theta + \eta \nabla_\theta V(S0)$$

Gradient of expected return: other formulations

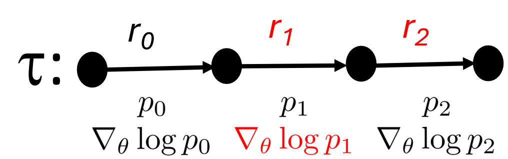
Notation: $\tau[i,j]$ = segment of trajectory between steps i and j inclusive



$$\nabla_\theta \mathbb{E}_{\tau\sim\pi^\theta}\Big[R(\tau)\Big] = \underbrace{\mathbb{E}_{\tau\sim\pi^\theta}[\nabla_\theta log(\pi^\theta(\tau)) * R(\tau)]}_{\text{trajectory formulation}} \qquad (1)$$



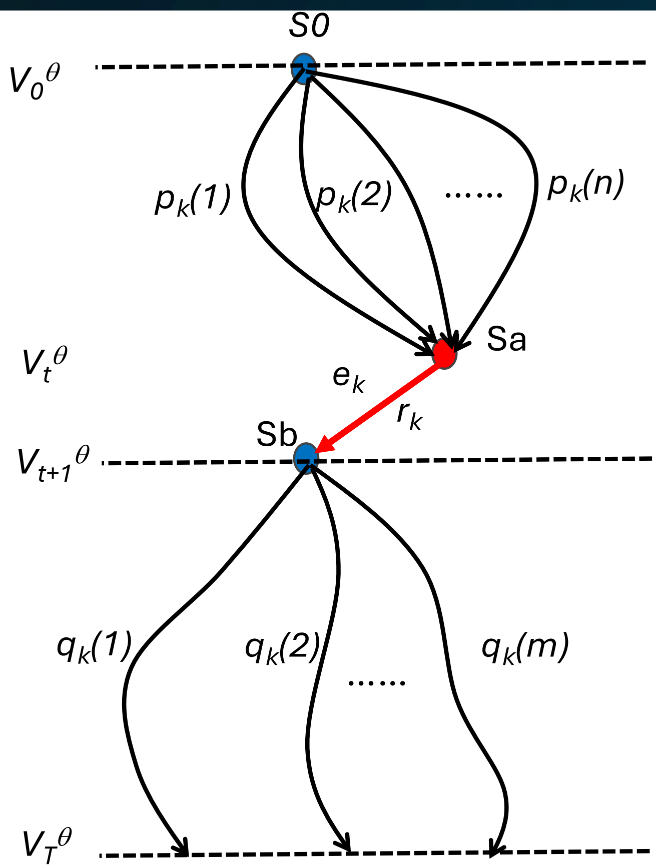
$$= \underbrace{\mathbb{E}_{\tau\sim\pi^\theta}\left[\sum_{t=0}^{T-1} R(\tau[t,t])\nabla_\theta \log \pi^\theta(\tau[0,t])\right]}_{\text{rewards formulation}} \qquad (2)$$



$$= \underbrace{\mathbb{E}_{\tau\sim\pi^\theta}\left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi^\theta(\tau[t,t]) R(\tau[t,T-1])\right]}_{\text{rewards-to-go formulation}} \qquad (3)$$

Formulations (2) and (3) are obviously equivalent:
each reward is multiplied by log prob’s earlier in trajectory and summed

Proof: trajectory and rewards formulations equivalent



Grad-Log-Pi Lemma: $\mathbb{E}_{\tau\sim\pi^\theta}[\nabla_\theta log(\pi^\theta(\tau))] = 0$

Proof:

$$\sum_{\tau\sim\pi^\theta} \rho(\tau) = 1 \Rightarrow \sum_{\tau\sim\pi^\theta} \nabla_\theta \rho(\tau) = 0 \Rightarrow \sum_{\tau\sim\pi^\theta} P(\tau)\nabla_\theta \pi(\tau) = 0 \Rightarrow \sum_{\tau\sim\pi^\theta} P(\tau)\pi^\theta(\tau)\nabla_\theta log(\pi(\tau)) = 0$$

$$\Rightarrow E_{\tau\sim\pi^\theta}[\nabla_\theta log\pi^\theta(\tau)] = 0$$

$$\begin{aligned}
\nabla_\theta \mathbb{E}_{\tau\sim\pi^\theta}\Big[R(\tau)\Big] &= \underbrace{\mathbb{E}_{\tau\sim\pi^\theta}[\nabla_\theta log(\pi^\theta(\tau)) * R(\tau)]}_{\text{trajectory formulation}} \\
&= \underbrace{\mathbb{E}_{\tau\sim\pi^\theta}\left[\sum_{t=0}^{T-1} R(\tau[t,t])\nabla_\theta \log \pi^\theta(\tau[0,t])\right]}_{\text{rewards formulation}} \\
&+ \underbrace{\mathbb{E}_{\tau\sim\pi^\theta}\left[\sum_{t=0}^{T-1} R(\tau[t,t])\nabla_\theta \log \pi^\theta(\tau[t{+}1,T{-}1])\right]}_{=0}
\end{aligned}$$

Example

Bellman: $V(S0) = \boxed{p0{*}(r0 + (p1{*}r1 + p2{*}r2))}$

$= p0{*}r0 + p0{*}p1{*}r1 + p0{*}p2{*}r2$

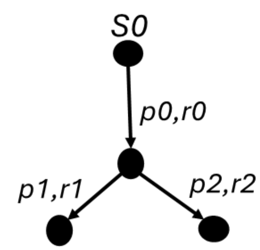
Trajectory: $V(S0) = \boxed{p0{*}p1{*}(r0{+}r1) + p0{*}p2{*}(r0{+}r2)}$

$= p0{*}p1{*}r0 + p0{*}p1{*}r1 + p0{*}p2{*}r0 + p0{*}p2{*}r2$

$= p0{*}r0{*}\underbrace{(p1{+}p2)}_{=1}{+}p0{*}p1{*}r1 + p0{*}p2{*}r2$

$= p0{*}r0 + p0{*}p1{*}r1 + p0{*}p2{*}r2$

$$\nabla V(S0) = \boxed{p0{*}p1{*}(\nabla \log p0{+}\nabla \log p1)(r0{+}r1) + p0{*}p2{*}(\nabla \log p0{+}\nabla \log p2)(r0{+}r2)}$$



$p1 + p2 = 1$

$\nabla p1 + \nabla p2 = 0$

Reward formulation:

$$\nabla V(S0) = p0{*}p1{*}\boxed{(r0{*}\nabla \log p0{+}r1{*}(\nabla \log p0{+}\nabla \log p1))} + p0{*}p2{*}\boxed{(r0{*}\nabla \log p0{+}r2{*}(\nabla \log p0{+}\nabla \log p2))}$$

Rewards-to-go formulation:

$$\nabla V(S0) = p0{*}p1{*}\boxed{((r0{+}r1){*}\nabla \log p0{+}r1{*}\nabla \log p1))} + p0{*}p2{*}\boxed{((r0{+}r2){*}\nabla \log p0{+}r2{*}\nabla \log p2))}$$

Sample gradient and variance

$$V^{\theta}[S0] = \sum_{\tau \sim \pi^{\theta}} \rho(\tau) R(\tau)$$

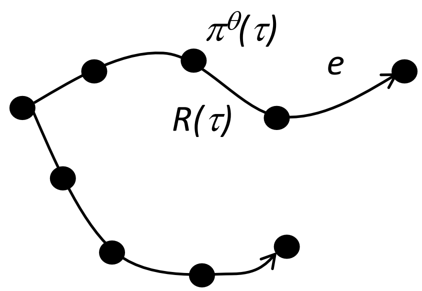
$$\nabla_{\theta} V^{\theta}[S0] = \sum_{\tau \sim \pi^{\theta}} \nabla_{\theta} \rho(\tau) * R(\tau) = \sum_{\tau \sim \pi^{\theta}} \rho(\tau) * \nabla_{\theta} log(\pi^{\theta}(\tau)) * R(\tau) = \mathbb{E}_{\tau \sim \pi^{\theta}} [\, \underbrace{\nabla_{\theta} log(\pi^{\theta}(\tau)) * R(\tau)}_{sample\ gradient\ SG^{\theta}(\tau)} \,]$$

Interpretation of $\nabla_{\theta} V^{\theta}[S0]$: expectation of a *random variable* SG^{θ} (*for Sample Gradient*) that takes value $\nabla_{\theta} log(\pi^{\theta}(\tau)) * R(\tau)$ with probability $\rho(\tau)$ for $\tau \sim \pi^{\theta}$

Variance of SG^{θ}: assume θ is scalar (otherwise we need covariance matrices)

$$\sigma^2(SG^{\theta}) = \sum_{\tau \sim \pi^{\theta}} \rho(\tau) * (\nabla_{\theta} log(\pi^{\theta}(\tau)) * R(\tau))^2 - \left(\sum_{\tau \sim \pi^{\theta}} \rho(\tau) * \nabla_{\theta} log(\pi^{\theta}(\tau)) * R(\tau) \right)^2$$

Intuition for θ updates in REINFORCE



$$\nabla_\theta V^\theta(S0) \approx \frac{1}{|M|} \sum_{e \in M \sim \pi^\theta} \Big(R(e) \nabla_\theta \log \pi^\theta(e) \Big)$$

Update to gradient from sample $e \propto R(e)\nabla_\theta \log \pi^\theta(e)$

$\theta = \theta + \eta \nabla_\theta V^\theta(S0)$

Claim: R(τ) > 0: update to θ makes τ more probable
R(τ) < 0 : update to θ makes τ less probable

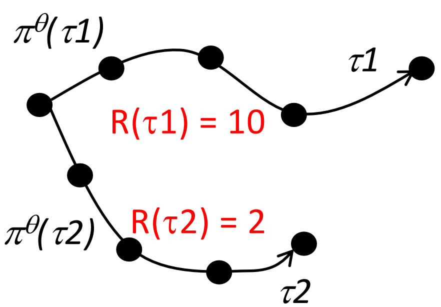
Proof: Assume θ is scalar; proof for vector θ applies proof below dimension by dimension. Assume $R(\tau)>0$, $\nabla_\theta \pi^\theta(\tau) > 0$; proof for other cases is similar.

$\nabla_\theta \pi^\theta(\tau)>0 \Rightarrow \nabla_\theta \log \pi^\theta(\tau)>0 \Rightarrow R(\tau) * \nabla_\theta \log \pi^\theta(\tau)>0$

$\Rightarrow \tau$ samples push $\nabla_\theta V^\theta(S0)$ higher $\Rightarrow \theta$ is pushed higher.

Since $\nabla_\theta \pi^\theta(\tau)>0$, this implies $\pi^\theta(\tau)$ is pushed higher.

Inefficiency in REINFORCE



$$\nabla_\theta V^\theta(S0) \approx \frac{1}{|M|} \sum_{e \in M \sim \pi^\theta} \Big(R(e) \nabla_\theta \log \pi^\theta(e) \Big)$$

Update to gradient from sample $e \propto R(e) \nabla_\theta \log \pi^\theta(e)$

$$\theta = \theta + \eta \nabla_\theta V^\theta(S0)$$

R(τ) > 0: trajectory becomes more probable
R(τ) < 0 : trajectory becomes less probable

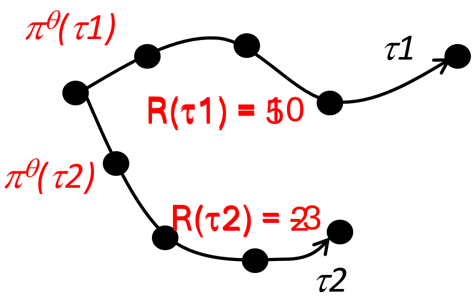
Gradient ascent should converge to θ that maximizes $\pi^\theta(\tau 1)$

However, whenever $\tau 2$ is sampled, it pulls π^θ to itself since $R(\tau 2) > 0$.

Result: θ updates are noisy, and convergence is slow.

Intuitively, REINFORCE has no memory of past samples except to extent they are incorporated implicitly into π^θ

Intuitive idea of constant baseline



$$\nabla_\theta V^\theta(S0) \approx \frac{1}{|M|} \sum_{e \in M \sim \pi^\theta} \Big(R(e) \nabla_\theta \log \pi^\theta(e) \Big)$$

Update to gradient from sample $e \propto R(e)\nabla_\theta \log \pi^\theta(e)$

$$\theta = \theta + \eta \nabla_\theta V^\theta(S0)$$

R(τ) > 0: trajectory becomes more probable
R(τ) < 0 : trajectory becomes less probable

Idea: subtract a constant *b*(aseline) between 2 and 10 (say 5) from both rewards.

Now τ2 has negative reward so it repels probabilities while τ1 still attracts probabilities

Result: θ updates are less noisy, and convergence is faster

If b > 10 or b < 2, variance increases!

Question: is there an *optimal* baseline value for reducing variance of gradient estimates?

One guess: optimal $b = (R_{max} - \varepsilon)$ where ε is some small constant (turns out to be wrong!)

Two main results (I)

$$V^{\theta}[S0] = \sum_{\tau \sim \pi^{\theta}} \rho(\tau) R(\tau)$$

$$\nabla_{\theta} V^{\theta}[S0] = \sum_{\tau \sim \pi^{\theta}} \nabla_{\theta} \rho(\tau) * R(\tau) = \sum_{\tau \sim \pi^{\theta}} \rho(\tau) * \nabla_{\theta} log(\pi^{\theta}(\tau)) * R(\tau) = \mathbb{E}_{\tau \sim \pi^{\theta}}[\ \underbrace{\nabla_{\theta} log(\pi^{\theta}(\tau)) * R(\tau)}_{sample\ gradient\ SG^{\theta}(\tau)}\ ]$$

(1) Introducing a constant baseline does not change $\nabla_{\theta} V^{\theta}[S0]$.
Proof:

$$\mathbb{E}_{\tau \sim \pi^{\theta}}[\nabla_{\theta} log(\pi^{\theta}(\tau)) * (R(\tau) - b)] = \mathbb{E}_{\tau \sim \pi^{\theta}}[\nabla_{\theta} log(\pi^{\theta}(\tau)) * R(\tau)] - b * \underbrace{\mathbb{E}_{\tau \sim \pi^{\theta}}[\nabla_{\theta} log(\pi^{\theta}(\tau))}_{=0 \text{ from Grad-Log-Pi Lemma}}\ ] = \mathbb{E}_{\tau \sim \pi^{\theta}}[\nabla_{\theta} log(\pi^{\theta}(\tau)) * R(\tau)]$$

Point: updates to θ for gradient ascent do not change if we introduce a constant baseline.

Two main results (II)

$$\boxed{\sigma^2(SG^\theta; b) = \sum_{\tau\sim\pi^\theta} \rho(\tau) * \Big(\nabla_\theta log(\pi^\theta(\tau)) * (R(\tau) - b)\Big)^2 - \Big(\sum_{\tau\sim\pi^\theta} \rho(\tau) * \nabla_\theta log(\pi^\theta(\tau)) * R(\tau)\Big)^2}$$

(2) Differentiating $\sigma^2(.)$ wrt b gives optimal baseline constant b^*:

$$b^* = \frac{\mathbb{E}_{\tau\sim\pi^\theta}[R(\tau)(\nabla_\theta log\pi^\theta(\tau))^2]}{\mathbb{E}_{\tau\sim\pi^\theta}[(\nabla_\theta log\pi^\theta(\tau))^2]}$$

Optimal b is *convex combination* of trajectory rewards $R(\tau)$

Generalization to multiple parameters: one approach is to ignore off-diagonal terms in covariance matrix and use different baseline values for each dimension of gradient vector (intuitively, we minimize variance in each dimension separately)

Advantage

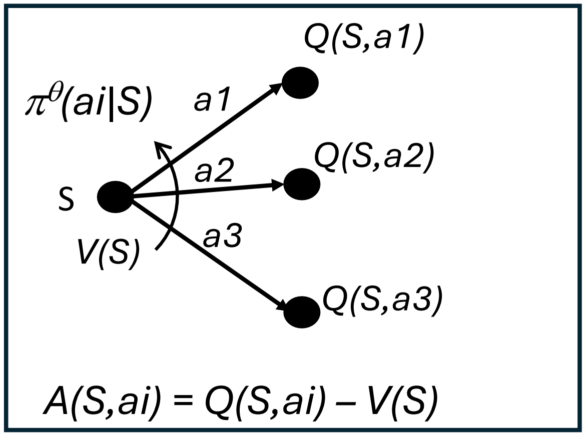
In practice, we use *V(S) as baseline*

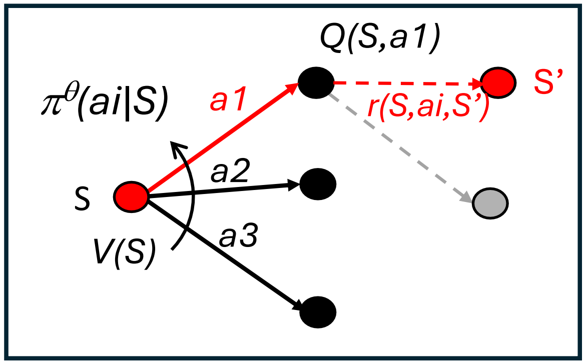
- It too is convex combination of Q values
- Intuition: V(S) is like class average in relative grading
- *Advantage A(S,ai) = Q(S,ai) – V(S)*

Intuition: using *V(S)* as a baseline adds "history-sensitivity" to gradient updates. If we sampled better trajectories at S the last few times we visited *S*, reduce gradient update from current sample.

Some implementations approximate *A(S,ai)* by TD(0) error

$$A(S,ai) \approx \underbrace{r(S,ai,S') + \gamma V(S')}_{\text{Approximation to } Q(S,ai)} - V(S)$$





Actor-critic mechanism

Actor NN updates θ by gradient ascent using V values estimated by the critic

Critic NN estimates V values which serve as baseline

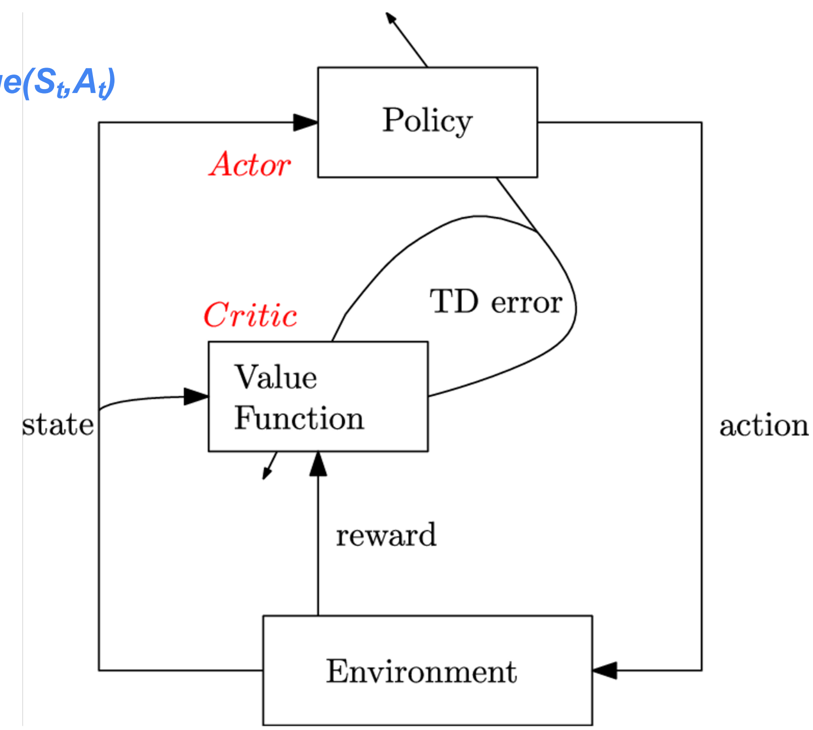
TD error: $\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$ Approximation to *Advantage(S_t,A_t)*

Critic update: $w \leftarrow w + \alpha_w \delta_t \nabla_w V(S_t; w)$

Actor update: $\theta \leftarrow \theta + \alpha \nabla_\theta \log \pi_\theta(A_t|S_t)\delta_t$

Algorithm 3 Actor-Critic Method

```
Initialize policy parameter θ ∈ R^d
Initialize value function parameter w ∈ R^d
for each episode do
  for each step of the episode do
    Take action A_t according to π_θ
    Observe reward R_{t+1} and next state S_{t+1}
    δ_t ← R_{t+1} + γV(S_{t+1}; w) − V(S_t; w)      → Calculate TD error
    w ← w + α_w δ_t ∇_w V(S_t; w)                    → Update the value function
    θ ← θ + α∇_θ log π_θ(A_t|S_t)δ_t                 → Update policy parameters
  end for
end for
```



Sutton & Barto REINFORCE with Value Baseline

Algorithm 2 REINFORCE with Baseline

Initialize policy parameter $\theta \in R^d$
Initialize baseline parameter $w \in R^d$
for each episode **do**
  Generate an episode $S_0, A_0, R_1, \ldots, S_T, A_T, R_{T+1}$ following π_θ
  for $t = 0$ to T **do**
    $G_t \leftarrow \sum_{k=t}^{T} \gamma^{k-t} R_{k+1}$
    $\delta_t \leftarrow G_t - V(S_t; w)$ — **Calculate advantage**
    $\theta \leftarrow \theta + \alpha \gamma^t \delta_t \nabla_\theta \log \pi_\theta(A_t|S_t)$ — **Update the policy parameter θ using rewards (advantage)-to-go**
    $w \leftarrow w + \alpha_w \delta_t \nabla_w V(S_t; w)$ — **Update the Value NN parameter *w***
  end for
end for

From Sutton & Barto

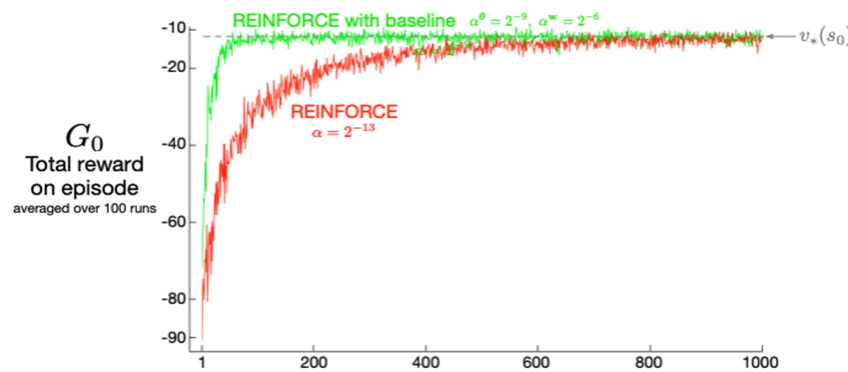
Experiments

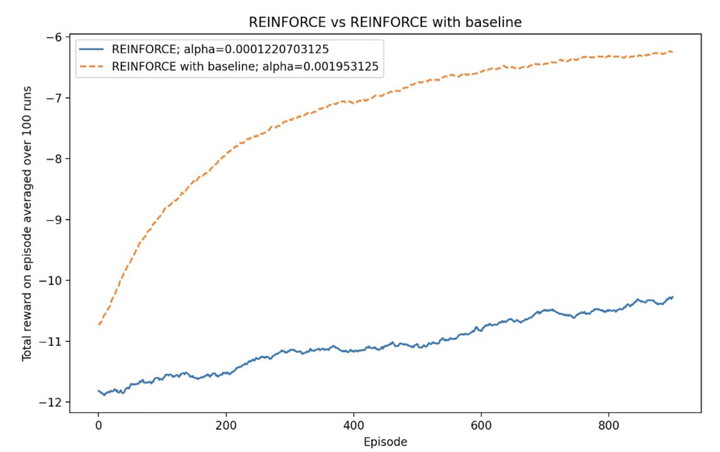
Learning Rate for Policy Parameters (α_θ): 2^{-13}

Learning Rate for Policy Parameters (α_θ): 2^{-9}

Learning Rate for Baseline Parameters (α_w): 2^{-6}

Baseline reduces variance, allowing
REINFORCE with baseline to converge faster.





Key concepts covered in lecture

Control variate method for reducing variance in Monte Carlo sampling

- In RL, usually only constant baseline functions

Basic policy gradient method: REINFORCE

- Trajectory formulation, rewards, *rewards-to-go*
- Sample gradients
- Monte Carlo sampling to estimate gradient of expected return
- Suffers from high variance → requires many samples

Baseline functions for reducing variance: usually constant in RL

- Optimal baseline
- *Advantage*: use state valuations as baselines
- *Actor-critic* mechanism
    - Critic NN: estimates state valuations
    - Actor NN: policy network, uses state valuations from actor to compute advantage

Other presentations: much more complex

Based on continuous state, continuous action MDP’s
- End up with double integrals instead of double summations

Continuous tasks instead of finite-horizon tasks
- Unbounded trajectories and infinite sums
- Need ideas like continuity

Start with trajectory formulation
- Difficult to get intuition for rewards and rewards-to-go formulations

Assume start state can be any state, not a fixed state
- Need concepts like state occupancy distribution in Markov processes
- Easy to add to our narrative once simpler formulation is clear

Visualization
- Unrolled MDP for probabilistic policies

References

Lil'Log: Policy Gradient Algorithms: long blog post on policy gradient algorithms. Well-written.

- https://lilianweng.github.io/posts/2018-04-08-policy-gradient/

The Definitive Guide to Policy Gradients in Deep Reinforcement Learning: Theory, Algorithms, and Implementations, Matthias Lehmann. Considers continuous state, continuous action MDPs so lots of double integrals!

- https://arxiv.org/pdf/2401.13662

Variance Reduction in Policy Gradient, David Rosenberg NYU-CDS DS-GA 3001 lecture notes. Approaches subject from data science/statistics perspective.

- https://davidrosenberg.github.io/ttml2021fall/bandits/6.PG-variance-reduction.pdf

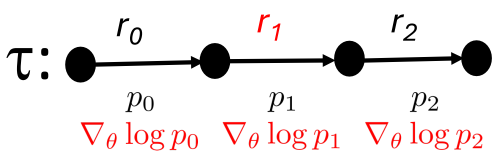
Deep Reinforcement Learning, Sergey Levine, CS 285 Berkeley. Course by one of the leaders in the field.

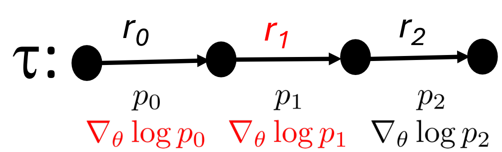
- https://rail.eecs.berkeley.edu/deeprlcourse/

OpenAI Spinning Up. Introduction to Policy Optimization. Has links to libraries and systems resources.

- https://spinningup.openai.com/en/latest/spinningup/rl_intro3.html

Gradient of expected return: other formulations

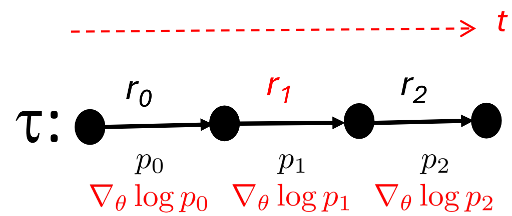




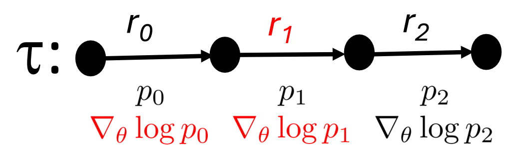
$$
\begin{aligned}
\nabla_\theta \mathbb{E}_{\tau\sim\pi^\theta}\Big[R(\tau)\Big] &= \underbrace{\mathbb{E}_{\tau\sim\pi^\theta}[\nabla_\theta log(\pi^\theta(\tau)) * R(\tau)]}_{\text{trajectory formulation}} \\
&= \underbrace{\mathbb{E}_{\tau\sim\pi^\theta}\left[\sum_{t=0}^{T-1} R(\tau[t,t])\nabla_\theta \log \pi^\theta(\tau[0,t])\right]}_{\text{rewards formulation}} + \underbrace{\mathbb{E}_{\tau\sim\pi^\theta}\left[\sum_{t=0}^{T-1} R(\tau[t,t])\nabla_\theta \log \pi^\theta(\tau[t{+}1,T{-}1])\right]}_{\text{Prove this is =0}}
\end{aligned}
$$

Gradient of expected return: other formulations

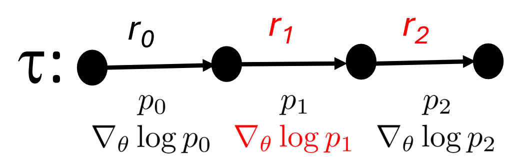
$$\nabla_\theta \mathbb{E}_{\tau\sim\pi^\theta}\Big[R(\tau)\Big] = \underbrace{\mathbb{E}_{\tau\sim\pi^\theta}\left[\left(\sum_{t=0}^{T-1} r_t\right)\cdot\left(\sum_{t=0}^{T-1}\nabla_\theta \log \pi^\theta(A_t|S_t)\right)\right]}_{\text{trajectory formulation}} \qquad (1)$$



$$= \underbrace{\mathbb{E}_{\tau\sim\pi^\theta}\left[\sum_{t=0}^{T-1}\left(r_t \sum_{t'=0}^{t}\nabla_\theta \log \pi^\theta(A'_t|S_{t'})\right)\right]}_{\text{rewards formulation}} \qquad (2)$$



$$= \underbrace{\mathbb{E}_{\tau\sim\pi^\theta}\left[\sum_{t=0}^{T-1}\left(\nabla_\theta \log \pi^\theta(A_t|S_t)\left(\sum_{t'=t}^{T-1} r_{t'}\right)\right)\right]}_{\text{rewards-to-go formulation}} \qquad (3)$$



Formulations (2) and (3) are obviously equivalent:
each reward is multiplied by log prob's earlier in trajectory and summed