Policy Search

Roberto Capobianco

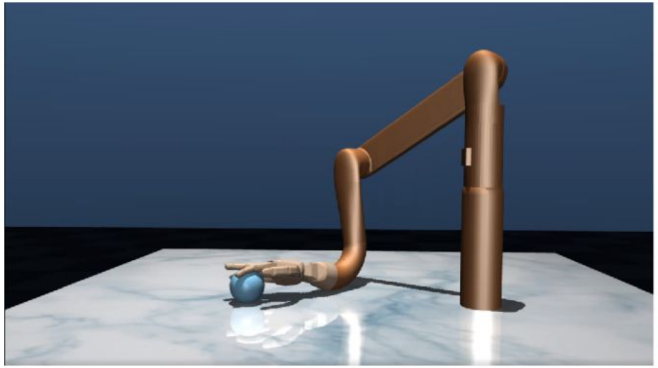
Sapienza Università di Roma

Recap

Model-Based RL: Motivation

We cannot write out the exact analytical dynamics, but we can learn it from data {s,a,s’}

And then find a policy by planning on such dynamic model



Basic Algorithm

The simplest algorithm is the following:

1. Generate data

   (e.g., execute a starting policy)

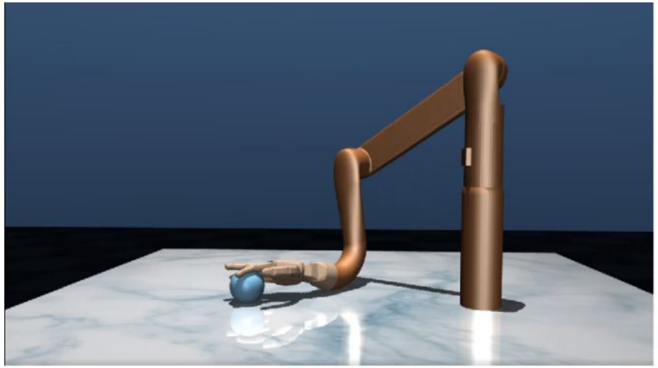
2. Fit a model using data

   (e.g., using least-squares, or maximum likelihood)

3. Plan on the learned model

   (e.g., using VI, PI, or LQR)

Often iterate this process several times



Simulation Lemma

Question: If I have an estimator of the true dynamics, what's the performance of a policy under this estimator, wrt the real world?

$$\widehat{V}^{\pi}(s_0) = \mathbb{E}\left[\sum_{h=0}^{\infty} \gamma^h r(s_h, a_h) \mid \pi, \widehat{P}\right]$$

value of policy in the simulator

$$V^{\pi}(s_0) = \mathbb{E}\left[\sum_{h=0}^{\infty} \gamma^h r(s_h, a_h) \mid \pi, P\right]$$

value of policy in the true dynamics

$$\widehat{V}^{\pi}(s_0) - V^{\pi}(s_0) = \frac{\gamma}{1-\gamma}\mathbb{E}_{s,a\sim d^{\pi}_{s_0}}\left[\mathbb{E}_{s'\sim \widehat{P}(s,a)}\widehat{V}^{\pi}(s') - \mathbb{E}_{s'\sim P(s,a)}\widehat{V}^{\pi}(s')\right]$$

Simulation Lemma

$$\widehat{V}^{\pi}(s_0) = \mathbb{E}\left[\sum_{h=0}^{\infty} \gamma^h r(s_h, a_h) \mid \pi, \widehat{P}\right]$$

value of policy in the simulator

$$V^{\pi}(s_0) = \mathbb{E}\left[\sum_{h=0}^{\infty} \gamma^h r(s_h, a_h) \mid \pi, P\right]$$

value of policy in the true dynamics

Question: If I have an estimator of the true dynamics, what’s the performance of a policy under this estimator, wrt the real world?

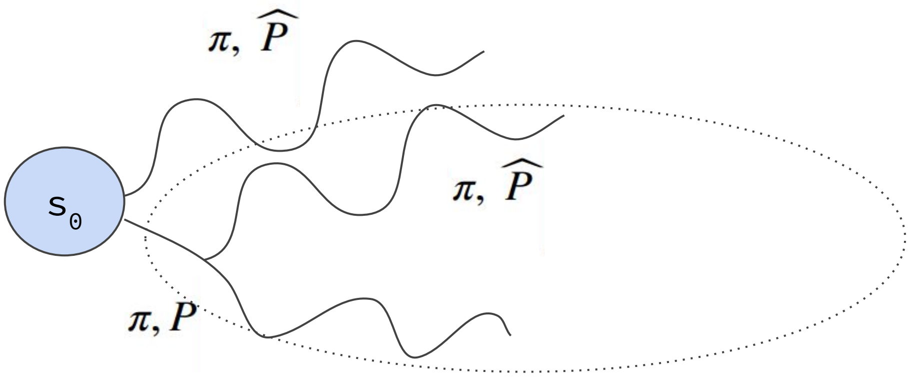
$$\widehat{V}^{\pi}(s_0) - V^{\pi}(s_0) = \frac{\gamma}{1-\gamma}\mathbb{E}_{s,a\sim d^{\pi}_{s_0}}\left[\mathbb{E}_{s'\sim \widehat{P}(s,a)} \widehat{V}^{\pi}(s') - \mathbb{E}_{s'\sim P(s,a)} \widehat{V}^{\pi}(s')\right]$$

Computing this is very difficult, but we can do it one step at a time.

At a single step the action distribution is the same, the only difference, is in the next state:

Let’s step in the real dynamics for one step, and then go back to the simulator.

We can do recursion and follow the same reasoning again.



Simulation Lemma

$$\widehat{V}^{\pi}(s_0) = \mathbb{E}\left[\sum_{h=0}^{\infty} \gamma^h r(s_h, a_h) \mid \pi, \widehat{P}\right]$$

value of policy in the simulator

$$V^{\pi}(s_0) = \mathbb{E}\left[\sum_{h=0}^{\infty} \gamma^h r(s_h, a_h) \mid \pi, P\right]$$

value of policy in the true dynamics

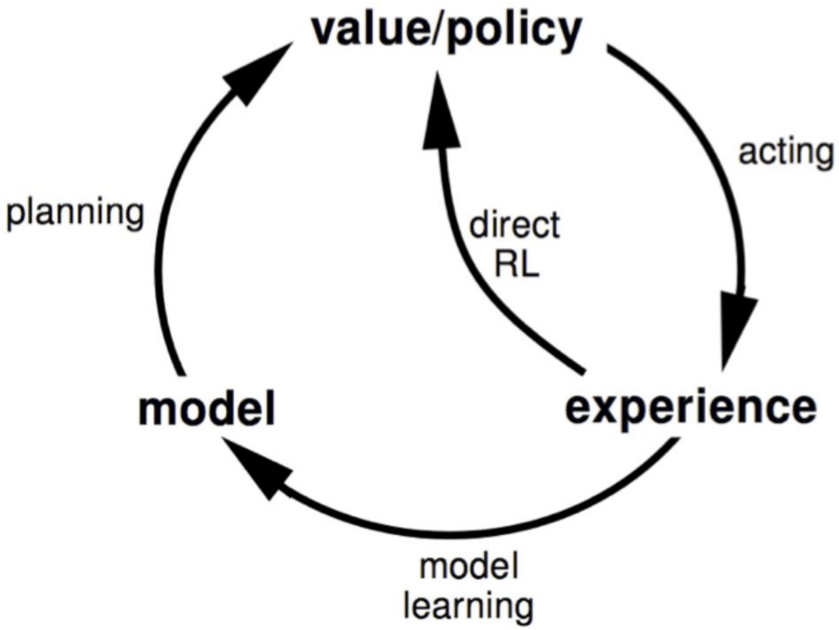
Question: If I have an estimator of the true dynamics, what’s the performance of a policy under this estimator, wrt the real world?

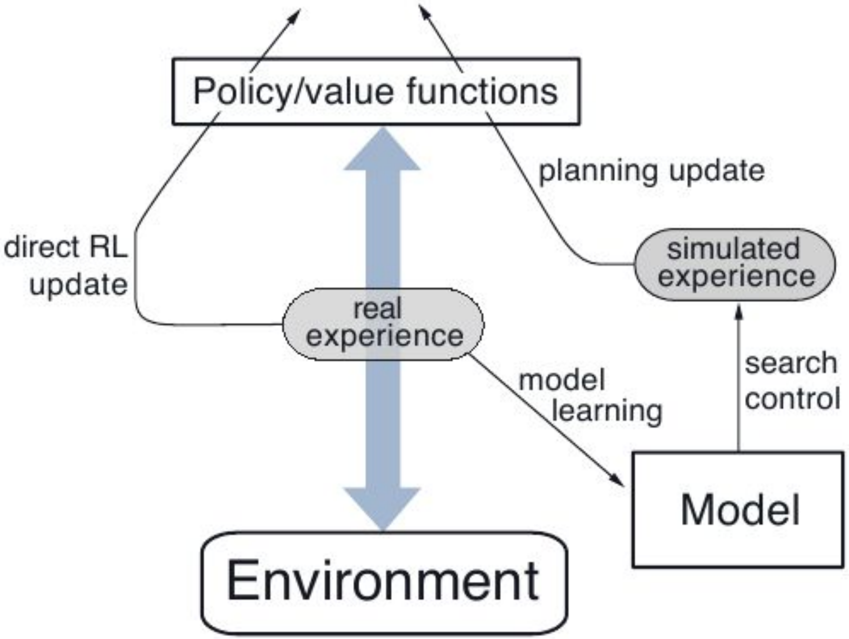
$$\widehat{V}^{\pi}(s_0) - V^{\pi}(s_0) = \frac{\gamma}{1-\gamma}\mathbb{E}_{s,a\sim d^{\pi}_{s_0}}\left[\mathbb{E}_{s'\sim \widehat{P}(s,a)} \widehat{V}^{\pi}(s') - \mathbb{E}_{s'\sim P(s,a)} \widehat{V}^{\pi}(s')\right]$$

$$\leq \frac{1}{(1-\gamma)^2}\mathbb{E}_{s,a\sim d^{\pi}_{s_0}} \left\| \widehat{P}(\cdot \mid s,a) - P(\cdot \mid s,a)\right\|_1$$

We can bound the policy performance difference by the total model disagreement measured on the real trajectory

Full Model-Based RL Loop





Model Fitting

How can we fit a model?

For example, very simply, collect N data-points and estimate it as follows (note that we're using the indicator function **1**)

$$\widehat{P}(s'|s,a) = \frac{\sum_{i=1}^{N} \mathbf{1}\{s_i' = s'\}}{N}$$

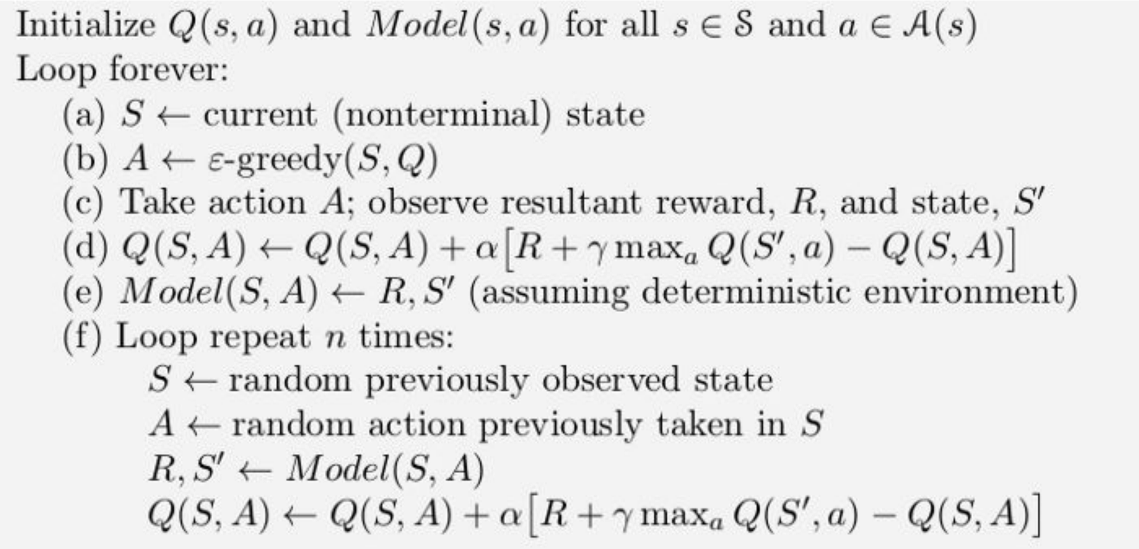
At infinity this should converge to the true P

How can we plan using a model?

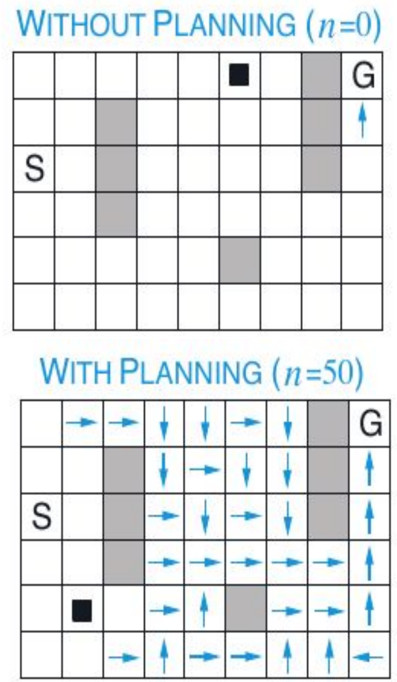
Use value iteration, policy iteration, LQR if we're in continuous space, or other solutions like:

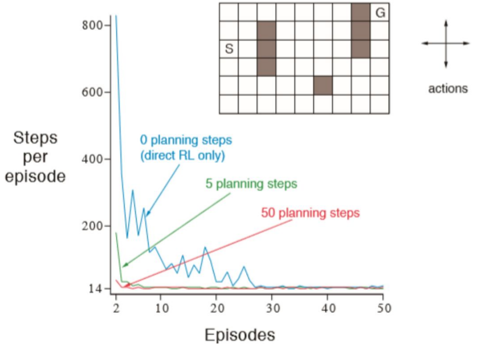
- Q-planning
- Monte-Carlo Tree Search

Dyna-Q



```
Initialize $Q(s,a)$ and $Model(s,a)$ for all $s \in \mathcal{S}$ and $a \in \mathcal{A}(s)$
Loop forever:
    (a) $S \leftarrow$ current (nonterminal) state
    (b) $A \leftarrow \varepsilon$-greedy$(S,Q)$
    (c) Take action $A$; observe resultant reward, $R$, and state, $S'$
    (d) $Q(S,A) \leftarrow Q(S,A) + \alpha\left[R + \gamma \max_a Q(S',a) - Q(S,A)\right]$
    (e) $Model(S,A) \leftarrow R, S'$ (assuming deterministic environment)
    (f) Loop repeat $n$ times:
            $S \leftarrow$ random previously observed state
            $A \leftarrow$ random action previously taken in $S$
            $R, S' \leftarrow Model(S,A)$
            $Q(S,A) \leftarrow Q(S,A) + \alpha\left[R + \gamma \max_a Q(S',a) - Q(S,A)\right]$
```



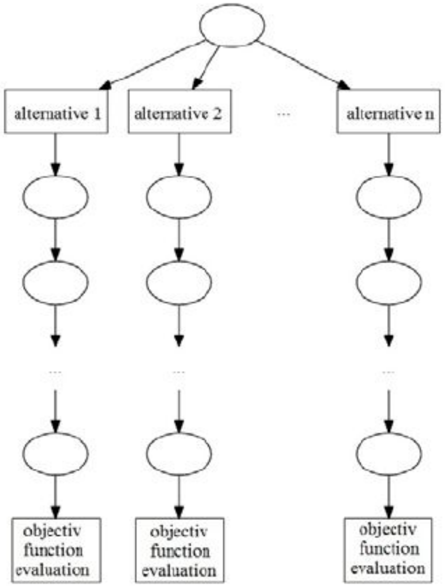


Rollout Planning Algorithm

Decision-time planning

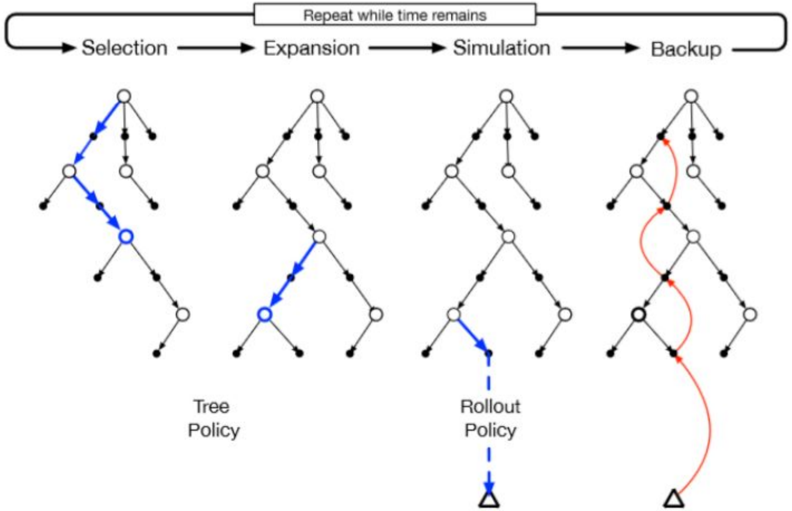
- Uses MC control applied to simulated trajectories starting at current state
- Estimate action values by averaging returns of many simulated trajectories: try each possible action for one step and then follow rollout policy
- When estimate accurate, highest value action is executed

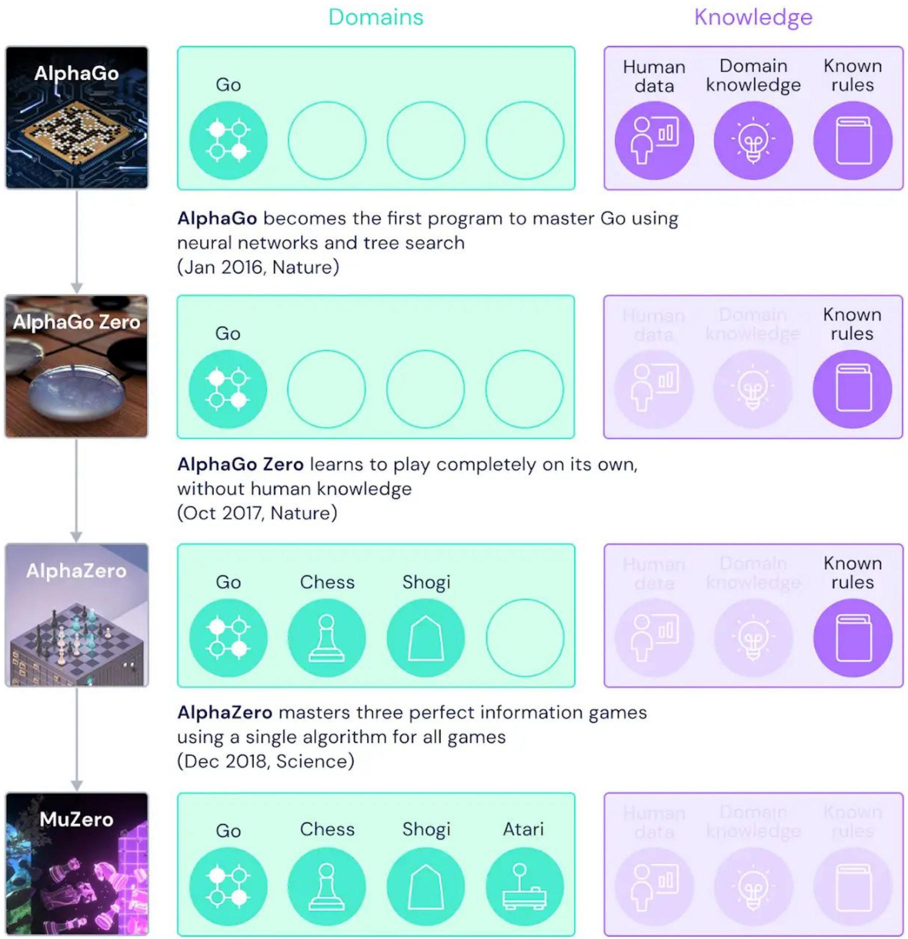
Does not estimate (unlike MC) full value-function, but only value of actions for current state and given policy


alternative 1
alternative 2
...
alternative n
...
objectiv function evaluation
objectiv function evaluation
objectiv function evaluation

Monte-Carlo Tree Search

Decision-time planning, like a rollout algorithm BUT accumulating value estimates





AlphaGo becomes the first program to master Go using neural networks and tree search
(Jan 2016, Nature)

AlphaGo Zero learns to play completely on its own, without human knowledge
(Oct 2017, Nature)

AlphaZero masters three perfect information games using a single algorithm for all games
(Dec 2018, Science)

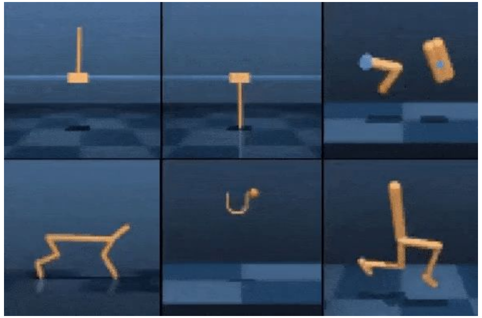
MuZero learns the rules of the game, allowing it to also master environments with unknown dynamics.
(Dec 2020, Nature)

Learning Dynamics from Pixels

Hafner, Danijar, et al. "Learning latent dynamics for planning from pixels." *International Conference on Machine Learning*. PMLR, 2019.

- Not always the state is available (POMDP): learn a compact representation
  - A recurrent model is needed
- Plan in the learned (latent!) dynamics space

$$\begin{aligned}
&\text{Transition function:} && s_t \sim \mathrm{p}(s_t \mid s_{t-1}, a_{t-1}) \\
&\text{Observation function:} && o_t \sim \mathrm{p}(o_t \mid s_t) \\
&\text{Reward function:} && r_t \sim \mathrm{p}(r_t \mid s_t) \\
&\text{Policy:} && a_t \sim \mathrm{p}(a_t \mid o_{\leqslant t}, a_{<t}),
\end{aligned} \tag{1}$$



Narrowing the Reality Gap

- Fine-tuning
- Progressive nets
- Improved modeling
  - System identification, better models, etc.
- Randomization
  - Random perturbations, Randomization in dynamics parameters

End Recap

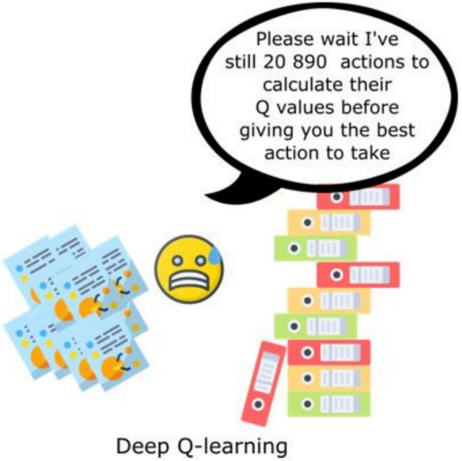
Policy from Value

So far, we derived a policy out of a tabular or parameterized state-action value function

Policy from Value: Problems (1)

So far, we derived a policy out of a tabular or parameterized state-action value function

Computing this is very difficult if the action space is large


Please wait I've still 20 890 actions to calculate their Q values before giving you the best action to take
Deep Q-learning

Policy from Value: Problems (2)

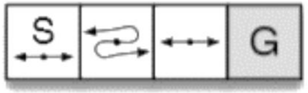
So far, we derived a policy out of a tabular or parameterized state-action value function

Reward: -1 per step

Actions: left, right

Action effects are as usual in first and third states, reversed in second state

All states appear identical in their featurization x(s,right)=[1,0] and x(s,left)=[0,1]

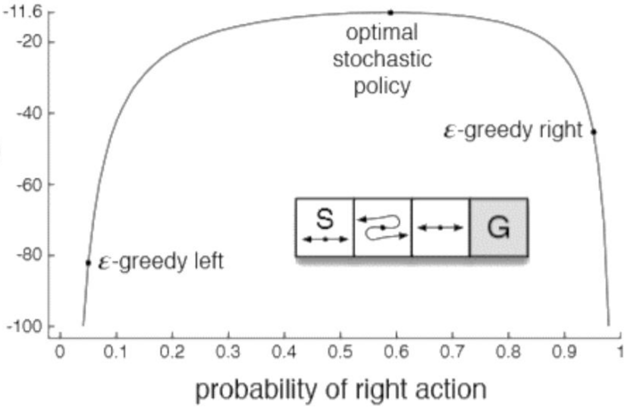


Policy from Value: Problems (2)

So far, we derived a policy out of a tabular or parameterized state-action value function

The optimal policy is not always deterministic and eps-greedy is not enough

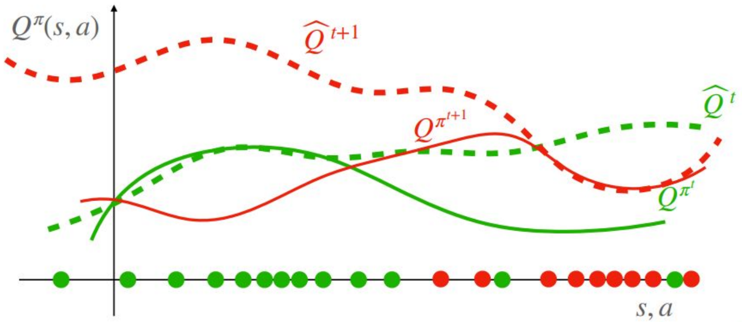
$J(\boldsymbol{\theta}) = v_{\pi_{\boldsymbol{\theta}}}(\mathsf{S})$



Policy from Value: Problems (3)

So far, we derived a policy out of a tabular or parameterized state-action value function

Dramatic policy oscillations due to small value changes



Green dots: (s, a) from π^t

Red dots: (s, a) from π^{t+1}

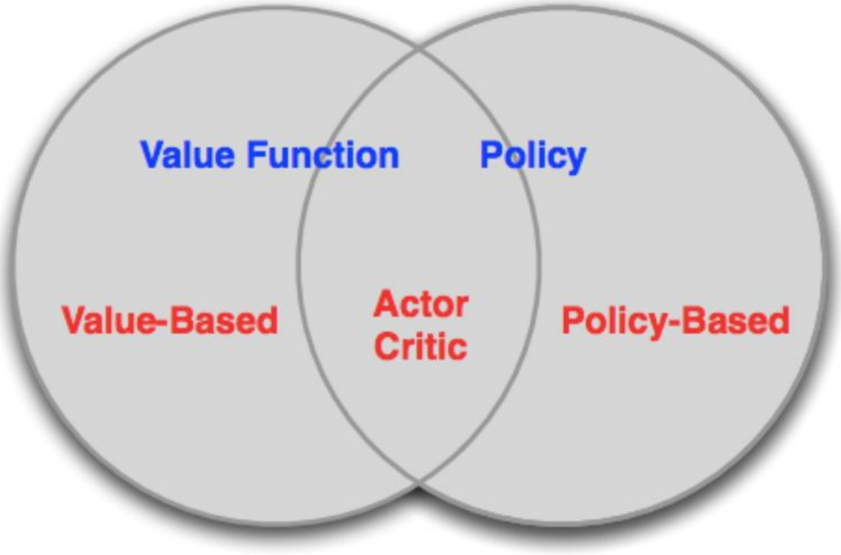
Credits: Wen Sun

Parameterized Policy

Can we directly represent and learn a parameterized policy?

$$a \sim \pi_{\boldsymbol{\theta}}(s) \text{ or } p(.|s,\boldsymbol{\theta})$$

Policy-Based RL



Credits: David Silver

Finding a Parameterized Policy

How do we search for a parameterized policy π_θ(s)?

We want to find the parameters θ such that π_θ is the best policy

Finding a Parameterized Policy

How do we search for a parameterized policy π_θ(s)?

We want to find the parameters θ such that π_θ is the best policy

How do we measure the quality for a policy?

Quality of a Parameterized Policy

How do we measure the quality for a policy?

In other words, what is our objective function to maximize through the policy parameters?

Quality of a Parameterized Policy

How do we measure the quality for a policy?

In other words, what is our objective function to maximize through the policy parameters?

- For episodic tasks:
  - Value at start state (undiscounted)
- For continuing tasks:
  - Value at start state (discounted)
  - Average value

    see [Sutton&Barto 10.4: Deprecating the Discounted Setting]

  - Average reward per timestep

    see [Sutton&Barto 10.4: Deprecating the Discounted Setting]

$$J(\pi) := \mathbb{E}\left[\sum_{h=0}^{H-1} r(s_h, a_h) \,\middle|\, s_0 \sim \mu, s_{h+1} \sim P_{s_h,a_h}, a_h \sim \pi(\cdot \mid s_h)\right]$$

$$J(\pi) := \mathbb{E}\left[\sum_{h=0}^{\infty} \gamma^h r(s_h, a_h) \,\middle|\, s_0 \sim \mu, s_{h+1} \sim P_{s_h,a_h}, a_h \sim \pi(\cdot \mid s_h)\right]$$

Policy Optimization

Can use gradient free optimization

- Hill climbing Simplex / Nelder Mead Genetic algorithms
- Cross-Entropy method (CEM)
- Covariance Matrix Adaptation (CMA)

Policy Optimization

Can use gradient free optimization

- Hill climbing Simplex / Nelder Mead Genetic algorithms
- Cross-Entropy method (CEM)
- Covariance Matrix Adaptation (CMA)

Evolution strategies can rival the performance of standard RL techniques on modern RL benchmarks (e.g. Atari/MuJoCo): https://openai.com/blog/evolution-strategies/

Policy Optimization

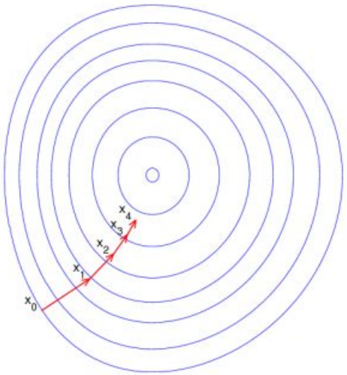
Can use gradient free optimization

- Hill climbing Simplex / Nelder Mead Genetic algorithms
- Cross-Entropy method (CEM)
- Covariance Matrix Adaptation (CMA)

Can work with any policy parameterizations, including non-differentiable ones, but can be sample inefficient and requires higher outcome variability

Policy Gradient

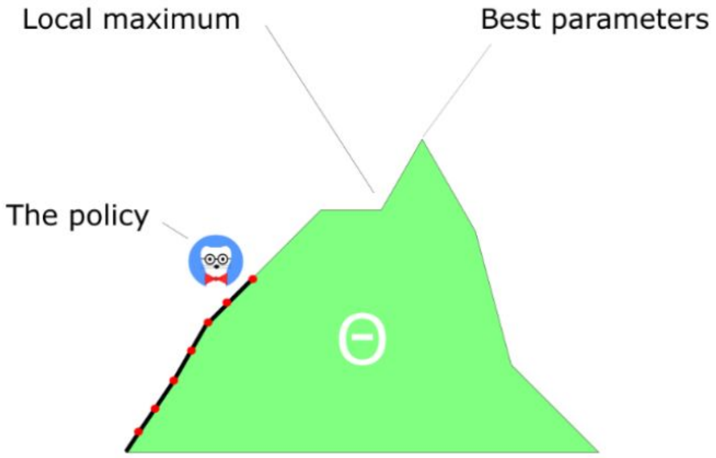
An efficient and widely used technique to find a parameterized policy is the policy gradient



Policy Gradient

An efficient and widely used technique to find a parameterized policy is the policy gradient

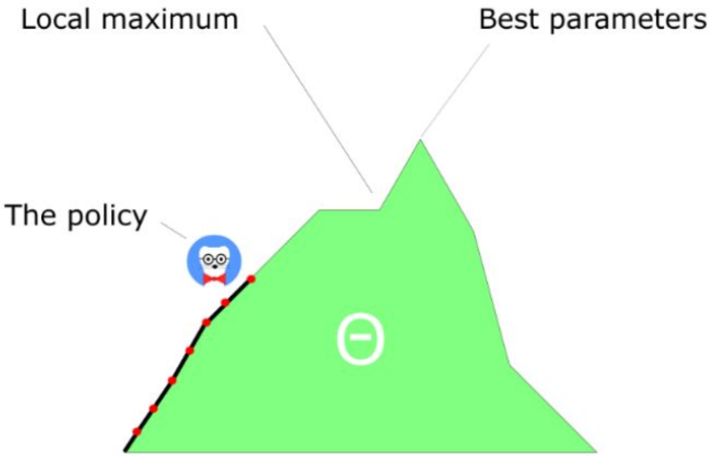
Guaranteed to converge to local maximum or global maximum, *but it often converges only to a local maximum*



Policy Gradient

An efficient and widely used technique to find a parameterized policy is the policy gradient

Only for convex function, this guarantees global optimality



Policy Gradient

$$\pi_\theta(a \mid s) = \pi(a \mid s; \theta) \qquad J(\pi_\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{h=0}^{\infty} \gamma^h r_h\right]$$

$$\theta_{t+1} = \theta_t + \eta \nabla_\theta J(\pi_\theta)\,|_{\theta=\theta_t}$$

Policy Gradient

$$\pi_\theta(a \mid s) = \pi(a \mid s; \theta) \qquad J(\pi_\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{h=0}^{\infty} \gamma^h r_h\right]$$

$$\theta_{t+1} = \theta_t + \eta \boxed{\nabla_\theta J(\pi_\theta)\,|_{\theta=\theta_t}}$$

HOW?

Finite Differences

$$\pi_\theta(a \mid s) = \pi(a \mid s; \theta) \qquad J(\pi_\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{h=0}^{\infty} \gamma^h r_h\right]$$

$$\theta_{t+1} = \theta_t + \eta \nabla_\theta J(\pi_\theta)\,|_{\theta=\theta_t}$$

Simple approach: compute $\nabla_\theta J(\pi_\theta)$ using finite differences

$$J(\pi_\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{h=0}^{\infty} \gamma^h r_h\right]$$

Finite Differences

$$\pi_\theta(a \mid s) = \pi(a \mid s; \theta) \qquad J(\pi_\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{h=0}^{\infty} \gamma^h r_h\right]$$

$$\theta_{t+1} = \theta_t + \eta \nabla_\theta J(\pi_\theta)\,|_{\theta=\theta_t}$$

Simple approach: compute $\nabla_\theta J(\pi_\theta)$ using finite differences

$$J(\pi_\theta) = \boxed{\mathbb{E}_{\pi_\theta}\left[\sum_{h=0}^{\infty} \gamma^h r_h\right]}$$

Finite Differences

$\pi_\theta(a \mid s) = \pi(a \mid s; \theta)$ $\quad J(\pi_\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{h=0}^{\infty} \gamma^h r_h\right]$

$\theta_{t+1} = \theta_t + \eta \nabla_\theta J(\pi_\theta)|_{\theta=\theta_t}$

Simple approach: compute $\nabla_\theta J(\pi_\theta)$ using finite differences

$$J(\pi_\theta) = \boxed{\mathbb{E}_{\pi_\theta}\left[\sum_{h=0}^{\infty} \gamma^h r_h\right]}$$

$V^\pi(s_0)$!

Finite Differences

$$\pi_\theta(a \mid s) = \pi(a \mid s; \theta) \qquad J(\pi_\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{h=0}^{\infty} \gamma^h r_h\right]$$

$$\theta_{t+1} = \theta_t + \eta \nabla_\theta J(\pi_\theta)\,|_{\theta=\theta_t}$$

Simple approach: compute $\nabla_\theta J(\pi_\theta)$ using finite differences

$$J(\pi_\theta) = \boxed{\mathbb{E}_{\pi_\theta}\left[\sum_{h=0}^{\infty} \gamma^h r_h\right]}$$

$V^\pi(s_0)$!

Finite Differences

$$\pi_\theta(a \mid s) = \pi(a \mid s; \theta) \qquad J(\pi_\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{h=0}^{\infty} \gamma^h r_h\right]$$

$$\theta_{t+1} = \theta_t + \eta \nabla_\theta J(\pi_\theta)\,|_{\theta=\theta_t}$$

Simple approach: compute $\nabla_\theta J(\pi_\theta)$ using finite differences

IDEA: perturb θ by small amount in k-th dimension

$$\frac{\partial V(s_0, \theta)}{\partial \theta_k} \approx \frac{V(s_0, \theta + \epsilon u_k) - V(s_0, \theta)}{\epsilon}$$

Finite Differences

$$\pi_\theta(a\,|\,s) = \pi(a\,|\,s;\theta) \qquad J(\pi_\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{h=0}^{\infty} \gamma^h r_h\right]$$

$$\theta_{t+1} = \theta_t + \eta \nabla_\theta J(\pi_\theta)\,|_{\theta=\theta_t}$$

Simple approach: compute $\nabla_\theta J(\pi_\theta)$ using finite differences

IDEA: perturb θ by small amount in k-th dimension

$$\frac{\partial V(s_0,\theta)}{\partial \theta_k} \approx \frac{V(s_0,\theta + \epsilon \boxed{u_k}) - V(s_0,\theta)}{\epsilon}$$

$$\begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \begin{matrix} \\ \\ \\ \\ \leftarrow \\ \\ \\ \\ \end{matrix}$$

k-th entry

Finite Differences

$$\pi_\theta(a \mid s) = \pi(a \mid s; \theta) \qquad J(\pi_\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{h=0}^{\infty} \gamma^h r_h\right]$$

$$\theta_{t+1} = \theta_t + \eta \nabla_\theta J(\pi_\theta)\,|_{\theta=\theta_t}$$

Simple approach: compute $\nabla_\theta J(\pi_\theta)$ using finite differences

IDEA: perturb θ by small amount in k-th dimension

$$\frac{\partial V(s_0, \theta)}{\partial \theta_k} \approx \frac{V(s_0, \theta + \epsilon \boxed{u_k}) - V(s_0, \theta)}{\epsilon}$$

$$\begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \leftarrow$$

k-th entry

Uses n evaluations to compute policy gradient in n dimensions

Finite Differences

$$\pi_\theta(a \mid s) = \pi(a \mid s; \theta) \qquad J(\pi_\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{h=0}^{\infty} \gamma^h r_h\right]$$

$$\theta_{t+1} = \theta_t + \eta \nabla_\theta J(\pi_\theta)\,|_{\theta=\theta_t}$$

Simple approach: compute $\nabla_\theta J(\pi_\theta)$ using finite differences

IDEA: perturb θ by small amount in k-th dimension

$$\frac{\partial V(s_0, \theta)}{\partial \theta_k} \approx \frac{V(s_0, \theta + \epsilon \boxed{u_k}) - V(s_0, \theta)}{\epsilon}$$

$$\begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \leftarrow$$

k-th entry

Uses n evaluations to compute policy gradient in n dimensions

Works for arbitrary policies, even if policy is not differentiable

Finite Differences

$$\pi_\theta(a \mid s) = \pi(a \mid s; \theta) \qquad J(\pi_\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{h=0}^{\infty} \gamma^h r_h\right]$$

$$\theta_{t+1} = \theta_t + \eta \nabla_\theta J(\pi_\theta)|_{\theta=\theta_t}$$

Simple approach: compute $\nabla_\theta J(\pi_\theta)$ using finite differences

IDEA: perturb θ by small amount in k-th dimension

$$\frac{\partial V(s_0,\theta)}{\partial \theta_k} \approx \frac{V(s_0, \theta + \epsilon \boxed{u_k}) - V(s_0,\theta)}{\epsilon}$$

$$\begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \leftarrow$$

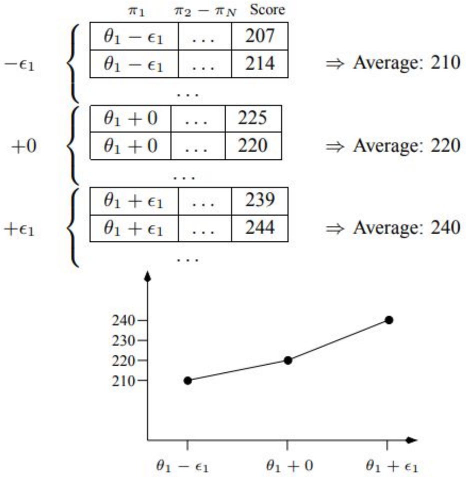
k-th entry

Uses n evaluations to compute policy gradient in n dimensions

Noise can dominate, but can be reduced by averaging over many samples or by reducing randomness if possible

Finite Differences: Example

Policy Gradient Reinforcement Learning for Fast Quadrupedal Locomotion. Nate Kohl and Peter Stone. Proceedings of the IEEE International Conference on Robotics and Automation, pp. 2619--2624, May 2004.



	π_1	$\pi_2 - \pi_N$	Score	
$-\epsilon_1$	$\theta_1 - \epsilon_1$	...	207	
	$\theta_1 - \epsilon_1$	...	214	⇒ Average: 210
	...			
$+0$	$\theta_1 + 0$	...	225	
	$\theta_1 + 0$	...	220	⇒ Average: 220
	...			
$+\epsilon_1$	$\theta_1 + \epsilon_1$	...	239	
	$\theta_1 + \epsilon_1$	...	244	⇒ Average: 240
	...			

```
$\pi \leftarrow InitialPolicy$
while !done do
    $\{R_1, R_2, \ldots, R_t\} = t$ random perturbations of $\pi$
    evaluate( $\{R_1, R_2, \ldots, R_t\}$ )
    for $n = 1$ to $N$ do
        $Avg_{+\epsilon,n} \leftarrow$ average score for all $R_i$ that have a positive
                perturbation in dimension $n$
        $Avg_{+0,n} \leftarrow$ average score for all $R_i$ that have a zero
                perturbation in dimension $n$
        $Avg_{-\epsilon,n} \leftarrow$ average score for all $R_i$ that have a
                negative perturbation in dimension $n$
        if $Avg_{+0,n} > Avg_{+\epsilon,n}$ and $Avg_{+0,n} > Avg_{-\epsilon,n}$ then
            $A_n \leftarrow 0$
        else
            $A_n \leftarrow Avg_{+\epsilon,n} - Avg_{-\epsilon,n}$
        end if
    end for
    $A \leftarrow \frac{A}{|A|} * \eta$
    $\pi \leftarrow \pi + A$
end while
```

Policy Gradient

$$\pi_\theta(a \mid s) = \pi(a \mid s; \theta) \qquad J(\pi_\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{h=0}^{\infty} \gamma^h r_h\right]$$

$$\theta_{t+1} = \theta_t + \eta \nabla_\theta J(\pi_\theta)\,|_{\theta=\theta_t}$$

Can we compute the gradient analytically?

Differentiable Policy Classes

Softmax Linear Policy

$$\pi_\theta(a \mid s) = \frac{\exp(\theta^\top \phi(s, a))}{\sum_{a'} \exp(\theta^\top \phi(s, a'))}$$

Differentiable Policy Classes

Softmax Linear Policy

$$\pi_\theta(a \mid s) = \frac{\exp(\theta^\top \phi(s,a))}{\sum_{a'} \exp(\theta^\top \boxed{\phi(s,a')})}$$

feature vector

Differentiable Policy Classes

Softmax Linear Policy

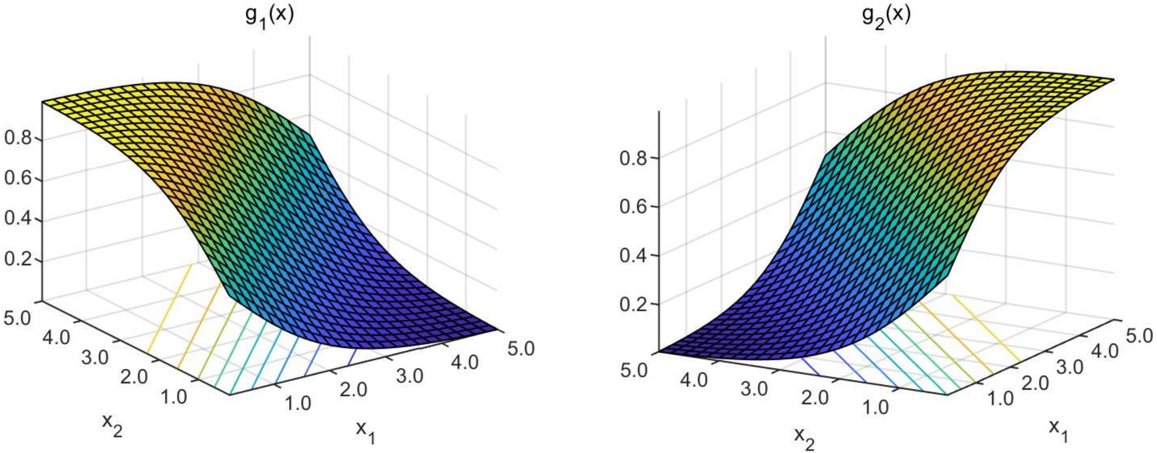
$$\pi_\theta(a \mid s) = \frac{\exp(\theta^\top \phi(s,a))}{\sum_{a'} \exp(\boxed{\theta^\top} \phi(s,a'))}$$

parameters

Differentiable Policy Classes

Softmax Linear Policy

$$\pi_\theta(a \mid s) = \frac{\exp(\theta^\top \phi(s,a))}{\sum_{a'} \exp(\theta^\top \phi(s,a'))}$$



Can be thought of as a classifier for discrete actions

Differentiable Policy Classes

Softmax Linear Policy

$$\pi_\theta(a \mid s) = \frac{\exp(\theta^\top \phi(s,a))}{\sum_{a'} \exp(\theta^\top \phi(s,a'))}$$

Softmax Policy

$$\pi_\theta(a \mid s) = \frac{\exp(f_\theta(s,a))}{\sum_{a'} \exp(f_\theta(s,a'))}$$

f can be a neural network

Differentiable Policy Classes

Softmax Linear Policy

$$\pi_\theta(a \mid s) = \frac{\exp(\theta^\top \phi(s,a))}{\sum_{a'} \exp(\theta^\top \phi(s,a'))}$$

Softmax Policy

$$\pi_\theta(a \mid s) = \frac{\exp(f_\theta(s,a))}{\sum_{a'} \exp(f_\theta(s,a'))}$$

Gaussian Policy

$$\pi_\theta(a \mid s) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(a - f_\theta(s))^2}{2\sigma^2}\right)$$

f can be a neural network

Likelihood Ratio Policy Gradient

Suppose we have a trajectory $\tau = \{s_0, a_0, s_1, a_1, \ldots\}$

it’s probability distribution, depending on θ is

$$\rho_\theta(\tau) = \mu(s_0)\pi_\theta(a_0 \mid s_0)P(s_1 \mid s_0, a_0)\pi_\theta(a_1 \mid s_1)\ldots$$

Likelihood Ratio Policy Gradient

Suppose we have a trajectory $\tau = \{s_0, a_0, s_1, a_1, \dots\}$

it’s probability distribution, depending on θ is

$$\rho_\theta(\tau) = \mu(s_0)\pi_\theta(a_0 \mid s_0)P(s_1 \mid s_0, a_0)\pi_\theta(a_1 \mid s_1)\dots$$

$$J(\pi_\theta) = \mathbb{E}_{\tau\sim\rho_\theta(\tau)} \underbrace{\left[\sum_{h=0}^{\infty} \gamma^h r(s_h, a_h)\right]}_{R(\tau)} = \sum_\tau P(\tau;\theta)R(\tau)$$

Likelihood Ratio Policy Gradient

Suppose we have a trajectory $\tau = \{s_0, a_0, s_1, a_1, \dots\}$

it’s probability distribution, depending on θ is

$$\rho_\theta(\tau) = \mu(s_0)\pi_\theta(a_0 \mid s_0)P(s_1 \mid s_0, a_0)\pi_\theta(a_1 \mid s_1)\dots$$

$$J(\pi_\theta) = \mathbb{E}_{\tau\sim\rho_\theta(\tau)} \underbrace{\left[\sum_{h=0}^{\infty} \gamma^h r(s_h, a_h)\right]}_{R(\tau)} = \sum_\tau \boxed{P(\tau;\theta)} R(\tau)$$

probability of each trajectory from the distribution

Likelihood Ratio Policy Gradient

We can then rewrite our objective as

$$\arg\max_{\theta} J(\pi_{\theta}) = \arg\max_{\theta} \sum_{\tau} P(\tau;\theta)R(\tau)$$

Credits: Pieter Abbeel

Likelihood Ratio Policy Gradient

We can then rewrite our objective as

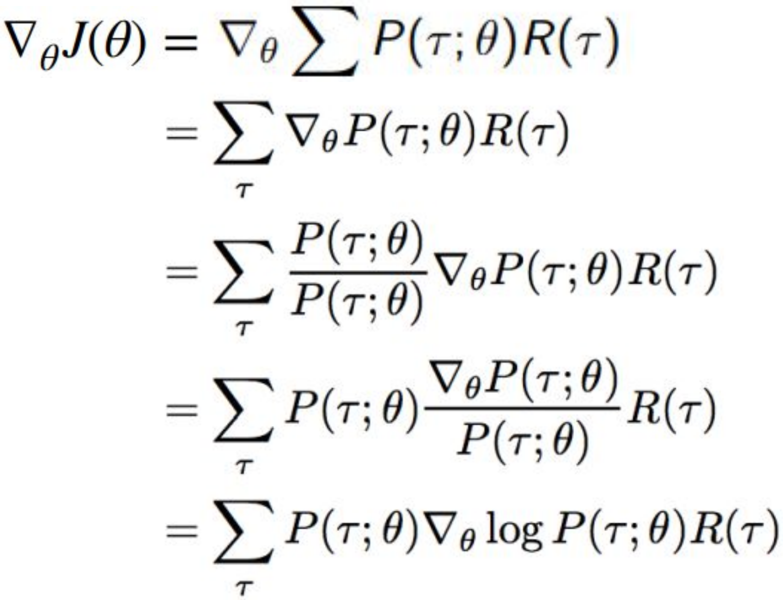
$$\arg\max_{\theta} J(\pi_{\theta}) = \arg\max_{\theta} \sum_{\tau} P(\tau;\theta)R(\tau)$$

$$\Downarrow$$

$$\nabla_{\theta} J(\theta) = \nabla_{\theta} \sum_{\tau} P(\tau;\theta)R(\tau)$$

Credits: Pieter Abbeel

Likelihood Ratio Policy Gradient



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

Credits: Pieter Abbeel

Likelihood Ratio Policy Gradient

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

Can also be derived/generalized through an importance sampling derivation

Credits: Pieter Abbeel

Likelihood Ratio Policy Gradient

$$\nabla_\theta J(\theta) = \nabla_\theta \sum_\tau P(\tau;\theta)R(\tau) \quad = \sum_\tau P(\tau;\theta)\nabla_\theta \log P(\tau;\theta)R(\tau)$$

Estimate this empirically as

$$\approx \quad \hat{g} = (1/m)\sum_{i=1}^{m} R(\tau^{(i)})\nabla_\theta \log P(\tau^{(i)};\theta)$$

Credits: Pieter Abbeel

Likelihood Ratio Policy Gradient

$$\nabla_\theta J(\theta) = \nabla_\theta \sum_\tau P(\tau;\theta)R(\tau) \quad = \sum_\tau P(\tau;\theta)\nabla_\theta \log P(\tau;\theta)R(\tau)$$

Estimate this empirically as

$$\approx \quad \hat{g} = (1/m)\sum_{i=1}^{m} R(\tau^{(i)})\nabla_\theta \log \boxed{P(\tau^{(i)};\theta)}$$

Remember now this is something like $\mu(s_0)\pi_\theta(a_0 \mid s_0)P(s_1 \mid s_0, a_0)\pi_\theta(a_1 \mid s_1)\ldots$

Likelihood Ratio Policy Gradient

$$\nabla_\theta J(\theta) = \nabla_\theta \sum_\tau P(\tau;\theta)R(\tau) = \sum_\tau P(\tau;\theta)\nabla_\theta \log P(\tau;\theta)R(\tau)$$

Estimate this empirically as

$$\approx \quad \hat{g} = (1/m)\sum_{i=1}^{m} R(\tau^{(i)})\nabla_\theta \log \boxed{P(\tau^{(i)};\theta)}$$

Do we need a transition model to compute it?

Score Function

$$
\begin{aligned}
\nabla_\theta \log P(\tau^{(i)};\theta) &= \nabla_\theta \log \left[\underbrace{\mu(s_0)}_{\text{Initial state distrib.}} \prod_{t=0}^{T-1} \underbrace{\pi_\theta(a_t|s_t)}_{\text{policy}} \underbrace{P(s_{t+1}|s_t,a_t)}_{\text{dynamics model}} \right] \\
&= \nabla_\theta \left[\log \mu(s_0) + \sum_{t=0}^{T-1} \log \pi_\theta(a_t|s_t) + \log P(s_{t+1}|s_t,a_t) \right] \\
&= \sum_{t=0}^{T-1} \underbrace{\nabla_\theta \log \pi_\theta(a_t|s_t)}_{\text{no dynamics model required!}}
\end{aligned}
$$

NO!

Credits: Pieter Abbeel

Score Function

$$
\begin{aligned}
\nabla_\theta \log P(\tau^{(i)}; \theta) &= \nabla_\theta \log \left[\underbrace{\mu(s_0)}_{\text{Initial state distrib.}} \prod_{t=0}^{T-1} \underbrace{\pi_\theta(a_t|s_t)}_{\text{policy}} \underbrace{P(s_{t+1}|s_t, a_t)}_{\text{dynamics model}} \right] \\
&= \nabla_\theta \left[\log \mu(s_0) + \sum_{t=0}^{T-1} \log \pi_\theta(a_t|s_t) + \log P(s_{t+1}|s_t, a_t) \right] \\
&= \sum_{t=0}^{T-1} \underbrace{\boxed{\nabla_\theta \log \pi_\theta(a_t|s_t)}}_{\text{no dynamics model required!}}
\end{aligned}
$$

Called **score function**

Credits: Pieter Abbeel

Likelihood Ratio Policy Gradient

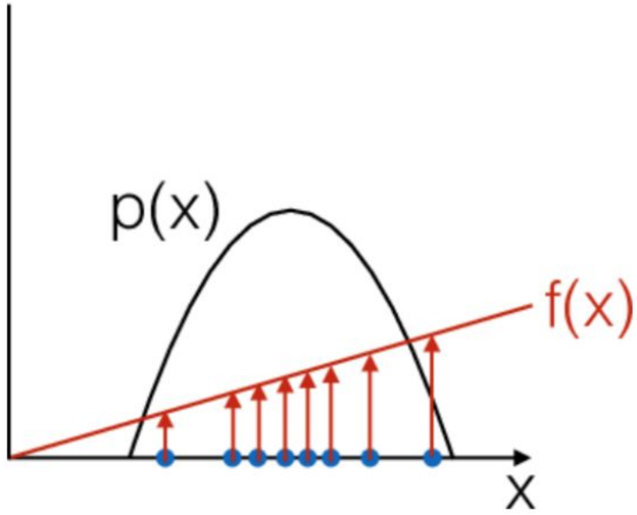
$$\nabla_\theta J(\theta) \approx \hat{g} = (1/m)\sum_{i=1}^{m} R(\tau^{(i)})\nabla_\theta \log P(\tau^{(i)};\theta)$$

$$= (1/m)\sum_{i=1}^{m} \boxed{R(\tau^{(i)})} \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t^{(i)}|s_t^{(i)})$$

Measure how good the sample trajectory is

Visualizing the PG

$$\hat{g}_i = f(x_i)\nabla_\theta \log p(x_i|\theta)$$

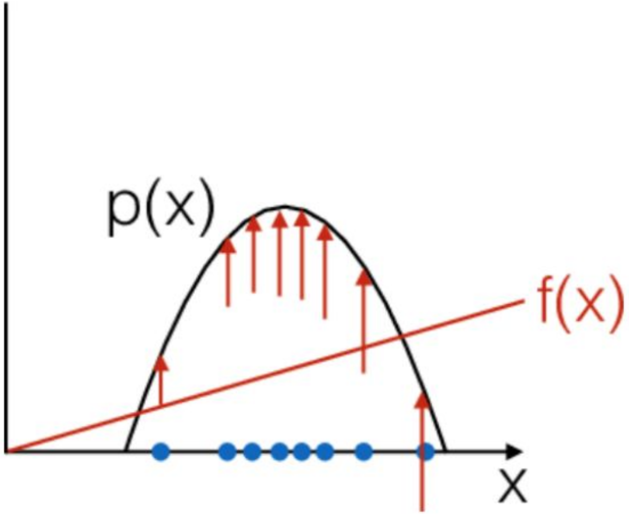


Credits: Emma Brunskill

Visualizing the PG

$$\hat{g}_i = f(x_i)\nabla_\theta \log p(x_i|\theta)$$

Moving in the direction the estimated gradient pushes up the logprob of the sample, in proportion to how good it is

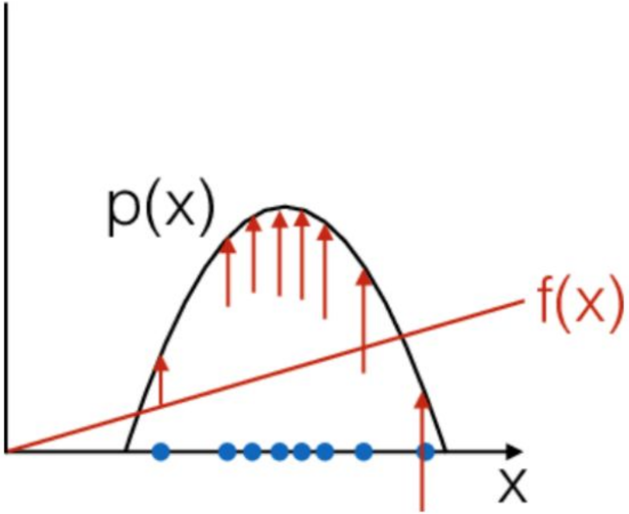


Credits: Emma Brunskill

Visualizing the PG

$$\hat{g}_i = f(x_i)\nabla_\theta \log p(x_i|\theta)$$

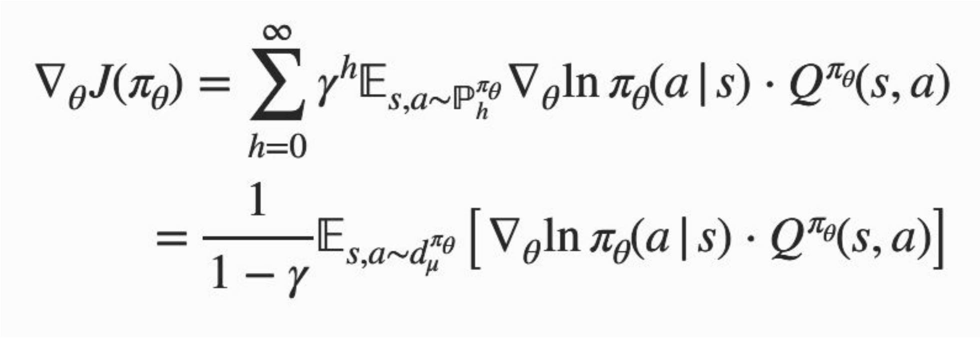
Increase the likelihood of sampling a trajectory with high total reward



Credits: Emma Brunskill

Policy Gradient Theorem (Infinite Setting)

The policy gradient theorem generalizes the likelihood ratio approach



$$\begin{aligned}\nabla_\theta J(\pi_\theta) &= \sum_{h=0}^{\infty} \gamma^h \mathbb{E}_{s,a\sim\mathbb{P}_h^{\pi_\theta}} \nabla_\theta \ln \pi_\theta(a\,|\,s) \cdot Q^{\pi_\theta}(s,a) \\ &= \frac{1}{1-\gamma} \mathbb{E}_{s,a\sim d_\mu^{\pi_\theta}} \left[\nabla_\theta \ln \pi_\theta(a\,|\,s) \cdot Q^{\pi_\theta}(s,a)\right]\end{aligned}$$

Policy Gradient Theorem (Infinite Setting)

The policy gradient theorem generalizes the likelihood ratio approach

$$
\begin{aligned}
\nabla_\theta J(\pi_\theta) &= \sum_{h=0}^{\infty} \gamma^h \mathbb{E}_{s,a\sim\mathbb{P}_h^{\pi_\theta}} \nabla_\theta \ln \pi_\theta(a\,|\,s) \cdot Q^{\pi_\theta}(s,a) \\
&= \frac{1}{1-\gamma} \mathbb{E}_{s,a\sim d_\mu^{\pi_\theta}} \left[\nabla_\theta \ln \pi_\theta(a\,|\,s) \cdot Q^{\pi_\theta}(s,a) \right]
\end{aligned}
$$

We want to slowly adjust the policy, such that probability is large at actions with large Q value

Policy Gradient Theorem (Finite Setting)

The policy gradient theorem generalizes the likelihood ratio approach

$$\nabla_\theta J(\pi_\theta) = \sum_{h=0}^{H-1} \mathbb{E}_{s_h, a_h \sim \mathbb{P}_h^{\pi_\theta}} \left[\nabla \ln \pi_\theta(a_h \mid s_h) \cdot Q_h^{\pi_\theta}(s_h, a_h) \right]$$