Policy Gradients methods

Ciprian Paduraru

University of Bucharest

Agenda

Part 1 (**today**):

- Recap & Motivation
- Policy objective and optimization
- Reinforce algorithm
- A3C and GAE
- Results

Part 2

- Other different ways to choose the policy network – gaussian, softmax, etc.
- Address the problem when the policy network is not differentiable
- A more formal look at Policy Theorem
- Examples from practice, real implementations

Part 3

- Importance of optimization method, step size tuning
- Use Importance sampling to have monotonic improvenets
- Lower bounds for local Approximation, Trust Regions, TRPO Algorithm
- PPO
- A look at Alpha-Go

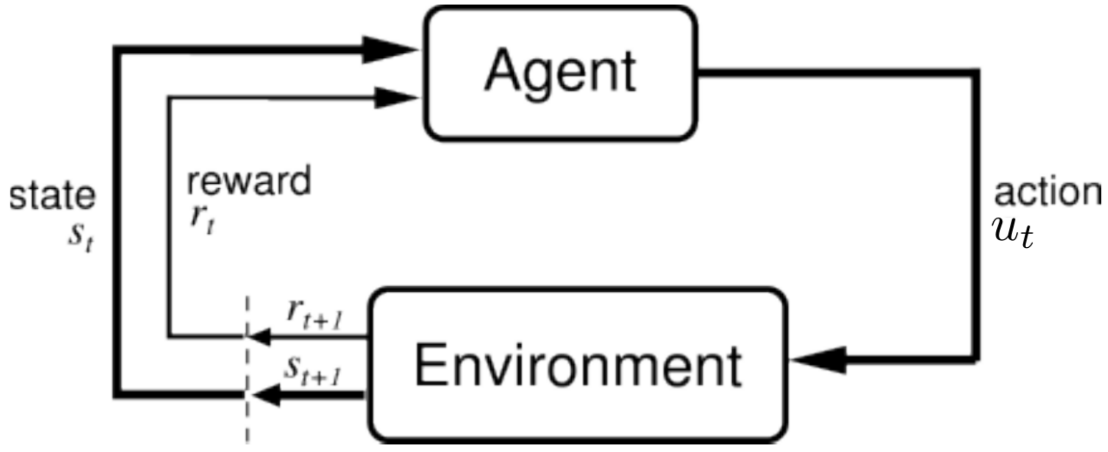
Lab 1: Experiment with base class algorithms on Cart-Pole

Lab 2: Experiment with various parameters on DeepMimic environment

1. Short recap

Last lectures:

- The general approach:



[Figure source: Sutton & Barto, 1998]

- Have looked at (exact or approximate) approaches for finding $V(s)$, $Q(s,a)$

- A policy was generated directly from the value function

$$\pi(s) = \arg\max_a Q(s,a)$$

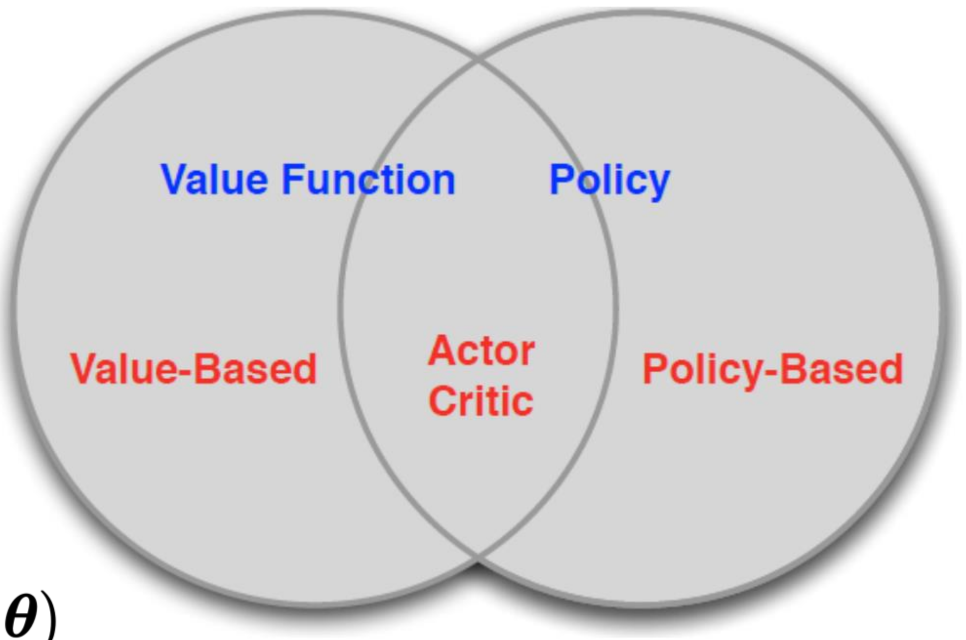
- In this lecture, we'll **directly parameterize the policy** (Policy-based RL)

- Focus will still be on **model-free reinforcement learning**

- **Value-based**
  - Learn $V(s)$, $Q(s,a)$
  - Extract the policy from $V(s)$, $Q(s,a)$

- **Policy-based**
  - No intermediate learning of value functions
  - Directly learn the policy $\pi_{\boldsymbol{\theta}}(a \mid s) = P(a \mid s; \boldsymbol{\theta})$

- **Actor-Critic** = Combination of both worlds
  - Learn $V(s)$, $Q(s,a)$ -> **CRITIC**
  - Learn Policy $\pi_{\boldsymbol{\theta}}(a \mid s)$ -> **ACTOR**
  - Note: many times, better than pure Policy-based approaches !



2. Properties of Policy-based RL, motivation

Advantages

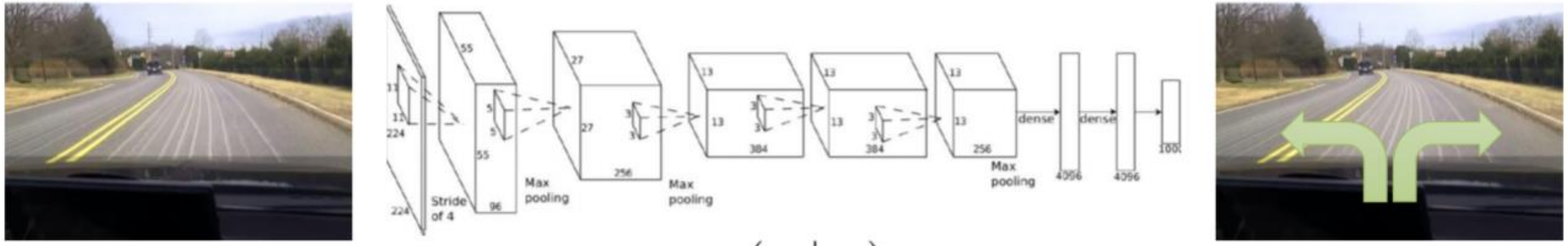
- Better convergence properties – more details in The Sutton and Barto reference, Ch 13.3
- Effectiveness in high-dimensional or continuous action spaces

Examples:

- Robotics: Isn't it harder to compute $V(s)$, $Q(s,a)$ for every state / action of a robot rather than a policy ?

- In IRL (Inverse Reinforcement Learning) settings,
  isn't it simpler to think in terms of a policy directly instead of computing the value for each state ?

Think about the self-driving cars field !



$\mathbf{o}_t$ $\quad\quad$ $\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$ $\quad\quad$ $\mathbf{a}_t$

- Can learn **stochastic policies** (remember previously with MDPs we had deterministic policies).

  Why is a stochastic policy needed ?
  - State representation is not Markov
  - Adversarial / non-stationary domain problems

  Example: think about *rock-paper-scissors* game. Can someone exploit a deterministic strategy ? ☺.

  https://www.timeforkids.com/k1/rock-paper-scissors/

Disadvantages

- Typically converges to a local instead of global optimum (however also true for value-based RL with function approximation)

- Evaluating policy (expected reward) is typically inefficient and high variance

3. Policy optimization

- Denote ***trajectory*** $\boldsymbol{\tau} = (\boldsymbol{s_0}, \boldsymbol{u_0}, \boldsymbol{r_1}, \boldsymbol{s_2}, \boldsymbol{u_1}, \boldsymbol{r_2}, \ldots., \boldsymbol{u_{H-1}}, \boldsymbol{r_H}, \boldsymbol{s_H}, \ldots.)$

- Actions along $\boldsymbol{\tau}$ are taken using policy $\pi_\theta(a|s)$, based on parameters θ used by a function π_θ
- Methods to evaluate the policy:
  - In episodic environments, can use the *start value* of the policy:

$$J(\theta) = V^{\pi_\theta}(s_1) = E_{\pi_\theta}[v_1]$$

  - In continuing environments, stationary distribution of π_θ over states can be used: $d^{\pi_\theta}(s)$

$$J_{avV}(\theta) = \sum_s d^{\pi_\theta}(s) V^{\pi_\theta}(s)$$

- Note: We'll discuss the episodic case, but all results can be reused in the non-episodic ones
- Also, considering finite horizons of length H.

3. Policy optimization

- ***Trajectory*** $\boldsymbol{\tau} = (\boldsymbol{s_0}, \boldsymbol{u_0}, \boldsymbol{r_1}, \boldsymbol{s_2}, \boldsymbol{u_1}, \boldsymbol{r_2}, \dots., \boldsymbol{u_{H-1}}, \boldsymbol{r_H}, \boldsymbol{s_H}, \dots.)$
- **Reward** of a trajectory: $R(\tau) = \sum_{t=0}^{H} R(s_t, u_t)$

- **Objective** function (expected sum of rewards along trajectories samples from π_θ:

$$U(\theta) = \mathrm{E}\left[\sum_{t=0}^{H} R(s_t, u_t); \pi_\theta\right] = \sum_{\tau} P(\tau;\theta) R(\tau)$$

- Our **goal** find the optimal θ:

$$\max_\theta U(\theta) = \max_\theta \sum_\tau P(\tau;\theta) R(\tau)$$

3.1 Non-differentiable optimization methods

- Having an objective function and a parameters space, we can treat the policy-based RL problem as an optimization one.

- There are some **gradient free methods** such as :
  - Hill climbing
  - Simplex / amoeba / Nelder Mead
  - Genetic algorithms
  - Cross-Entropy method (CEM)
  - Covariance Matrix Adaptation (CMA)
  - Evolution strategies

- **Their advantage**: These allows the policy parametrization to be non-differentiable and are often easy to parallelize

  Disadvantage: these methods ignore the temporal structure of rewards
  - ➢ Updates consider only the total episodes' rewards, do not break up the reward for each state in the trajectory !

3.2 Policy gradient

- We assume policy π_θ is differentiable whenever it is non-zero

Softmax Policy class

➢ Weight actions by using l.c. of features and parameters $\phi(s,a)^T\theta$

➢ Probability of actions , $\pi_\theta(s,a) = \frac{e^{\phi(s,a)^T\theta}}{\left(\sum_a e^{\phi(s,a)^T\theta}\right)}$ $\Rightarrow$ $\nabla_\theta \log \pi_\theta(s,a) = \phi(s,a) - \mathbb{E}_{\pi_\theta}[\phi(s,\cdot)]$

Full proof:

Using the log identity $\log(x/y) = \log(x) - \log(y)$ we can write

$$\log(\pi_\theta(s,a)) = \log(e^{\phi(s,a)^\intercal\theta}) - \log(\sum_{k=1}^{N} e^{\phi(s,a_k)^\intercal\theta})$$

“Likelihood Ratio” Policy Gradient

$$\max_\theta U(\theta) = \max_\theta \sum_\tau P(\tau;\theta)R(\tau)$$

- We take the gradient with respect $to\ \boldsymbol{\theta}$

$$\begin{aligned}\nabla_\theta U(\theta) &= \nabla_\theta \sum_\tau P(\tau;\theta)R(\tau) \\ &= \sum_\tau \nabla_\theta P(\tau;\theta)R(\tau)\end{aligned}$$

“Likelihood Ratio” Policy Gradient

$$\max_{\theta} U(\theta) = \max_{\theta} \sum_{\tau} P(\tau;\theta) R(\tau)$$

- We take the gradient with respect $to\ \boldsymbol{\theta}$

$$\begin{aligned}\nabla_\theta U(\theta) &= \nabla_\theta \sum_{\tau} P(\tau;\theta) R(\tau) \\ &= \sum_{\tau} \nabla_\theta P(\tau;\theta) R(\tau) \\ &= \sum_{\tau} \frac{P(\tau;\theta)}{P(\tau;\theta)} \nabla_\theta P(\tau;\theta) R(\tau)\end{aligned}$$

“Likelihood Ratio” Policy Gradient

$$\max_{\theta} U(\theta) = \max_{\theta} \sum_{\tau} P(\tau;\theta) R(\tau)$$

- We take the gradient with respect $to\ \boldsymbol{\theta}$

$$\begin{aligned}
\nabla_\theta U(\theta) &= \nabla_\theta \sum_{\tau} P(\tau;\theta) R(\tau) \\
&= \sum_{\tau} \nabla_\theta P(\tau;\theta) R(\tau) \\
&= \sum_{\tau} \frac{P(\tau;\theta)}{P(\tau;\theta)} \nabla_\theta P(\tau;\theta) R(\tau) \\
&= \sum_{\tau} P(\tau;\theta) \frac{\nabla_\theta P(\tau;\theta)}{P(\tau;\theta)} R(\tau)
\end{aligned}$$

“Likelihood Ratio” Policy Gradient

$$\max_{\theta} U(\theta) = \max_{\theta} \sum_{\tau} P(\tau;\theta) R(\tau)$$

- We take the gradient with respect $to\ \boldsymbol{\theta}$

$$\begin{aligned}
\nabla_\theta U(\theta) &= \nabla_\theta \sum_{\tau} P(\tau;\theta) R(\tau) \\
&= \sum_{\tau} \nabla_\theta P(\tau;\theta) R(\tau) \\
&= \sum_{\tau} \frac{P(\tau;\theta)}{P(\tau;\theta)} \nabla_\theta P(\tau;\theta) R(\tau) \\
&= \sum_{\tau} P(\tau;\theta) \frac{\nabla_\theta P(\tau;\theta)}{P(\tau;\theta)} R(\tau) \quad = \sum_{\tau} P(\tau;\theta) \nabla_\theta \log P(\tau;\theta) R(\tau)
\end{aligned}$$

“Likelihood Ratio” Policy Gradient

$$\max_{\theta} U(\theta) = \max_{\theta} \sum_{\tau} P(\tau;\theta) R(\tau)$$

- We take the gradient with respect to $\boldsymbol{\theta}$

$$\begin{aligned}
\nabla_\theta U(\theta) &= \nabla_\theta \sum_{\tau} P(\tau;\theta) R(\tau) \\
&= \sum_{\tau} \nabla_\theta P(\tau;\theta) R(\tau) \\
&= \sum_{\tau} \frac{P(\tau;\theta)}{P(\tau;\theta)} \nabla_\theta P(\tau;\theta) R(\tau) \\
&= \sum_{\tau} P(\tau;\theta) \frac{\nabla_\theta P(\tau;\theta)}{P(\tau;\theta)} R(\tau) \quad = \sum_{\tau} P(\tau;\theta) \nabla_\theta \log P(\tau;\theta) R(\tau) \\
&\qquad\qquad\qquad\qquad\qquad\qquad\quad = \mathbb{E}_\tau [\nabla_\theta \log P(\tau;\theta) R(\tau)]
\end{aligned}$$

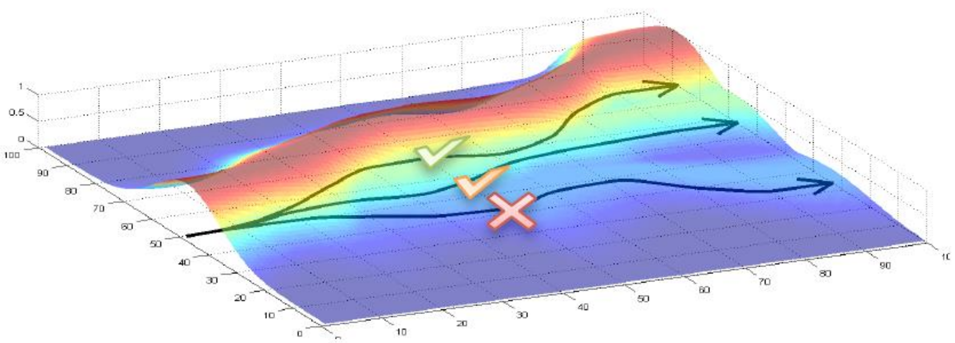
$$\nabla_\theta U(\theta) = \sum_\tau P(\tau;\theta)\nabla_\theta \log P(\tau;\theta)R(\tau) = \mathbb{E}_\tau[\nabla_\theta \log P(\tau;\theta)R(\tau)]$$

- We can now approximate with the empirical estimate for m sample paths under policy – Monte Carlo (MC) sampling method from the previous courses !

- Collect m episodes then compute the average

$$\nabla U(\theta) \approx \hat{g} = \frac{1}{m}\sum_{i=1}^{m} \nabla_\theta \log P\left(\tau^{(i)};\theta\right) R\left(\tau^{(i)}\right)$$

- **Intuition** - The gradient tries to:
  - ➢ Increase the probability of paths with positive reward
  - ➢ Decrease the probability of paths with negative rewards



- Let's go further: Decompose paths into states and actions along trajectory $\tau = (s_0, u_0, r_1, s_1, u_1, r_2, \ldots, u_{H-1}, r_H, s_H)$

$$\nabla U(\theta) \approx \hat{g} = \frac{1}{m} \sum_{i=1}^{m} \underbrace{\nabla_\theta \log P\left(\tau^{(i)}; \theta\right)}_{\text{Let's take a step closer at this}} R\left(\tau^{(i)}\right)$$

$$\nabla_\theta \log P\left(\tau^{(i)}; \theta\right) = \nabla_\theta \log\left[\mu(s_0) \cdot \prod_{t=0}^{H-1} P\left(s_{t+1}^{(i)} \mid s_t^{(i)}, u_t^{(i)}\right) \cdot \pi_\theta\left(u_t^{(i)} \mid s_t^{(i)}\right)\right]$$

- Let's go further: Decompose paths into states and actions along trajectory $\tau = (s_0, u_0, r_1, s_1, u_1, r_2, \ldots, u_{H-1}, r_H, s_H)$

$$\nabla U(\theta) \approx \hat{g} = \frac{1}{m} \sum_{i=1}^{m} \underbrace{\nabla_\theta \log P\left(\tau^{(i)}; \theta\right)}_{\text{Let's take a step closer at this}} R\left(\tau^{(i)}\right)$$

$$\nabla_\theta \log P\left(\tau^{(i)}; \theta\right) = \nabla_\theta \log \left[\mu(s_0) \cdot \prod_{t=0}^{H-1} P\left(s_{t+1}^{(i)} \mid s_t^{(i)}, u_t^{(i)}\right) \cdot \pi_\theta\left(u_t^{(i)} \mid s_t^{(i)}\right)\right]$$

Initial state distribution

Model's dynamic

Policy samples

- Let’s go further: Decompose paths into states and actions along trajectory $\tau = (s_0, u_0, r_1, s_1, u_1, r_2, \ldots, u_{H-1}, r_H, s_H)$

$$\nabla_\theta \log P\left(\tau^{(i)}; \theta\right) = \nabla_\theta \log\left[\mu(s_0) \cdot \prod_{t=0}^{H-1} P\left(s_{t+1}^{(i)} \mid s_t^{(i)}, u_t^{(i)}\right) \cdot \pi_\theta\left(u_t^{(i)} \mid s_t^{(i)}\right)\right]$$

(using log property of products to sum)

$$= \nabla_\theta\left[\mu(s_0) + \sum_{t=0}^{H-1} \log P\left(s_{t+1}^{(i)} \mid s_t^{(i)}, u_t^{(i)}\right) + \sum_{t=0}^{H} \log \pi_\theta\left(u_t^{(i)} \mid s_t^{(i)}\right)\right]$$

- Let's go further: Decompose paths into states and actions along trajectory $\tau = (s_0, u_0, r_1, s_1, u_1, r_2, \ldots, u_{H-1}, r_H, s_H)$

$$\nabla_\theta \log P(\tau^{(i)}; \theta) = \nabla_\theta \log\left[\mu(s_0) \cdot \prod_{t=0}^{H-1} P\left(s_{t+1}^{(i)} \mid s_t^{(i)}, u_t^{(i)}\right) \cdot \pi_\theta\left(u_t^{(i)} \mid s_t^{(i)}\right)\right]$$

(using log property of products to sum) $$= \nabla_\theta\left[\mu(s_0) + \sum_{t=0}^{H-1} \log P\left(s_{t+1}^{(i)} \mid s_t^{(i)}, u_t^{(i)}\right) + \sum_{t=0}^{H} \log \pi_\theta\left(u_t^{(i)} \mid s_t^{(i)}\right)\right]$$

(first two don't depend on θ) $$= \nabla_\theta \sum_{t=0}^{H} \log \pi_\theta\left(u_t^{(i)} \mid s_t^{(i)}\right)$$

- Let's go further: Decompose paths into states and actions along trajectory $\tau = (s_0, u_0, r_1, s_1, u_1, r_2, \ldots, u_{H-1}, r_H, s_H)$

$$\nabla_\theta \log P(\tau^{(i)};\theta) = \nabla_\theta \log\Big[\mu(s_0)\cdot \textstyle\prod_{t=0}^{H-1} P\Big(s_{t+1}^{(i)} \mid s_t^{(i)}, u_t^{(i)}\Big)\cdot \pi_\theta\Big(u_t^{(i)} \mid s_t^{(i)}\Big)\Big]$$

(using log property of products to sum) $$= \nabla_\theta\Big[\mu(s_0) + \textstyle\sum_{t=0}^{H-1} \log P\Big(s_{t+1}^{(i)} \mid s_t^{(i)}, u_t^{(i)}\Big) + \sum_{t=0}^{H} \log \pi_\theta\Big(u_t^{(i)} \mid s_t^{(i)}\Big)\Big]$$

(first two don't depend on θ) $$= \nabla_\theta \textstyle\sum_{t=0}^{H} \log \pi_\theta\Big(u_t^{(i)} \mid s_t^{(i)}\Big)$$

$$= \textstyle\sum_{t=0}^{H} \nabla_\theta \log \pi_\theta\Big(u_t^{(i)} \mid s_t^{(i)}\Big)$$

- Let's go further: Decompose paths into states and actions along trajectory $\tau = (s_0, u_0, r_1, s_1, u_1, r_2, \dots, u_{H-1}, r_H, s_H)$

$$\nabla_\theta \log P(\tau^{(i)}; \theta) = \nabla_\theta \log\left[\mu(s_0) \cdot \prod_{t=0}^{H-1} P\left(s_{t+1}^{(i)} \mid s_t^{(i)}, u_t^{(i)}\right) \cdot \pi_\theta\left(u_t^{(i)} \mid s_t^{(i)}\right)\right]$$

(using log property of products to sum) $$= \nabla_\theta\left[\mu(s_0) + \sum_{t=0}^{H-1} \log P\left(s_{t+1}^{(i)} \mid s_t^{(i)}, u_t^{(i)}\right) + \sum_{t=0}^{H} \log \pi_\theta\left(u_t^{(i)} \mid s_t^{(i)}\right)\right]$$

(first two don't depend on θ) $$= \nabla_\theta \sum_{t=0}^{H} \log \pi_\theta\left(u_t^{(i)} \mid s_t^{(i)}\right)$$

$$= \sum_{t=0}^{H} \underbrace{\nabla_\theta \log \pi_\theta\left(u_t^{(i)} \mid s_t^{(i)}\right)}_{\text{Also named } \textbf{\textit{score function}}}$$

Note: **No model dynamics is no longer required !**

No initial state distribution !

Let’s recap !

- We have an **unbiased** estimate of the gradient
- **No need to access model dynamics**

$$\nabla U(\theta) \approx \hat{g} = \frac{1}{m} \sum_{i=1}^{m} \nabla_\theta \log P\left(\tau^{(i)}; \theta\right) R\left(\tau^{(i)}\right)$$

- **Unbiased but very noisy** (remember the MC properties from previous courses)

- There are a couple of ways to address this and reduce variance in practice:
  - Exploit the temporal structure
  - Baseline method
  - Next course: KL-Divergence trust region / natural gradient

Optimization 1: Exploit the temporal structure

- Current estimation: $$\hat{g} = \frac{1}{m} \sum_{i=1}^{m} \nabla_\theta \log P\left(\tau^{(i)}; \theta\right) R\left(\tau^{(i)}\right)$$

Optimization 1: Exploit the temporal structure

- Current estimation: $$\hat{g} = \frac{1}{m}\sum_{i=1}^{m} \nabla_\theta \log P\left(\tau^{(i)};\theta\right) R\left(\tau^{(i)}\right)$$

(expanding the trajectory's steps as before) $$= \frac{1}{m}\sum_{i=1}^{m} \left(\sum_{t=0}^{H-1} \nabla_\theta \log \pi_\theta\left(u_t^{(i)} \mid s_t^{(i)}\right)\right)\left(\sum_{t=0}^{H-1} R\left(s_t^{(i)}, u_t^{(i)}\right)\right)$$

Optimization 1: Exploit the temporal structure

- Current estimation: $$\hat{g} = \frac{1}{m} \sum_{i=1}^{m} \nabla_\theta \log P\left(\tau^{(i)}; \theta\right) R\left(\tau^{(i)}\right)$$

(expanding the trajectory's steps as before) $$= \frac{1}{m} \sum_{i=1}^{m} \left(\sum_{t=0}^{H-1} \nabla_\theta \log \pi_\theta \left(u_t^{(i)} \mid s_t^{(i)} \right) \right) \left(\sum_{t=0}^{H-1} R\left(s_t^{(i)}, u_t^{(i)} \right) \right)$$

(splitting rewards before/after t) $$= \frac{1}{m} \sum_{i=1}^{m} \left(\sum_{t=0}^{H-1} \nabla_\theta \log \pi_\theta \left(u_t^{(i)} \mid s_t^{(i)} \right) \left[\left(\sum_{k=0}^{t-1} R\left(s_k^{(i)}, u_k^{(i)} \right) \right) + \left(\sum_{k=t}^{H-1} R\left(s_k^{(i)}, u_k^{(i)} \right) \right) \right] \right)$$

Any observation ?

Optimization 1: Exploit the temporal structure

- Current estimation: $$\hat{g} = \frac{1}{m}\sum_{i=1}^{m} \nabla_\theta \log P\left(\tau^{(i)};\theta\right) R\left(\tau^{(i)}\right)$$

(expanding the trajectory's steps as before) $$= \frac{1}{m}\sum_{i=1}^{m}\left(\sum_{t=0}^{H-1}\nabla_\theta \log \pi_\theta\left(u_t^{(i)} \mid s_t^{(i)}\right)\right)\left(\sum_{t=0}^{H-1} R\left(s_t^{(i)}, u_t^{(i)}\right)\right)$$

(splitting rewards before/after t) $$= \frac{1}{m}\sum_{i=1}^{m}\left(\sum_{t=0}^{H-1}\nabla_\theta \log \pi_\theta\left(u_t^{(i)} \mid s_t^{(i)}\right)\left[\cancel{\left(\sum_{k=0}^{t-1} R\left(s_k^{(i)}, u_k^{(i)}\right)\right)} + \left(\sum_{k=t}^{H-1} R\left(s_k^{(i)}, u_k^{(i)}\right)\right)\right]\right)$$

This term (previous rewards) do not depend on u_t !
Whatever probability we choose now, this remains fixed ! (More formal definition in seminar)

- Improved version, less variance $$\hat{g} = \frac{1}{m}\sum_{i=1}^{m}\left(\sum_{t=0}^{H-1}\nabla_\theta \log \pi_\theta\left(u_t^{(i)} \mid s_t^{(i)}\right)\left(\sum_{k=t}^{H-1} R\left(s_k^{(i)}, u_k^{(i)}\right)\right)\right)$$

REINFORCE algorithm

REINFORCE (**Williams**, 1992).

- We are here:

$$\nabla U(\theta) \approx \hat{g} = \frac{1}{m} \sum_{i=1}^{m} \left(\sum_{t=0}^{H-1} \nabla_\theta \log \pi_\theta \left(u_t^{(i)} \mid s_t^{(i)} \right) \left(\sum_{k=t}^{H-1} R\left(s_k^{(i)}, u_k^{(i)} \right) \right) \right)$$

REINFORCE algorithm

REINFORCE (**Williams**, 1992).

- We are here:

$$\nabla U(\theta) \approx \hat{g} = \frac{1}{m} \sum_{i=1}^{m} \left(\sum_{t=0}^{H-1} \nabla_\theta \log \pi_\theta \left(u_t^{(i)} \mid s_t^{(i)} \right) \underbrace{\left(\sum_{k=t}^{H-1} R\left(s_k^{(i)}, u_k^{(i)} \right) \right)}_{G_t \text{ notation from MC course}} \right)$$

- At each trajectory, and step we could update the parameters of the policy by:

$$\Delta\theta_t = \alpha \nabla_\theta \log \pi_\theta(a_t | s_t) G_t$$

REINFORCE algorithm

REINFORCE (**Williams**, 1992).

- We are here: $$\nabla U(\theta) \approx \hat{g} = \frac{1}{m} \sum_{i=1}^{m} \left(\sum_{t=0}^{H-1} \nabla_\theta \log \pi_\theta \left(u_t^{(i)} \mid s_t^{(i)} \right) \left(\underbrace{\sum_{k=t}^{H-1} R \left(s_k^{(i)}, u_k^{(i)} \right)}_{G_t \text{ notation from MC course}} \right) \right)$$

Pseudocode

Initialize policy parameters θ arbitrarily
for each episode $\{s_1, a_1, r_2, \cdots, s_{T-1}, a_{T-1}, r_T\} \sim \pi_\theta$ **do**
  for $t = 1$ to $T - 1$ **do**
    $\theta \leftarrow \theta + \alpha \nabla_\theta \log \pi_\theta(s_t, a_t) G_t$
  endfor
endfor
return θ

3.3 Differentiable policy classes

- Assume policy π_θ is differentiable whenever it is non-zero

- Many choices for differentiable policy classes. Popular ones:

  - Softmax
  - Gaussian
  - Neural networks

<u>Softmax Policy class</u> – good for *discrete action spaces*

➢ Weight actions by using a linear comb of features and parameters $\phi(s, a)^T\theta$

➢ Probability of actions , $\pi_\theta(s,a) = \dfrac{e^{\phi(s,a)^T\theta}}{\left(\sum_a e^{\phi(s,a)^T\theta}\right)}$ ➡ $\nabla_\theta \log \pi_\theta(s, a) = \phi(s, a) - \mathbb{E}_{\pi_\theta}[\phi(s, \cdot)]$

Full proof:

Using the log identity $\log(x/y) = \log(x) - \log(y)$ we can write

$$\log(\pi_\theta(s,a)) = \log(e^{\phi(s,a)^\intercal\theta}) - \log(\sum_{k=1}^{N} e^{\phi(s,a_k)^\intercal\theta})$$

$$\nabla_\theta \log(\pi_\theta(s,a)) = \nabla_\theta \log(e^{\phi(s,a)^\intercal\theta}) - \nabla_\theta \log(\sum_{k=1}^{N} e^{\phi(s,a_k)^\intercal\theta})$$

$$left = \nabla_\theta \log(e^{\phi(s,a)^\intercal\theta}) = \nabla_\theta \phi(s,a)^\intercal\theta = \phi(s,a)$$

The right term simplifies as follows:

Using the chain rule:

$$\nabla_x \log(f(x)) = \frac{\nabla_x f(x)}{f(x)}$$

We can write:

$$right = \nabla_\theta \log(\sum_{k=1}^{N} e^{\phi(s,a_k)^\intercal\theta}) = \frac{\nabla_\theta \sum_{k=1}^{N} e^{\phi(s,a_k)^\intercal\theta}}{\sum_{k=1}^{N} e^{\phi(s,a_k)^\intercal\theta}}$$

Taking the gradient of the numerator we get:

$$right = \frac{\sum_{k=1}^{N} \phi(s,a_k) e^{\phi(s,a_k)^\intercal\theta}}{\sum_{k=1}^{N} e^{\phi(s,a_k)^\intercal\theta}}$$

Substituting the definition of $\pi_\theta(s,a)$ we can simplify to:

$$right = \sum_{k=1}^{N} \phi(s,a_k)\pi_\theta(s,a_k)$$

Given the definition of Expected Value:

$$\mathrm{E}[X] = X \cdot P = x_1p_1 + x_2p_2 + \ldots + x_np_n$$

Which in English is just the sum of each feature times its probability.

$$X = features = \phi(s,a)$$

$$P = probabilities = \pi_\theta(s,a)$$

So now we can write the expected value of the features:

$$right = \mathrm{E}_{\pi_\theta}[\phi(s,\cdot)]$$

where $\cdot$ means all possible actions.

Putting it all together:

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

Interpretation: difference between the features for certain states and each individual action minus mean of features for the same state and all possible actions.

Gaussian Policy - *continuous action spaces* (e.g. robotics)

- Mean could be a linear combination of state features $\mu(s) = \phi(s)^T\theta$
- Variance σ^2 could be parametrized or fixed
- Policy is then a Gaussian such that actions are sampled:

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

- Derivative (i.e., score function) becomes:

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

  - Note: Nice interpretation again ! An action is more probable if close to the mean estimated by the parameters of the state.
  - It works for continuous cases !

- Neural networks – most common

Optimization 2: Baseline trick

- We are here: $$\nabla U(\theta) \approx \hat{g} = \frac{1}{m} \sum_{i=1}^{m} \left(\sum_{t=0}^{H-1} \nabla_\theta \log \pi_\theta \left(u_t^{(i)} \mid s_t^{(i)} \right) \left(\sum_{k=t}^{H-1} R\left(s_k^{(i)}, u_k^{(i)} \right) \right) \right)$$

Optimization 2: Baseline trick

- We are here: $$\nabla U(\theta) \approx \hat{g} = \frac{1}{m} \sum_{i=1}^{m} \left(\sum_{t=0}^{H-1} \nabla_\theta \log \pi_\theta \left(u_t^{(i)} \mid s_t^{(i)} \right) \left(\sum_{k=t}^{H-1} R\left(s_k^{(i)}, u_k^{(i)} \right) \right) \right)$$

Idea: Extract a **baseline value** to improve the variance further

$$\nabla U(\theta) \approx \hat{g} = \frac{1}{m} \sum_{i=1}^{m} \left(\sum_{t=0}^{H-1} \nabla_\theta \log \pi_\theta \left(u_t^{(i)} \mid s_t^{(i)} \right) \left(\sum_{k=t}^{H-1} R\left(s_k^{(i)}, u_k^{(i)} \right) - b(s_t) \right) \right)$$

Optimization 2: Baseline trick

- We are here: $$\nabla U(\theta) \approx \hat{g} = \frac{1}{m} \sum_{i=1}^{m} \left(\sum_{t=0}^{H-1} \nabla_\theta \log \pi_\theta \left(u_t^{(i)} \mid s_t^{(i)} \right) \left(\sum_{k=t}^{H-1} R\left(s_k^{(i)}, u_k^{(i)} \right) \right) \right)$$

Idea: Extract a **baseline value** to improve the variance further

$$\nabla U(\theta) \approx \hat{g} = \frac{1}{m} \sum_{i=1}^{m} \left(\sum_{t=0}^{H-1} \nabla_\theta \log \pi_\theta \left(u_t^{(i)} \mid s_t^{(i)} \right) \left(\sum_{k=t}^{H-1} R\left(s_k^{(i)}, u_k^{(i)} \right) - b(s_t) \right) \right)$$

$$= \mathbb{E}_\tau \left[\sum_{t=0}^{H-1} \nabla_\theta \log \pi_\theta \left(u_t \mid s_t \right) \left(\sum_{k=t}^{H-1} R\left(s_k, u_k \right) - b\left(s_t \right) \right) \right]$$

(Just coming back a bit to E form)

Optimization 2: Baseline trick

- We are here: $$\nabla U(\theta) \approx \hat{g} = \frac{1}{m} \sum_{i=1}^{m} \left(\sum_{t=0}^{H-1} \nabla_\theta \log \pi_\theta \left(u_t^{(i)} \mid s_t^{(i)} \right) \left(\sum_{k=t}^{H-1} R\left(s_k^{(i)}, u_k^{(i)} \right) \right) \right)$$

Idea: Extract a **baseline value** to improve the variance further

$$\nabla U(\theta) \approx \hat{g} = \frac{1}{m} \sum_{i=1}^{m} \left(\sum_{t=0}^{H-1} \nabla_\theta \log \pi_\theta \left(u_t^{(i)} \mid s_t^{(i)} \right) \left(\sum_{k=t}^{H-1} R\left(s_k^{(i)}, u_k^{(i)} \right) - b(s_t) \right) \right)$$

- We are here:

(Just coming back a bit to E form)

- For any choice of baseline $b(s_t)$, gradient estimator is unbiased.
NOTE: it should only depend on previous states, not future, i.e., those affected by future decisions

- **For any choice of baseline $b(s_t)$, gradient estimator is unbiased**.
  NOTE: it should only depend on previous states, not future, i.e., those affected by future decisions

  We must prove that: $\mathbb{E}_\tau \left[\sum_{t=0}^{H-1} \nabla_\theta \log \pi_\theta \left(u_t \mid s_t \right) b\left(s_t \right) \right] = 0$ or even simpler, that $\mathbb{E}_\tau \left[\nabla_\theta \log \pi_\theta \left(u_t \mid s_t \right) b\left(s_t \right) \right] = 0$

[**Read more**: Policy Gradient Theorem: SuTon et al, NIPS 1999; GPOMDP: BartleT & Baxter, JAIR 2001; Survey: Peters & Schaal, IROS 2006]

Proof that by introducing baseline gradient is still unbiased:

$$\mathbb{E}_\tau\left[\nabla_\theta \log \pi_\theta\left(u_t \mid s_t\right) b\left(s_t\right)\right]$$

$$=\mathbb{E}_{s_{0:t}, a_{0:(t-1)}}\left[\mathbb{E}_{s_{(t+1):T}, a_{t:(T-1)}}\left[\nabla_\theta \log \pi_\theta\left(u_t \mid s_t\right) b\left(s_t\right)\right]\right] \text{ (split expectation before and after t)}$$

$$=\mathbb{E}_{s_{0:t}, a_{0:(t-1)}}\left[b\left(s_t\right) \mathbb{E}_{s_{(t+1):H}, a_{t:(H-1)}}\left[\nabla_\theta \log \pi_\theta\left(u_t \mid s_t\right)\right]\right] \text{ (baseline doesn't affect inner E)}$$

$$=\mathbb{E}_{s_{0:t}, a_{0:(t-1)}}\left[b\left(s_t\right) \mathbb{E}_{u_t}\left[\nabla_\theta \log \pi\left(u_t \mid s_t\right)\right]\right] \text{ (remove irrelevant s iteration)}$$

$$=\mathbb{E}_{s_{0:t}, a_{0:(t-1)}}\left[b\left(s_t\right) \sum_u \pi_\theta\left(u_t \mid s_t\right) \frac{\nabla_\theta \pi_\theta\left(u_t \mid s_t\right)}{\pi_\theta\left(u_t \mid s_t\right)}\right] \text{ (iterate over actions)}$$

$$=\mathbb{E}_{s_{0:t}, a_{0:(t-1)}}\left[b\left(s_t\right) \sum_u \nabla_\theta \pi_\theta\left(u_t \mid s_t\right)\right]$$

$$=\mathbb{E}_{s_{0:t}, a_{0:(t-1)}}\left[b\left(s_t\right) \nabla_\theta \sum_u \pi_\theta\left(u_t \mid s_t\right)\right]$$

$$=\mathbb{E}_{s_{0:t}, a_{0:(t-1)}}\left[b\left(s_t\right) \nabla_\theta 1\right]$$

$$=\mathbb{E}_{s_0: t, a_0:(t-1)}\left[b\left(s_t\right) \cdot 0\right]=0$$

[**Read more**: Policy Gradient Theorem: SuTon et al, NIPS 1999; GPOMDP: BartleT & Baxter, JAIR 2001; Survey: Peters & Schaal, IROS 2006]

- Current state:

$$\nabla U(\theta) \approx \hat{g} = \frac{1}{m} \sum_{i=1}^{m} \left(\sum_{t=0}^{H-1} \nabla_\theta \log \pi_\theta \left(u_t^{(i)} \mid s_t^{(i)} \right) \left(\sum_{k=t}^{H-1} R\left(s_k^{(i)}, u_k^{(i)} \right) - b(s_t) \right) \right)$$

- **Intuition at this point**: Increase the probability of actions proportionally to how much its returns are better than the estimated return under the current policy !

- Also called **the Advantage** of that action over baseline

How to choose the baseline ?

How to choose the baseline ?

- Constant baseline: $b = \mathbb{E}\left[R(\tau)\right] \approx \frac{1}{m}\sum_{i=1}^{m} R(\tau^{(i)})$

How to choose the baseline ?

- Constant baseline: $b = \mathbb{E}\left[R(\tau)\right] \approx \frac{1}{m}\sum_{i=1}^{m} R(\tau^{(i)})$

- Optimal constant baseline (according to variance): $b = \frac{\sum_i \left(\nabla_\theta \log P(\tau^{(i)};\theta)\right)^2 R(\tau^{(i)})}{\sum_i \left(\nabla_\theta \log P(\tau^{(i)};\theta)\right)^2}$

[**Read**: Greensmith, BartleT, Baxter, JMLR 2004 for variance reducing techniques.]

How to choose the baseline ?

- Constant baseline: $b = \mathbb{E}\left[R(\tau)\right] \approx \frac{1}{m}\sum_{i=1}^{m} R(\tau^{(i)})$
- Optimal constant baseline (according to variance): $b = \frac{\sum_i \left(\nabla_\theta \log P(\tau^{(i)};\theta)\right)^2 R(\tau^{(i)})}{\sum_i \left(\nabla_\theta \log P(\tau^{(i)};\theta)\right)^2}$
- Time-based average: $b_t = \frac{1}{m}\sum_{i=1}^{m}\sum_{k=t}^{H-1} R(s_k^{(i)}, u_k^{(i)})$
- State-based expected return: $b(s_t) = \mathbb{E}\left[r_t + r_{t+1} + r_{t+2} + \ldots + r_{H-1}\right] = \mathbb{E}\left[G_t\right]$

[**Read**: Greensmith, BartleT, Baxter, JMLR 2004 for variance reducing techniques.]

"Vanilla" Policy Gradient pseudocode

(the template of all policy gradients algorithms)

Initialize policy parameter θ, baseline b
for iterations $= 1, 2, \cdots$ do
    Collect a set of trajectories τ by executing the current policy
    for each trajectory τ_i
        At each timestep t in trajectory τ^i
            Compute Return $G_t^i = \sum_{t'=t}^{T-1} r_{t'}^i$,
            Advantage estimate $\hat{A}_t^i = G_t^i - b(s_t)$.
    Re-fit the baseline, by minimizing $\sum_i \sum_t \left\| b(s_t) - G_t^i \right\|^2$
    Update the policy, using a policy gradient estimate $\hat{g}$,
        which is a sum of terms $\nabla_\theta \log \pi_\theta(u_t \mid s_t) \hat{A}_t$.
(Plug $\hat{g}$ into SGD or ADAM) endfor

How can we improve the baseline choice ?

- Current state: We estimate the baseline target as returns, G_t , from MC samples

$$\sum_i \sum_t \left\| b(s_t) - G_t^i \right\|^2$$

- This is unbiased, but high variance, collected from individual roll-outs.

- Ideas from previous courses:
  - Reduce variance by adding bias using **bootstrapping** and **discounting**
  - Using **function approximation**

How can we improve the baseline choice ?

- Now we use : $b(s_t) = \mathbb{E}\left[r_t + r_{t+1} + r_{t+2} + \ldots + r_{H-1}\right]$

- Recall **State-value function** – can serve as a **baseline estimator**

$$V^{\pi,\gamma}(\boldsymbol{s}) = \mathbb{E}_\pi\left[\boldsymbol{r}_0 + \gamma \boldsymbol{r}_1 + \gamma^2 \boldsymbol{r}_2 \cdots \mid \boldsymbol{s}_0 = \boldsymbol{s}\right]$$

- Recall **Q-function (state-action-value):**

$$Q^{\pi,\gamma}(\boldsymbol{s}, \boldsymbol{u}) = \mathbb{E}\left[\boldsymbol{r}_0 + \gamma \boldsymbol{r}_1 + \gamma^2 \boldsymbol{r}_2 + \cdots \mid \boldsymbol{s}_0 = \boldsymbol{s}, \boldsymbol{u}_0 = \boldsymbol{u}\right]$$

How can we improve the baseline choice ?

- Now we use : $b(s_t) = \mathbb{E}\left[r_t + r_{t+1} + r_{t+2} + \ldots + r_{H-1}\right]$

- Recall **State-value function** – can serve as a **baseline estimator**

$$V^{\pi,\gamma}(s) = \mathbb{E}_\pi\left[r_0 + \gamma r_1 + \gamma^2 r_2 \cdots \mid s_0 = s\right]$$

- Recall **Q-function (state-action-value):**

$$\begin{aligned}
Q^{\pi,\gamma}(s,u) &= \mathbb{E}[r_0 + \gamma r_1 + \gamma^2 r_2 + \cdots \mid s_0 = s, u_0 = u] \\
&= \mathbb{E}[r_0 + \gamma V^\pi(s_1) \mid s_0 = s, u_0 = u] \\
&= \mathbb{E}[r_0 + \gamma r_1 + \gamma^2 V^\pi(s_2) \mid s_0 = s, u_0 = u] \\
&= \mathbb{E}[r_0 + \gamma r_1 + \gamma^2 r_2 + \gamma^3 V^\pi(s_3) \mid s_0 = s, u_0 = u] \\
&= \cdots
\end{aligned}$$

How can we improve the baseline choice ?

- Current state:

$$\nabla U(\theta) \approx \hat{g} = \frac{1}{m} \sum_{i=1}^{m} \left(\sum_{t=0}^{H-1} \nabla_\theta \log \pi_\theta \left(u_t^{(i)} \mid s_t^{(i)} \right) \left(\sum_{k=t}^{H-1} R\left(s_k^{(i)}, u_k^{(i)} \right) - b(s_t) \right) \right)$$

How can we improve the baseline choice ?

- **Current state:**

$$\nabla U(\theta) \approx \hat{g} = \frac{1}{m} \sum_{i=1}^{m} \left(\sum_{t=0}^{H-1} \nabla_\theta \log \pi_\theta \left(u_t^{(i)} \mid s_t^{(i)} \right) \left(\sum_{k=t}^{H-1} R\left(s_k^{(i)}, u_k^{(i)} \right) - b(s_t) \right) \right)$$

Replacing with bootstrapping and action value estimation from the previous slide

$$\nabla U(\theta) \approx \hat{g} = \frac{1}{m} \sum_{i=1}^{m} \left(\sum_{t=0}^{H-1} \nabla_\theta \log \pi_\theta \left(u_t^{(i)} \mid s_t^{(i)} \right) \left(Q^{\pi,\gamma}(s_t, u_t) - V^{\pi,\gamma}(s_t) \right) \right)$$

Estimated value of taking action u_t in state s_t

Estimated value of state s_t

How can we improve the baseline choice ?

- Current state:

$$\nabla U(\theta) \approx \hat{g} = \frac{1}{m} \sum_{i=1}^{m} \left(\sum_{t=0}^{H-1} \nabla_\theta \log \pi_\theta \left(u_t^{(i)} \mid s_t^{(i)} \right) \left(\sum_{k=t}^{H-1} R\left(s_k^{(i)}, u_k^{(i)} \right) - b(s_t) \right) \right)$$

Replacing with bootstrapping and action value estimation from the previous slide

$$\nabla U(\theta) \approx \hat{g} = \frac{1}{m} \sum_{i=1}^{m} \left(\sum_{t=0}^{H-1} \nabla_\theta \log \pi_\theta \left(u_t^{(i)} \mid s_t^{(i)} \right) \left(Q^{\pi,\gamma}(s_t, u_t) - V^{\pi,\gamma}(s_t) \right) \right)$$

Estimated value of taking action u_t in state s_t

Estimated value of state s_t

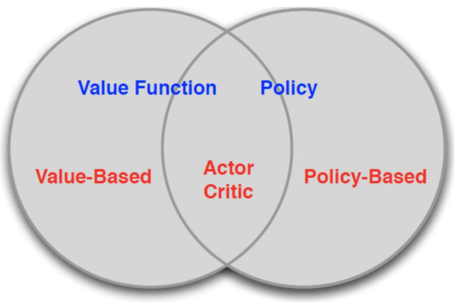
Replacing with the advantage function:

what is the advantage over average if agent takes action u_t in state s_t

$$\nabla U(\theta) \approx \hat{g} = \frac{1}{m} \sum_{i=1}^{m} \left(\sum_{t=0}^{H-1} \nabla_\theta \log \pi_\theta \left(u_t^{(i)} \mid s_t^{(i)} \right) A^{\pi,\gamma}(s_t, u_t) \right)$$

How can we improve the baseline choice ?

- Actor-Critic method:
  - Estimation of $V\ and\ Q$ done by a **critic**
  - Policy decisions are taken by an **actor**



- **A3C** (Mnih et al. ICML 2016) commonly used actor-critic method
  - **Critic** can select any blend between TD and MC estimators for estimating the state-action value function Q.

$$
\begin{aligned}
Q^{\pi,\gamma}(s,u) &= \mathbb{E}[r_0 + \gamma r_1 + \gamma^2 r_2 + \cdots \mid s_0 = s, u_0 = u] \\
&= \mathbb{E}[r_0 + \gamma V^{\pi}(s_1) \mid s_0 = s, u_0 = u] \\
&= \mathbb{E}[r_0 + \gamma r_1 + \gamma^2 V^{\pi}(s_2) \mid s_0 = s, u_0 = u] \\
&= \mathbb{E}[r_0 + \gamma r_1 + \quad \gamma^2 r_2 + \gamma^3 V^{\pi}(s_3) \mid s_0 = s, u_0 = u] \\
&= \cdots
\end{aligned}
$$

  - In A3C the look-ahead number of steps is a hyperparameter. E.g., k=5

How can we improve the baseline choice ?

Let’s work on multiple look-ahead steps

- Recall: $$\nabla U(\theta) \approx \hat{g} = \frac{1}{m} \sum_{i=1}^{m} \left(\sum_{t=0}^{H-1} \nabla_\theta \log \pi_\theta \left(u_t^{(i)} \mid s_t^{(i)} \right) A^{\pi,\gamma}(s_t, u_t) \right)$$

$$A^{\pi,\gamma}(s_t, u_t) = Q^{\pi,\gamma}(s_t, u_t) - V^{\pi,\gamma}(s_t)$$

How can we improve the baseline choice ?

Let's work on multiple look-ahead steps

- Recall: $\nabla U(\theta) \approx \hat{g} = \frac{1}{m} \sum_{i=1}^{m} \left(\sum_{t=0}^{H-1} \nabla_\theta \log \pi_\theta \left(u_t^{(i)} \mid s_t^{(i)} \right) A^{\pi,\gamma}(s_t, u_t) \right)$

$$A^{\pi,\gamma}(s_t, u_t) = Q^{\pi,\gamma}(s_t, u_t) - V^{\pi,\gamma}(s_t)$$

Simplifying notations (getting out π, γ) and explicating multiple steps inside Q

Look-ahead index

$$A_t^{(1)} = r_t + \gamma V(s_{t+1}) - V(s_t)$$

$$A_t^{(2)} = r_t + \gamma V(s_{t+1}) + \gamma^2 V(s_{t+2}) - V(s_t)$$

....

$$A_t^{(inf)} = r_t + \gamma V(s_{t+1}) + \gamma^2 V(s_{t+2}) + \ldots - V(s_t)$$

Check

Look-ahead index

$$A_t^{(1)} = r_t + \gamma V(s_{t+1}) - V(s_t)$$

$$A_t^{(2)} = r_t + \gamma V(s_{t+1}) + \gamma^2 V(s_{t+2}) - V(s_t)$$

....

$$A_t^{(inf)} = r_t + \gamma V(s_{t+1}) + \gamma^2 V(s_{t+2}) + \ldots - V(s_t)$$

Select which ones are true:

- ❑ $A_t^{(1)}$ has low variance and low bias
- ❑ $A_t^{(1)}$ has high variance and low bias
- ❑ $A_t^{(\text{inf})}$ has low variance and high bias
- ❑ $A_t^{(\text{inf})}$ has high variance and low bias

Check

Look-ahead index

$$A_t^{(1)} = r_t + \gamma V(s_{t+1}) - V(s_t)$$

$$A_t^{(2)} = r_t + \gamma V(s_{t+1}) + \gamma^2 V(s_{t+2}) - V(s_t)$$

....

$$A_t^{(inf)} = r_t + \gamma V(s_{t+1}) + \gamma^2 V(s_{t+2}) + \dots - V(s_t)$$

Select which ones are true:

- ❑ $A_t^{(1)}$ has low variance and low bias
- ❑ $A_t^{(1)}$ has high variance and low bias
- ❑ $A_t^{(\mathrm{inf})}$ has low variance and high bias
- ❑ $A_t^{(\mathrm{inf})}$ has high variance and low bias

Notes:

- ➢ $A_t^{(1)}$ has low variance but high bias !
- ➢ As we perform multiple look-ahead steps we increase the variance but lower the bias !

How can we improve the baseline choice ?

Let's improve further !

- **Generalized Advantage Estimation (GAE)** [Schulman et al, ICLR 2016]
- ~ TD(lambda) / eligibility traces [Sutton and Barto, 1990]

$$
\begin{aligned}
Q^{\pi,\gamma}(s,u) &= \mathbb{E}[r_0 + \gamma r_1 + \gamma^2 r_2 + \cdots \mid s_0 = s, u_0 = u] \\
&= \mathbb{E}[r_0 + \gamma V^\pi(s_1) \mid s_0 = s, u_0 = u] \\
&= \mathbb{E}[r_0 + \gamma r_1 + \gamma^2 V^\pi(s_2) \mid s_0 = s, u_0 = u] \\
&= \mathbb{E}[r_0 + \gamma r_1 + \quad \gamma^2 r_2 + \gamma^3 V^\pi(s_3) \mid s_0 = s, u_0 = u] \\
&= \cdots
\end{aligned}
$$

How can we improve the baseline choice ?

Let's improve further ! **Key Idea**: why not averaging all look-ahead steps in estimating Q ?

- **Generalized Advantage Estimation (GAE)** [Schulman et al, ICLR 2016]
- ~ TD(lambda) / eligibility traces [Sutton and Barto, 1990]

$$
\begin{aligned}
Q^{\pi,\gamma}(s,u) &= \mathbb{E}[r_0 + \gamma r_1 + \gamma^2 r_2 + \cdots \mid s_0 = s, u_0 = u] && (1-\lambda) \\
&= \mathbb{E}[r_0 + \gamma V^\pi(s_1) \mid s_0 = s, u_0 = u] && (1-\lambda)\lambda \\
&= \mathbb{E}[r_0 + \gamma r_1 + \gamma^2 V^\pi(s_2) \mid s_0 = s, u_0 = u] && (1-\lambda)\lambda^2 \\
&= \mathbb{E}[r_0 + \gamma r_1 + \gamma^2 r_2 + \gamma^3 V^\pi(s_3) \mid s_0 = s, u_0 = u] && (1-\lambda)\lambda^3 \\
&= \cdots
\end{aligned}
$$

$\Rightarrow$ $\hat{Q}$ Averaged weighted of all steps

Actor-Critic with A3C + GAE

- Use two networks: one for policy, one for value estimation
- **Note**: Can update both independently, e.g., can use k-steps for V, full roll-out for π

Init π_{θ_0}, $V^{\pi}_{\phi_0}$

Run episode = 1,2,....

Collect roll-outs $\{\mathrm{s}, \mathrm{u}, \mathrm{s}', \mathrm{r}\}$

Estimate $\hat{Q}_i(s, u)$ using GAE

Update the two networks:

$$\phi_{i+1} \leftarrow \min_{\phi} \sum_{(s,u,s',r)} \left\| \hat{Q}_i(s,u) - V^{\pi}_{\phi}(s) \right\|_2^2 + \kappa \|\phi - \phi_i\|_2^2$$

$$\theta_{i+1} \leftarrow \theta_i + \alpha \frac{1}{m} \sum_{k=1}^{m} \sum_{t=0}^{H-1} \nabla_\theta \log \pi_{\theta_i}\left(u_t^{(k)} \mid s_t^{(k)}\right) \underbrace{\left(\hat{Q}_i\left(s_t^{(k)}, u_t^{(k)}\right) - V^{\pi}_{\phi_i}\left(s_t^{(k)}\right)\right)}_{\hat{A}_t}$$

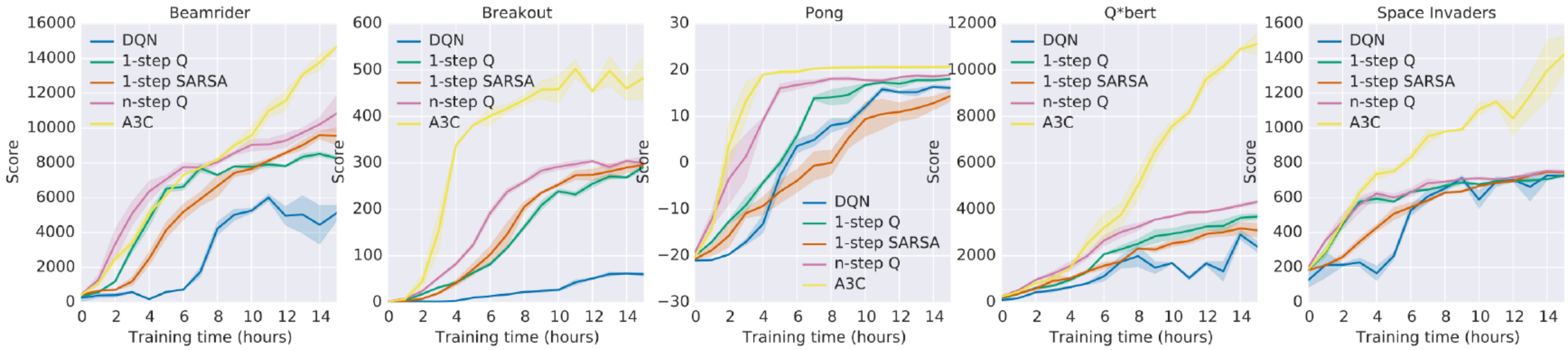
(advantage at time t)

Some results on ATARI games:

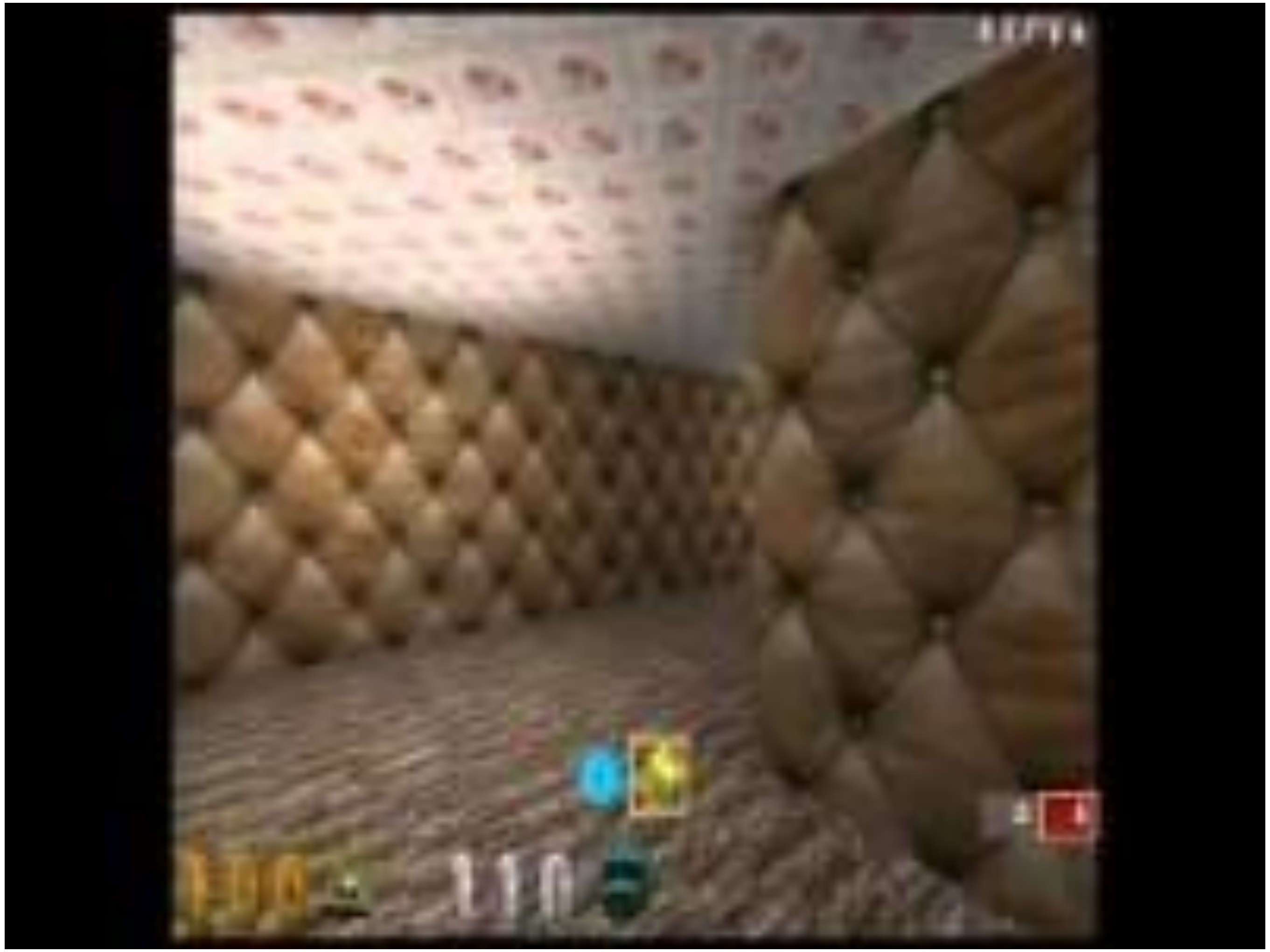
[Mnih et al, ICML 2016]

Likelihood Ratio Policy Gradient

n-step Advantage Estimation



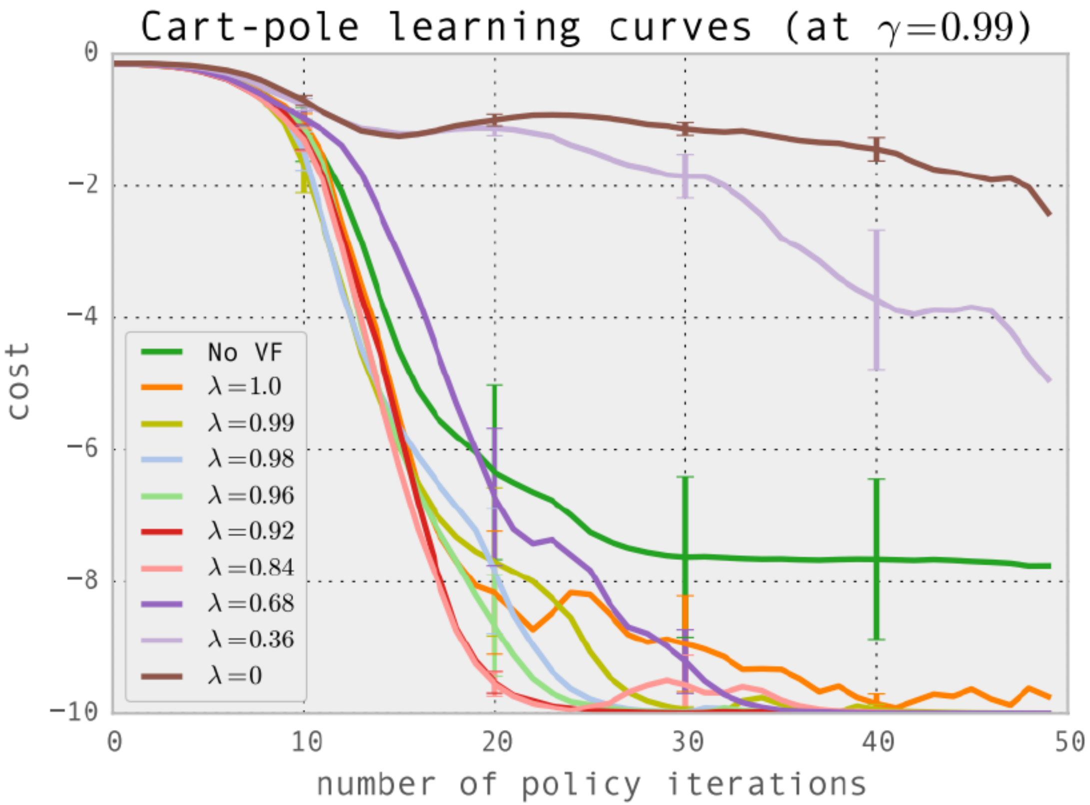
DeepMine A3C Labyrinth

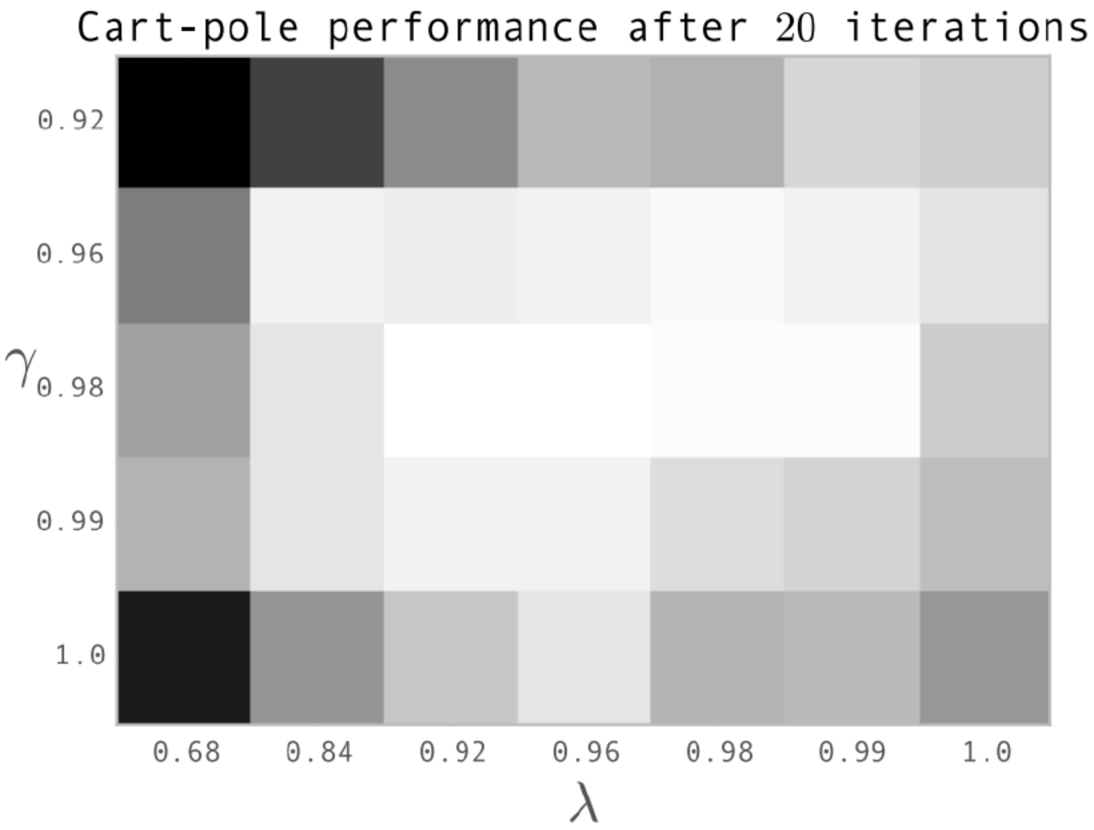


The video shows an agent collecting rewards in previously unseen mazes using only raw pixels as input. The agent was trained using the Asynchronous Advantage Actor-Critic (A3C) algorithm and was only rewarded for picking up apples and orange portals during training.

Paper link - http://arxiv.org/pdf/1602.01783.pdf

Cart-Pole, GAE: Effect of gamma and lambda





[Schulman et al, 2016 -- GAE]

Thank you !
Questions