

Approximate Policy Iteration

Reinforcement Learning

Roberto Capobianco



SAPIENZA
UNIVERSITÀ DI ROMA

Recap

Finite-Horizon MDPs

Slightly different formulation:

$$(S, A, R, T, \mathbf{H}, \mu_0)$$

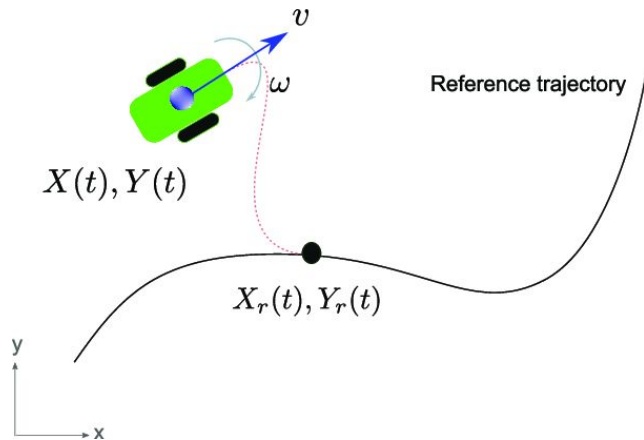
(time-horizon) $\mathbf{H} \geq 0$ and $s_0 \sim \mu_0$ (initial state distribution)

We consider time-dependent policies π

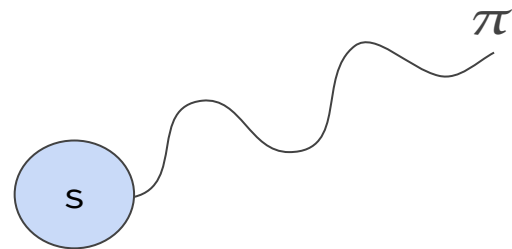
$$\pi = \{\pi_0, \pi_1, \pi_2 \dots \pi_{H-1}\}$$

Actions might be different for the same state depending on t

Very common in control!



Finite-Horizon MDP: V & Q

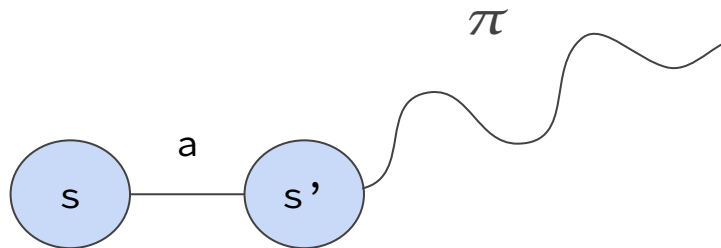


$$V_h^\pi(s) = \mathbb{E}_\pi[\sum_{\square=h}^{H-1} r(s_\square, a_\square)]$$

where $s_h=s$, $a_\square=\pi_\square(s_\square)$ and $s_{\square+1}\sim P(\cdot|s_\square, a_\square)$

$$Q_h^\pi(s, a) = \mathbb{E}_\pi[\sum_{\square=h}^{H-1} r(s_\square, a_\square)]$$

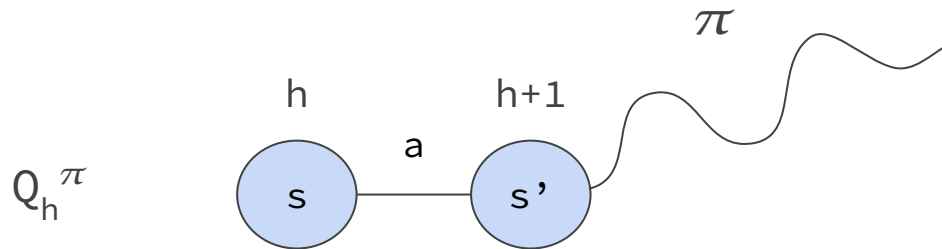
where $s_h=s$, $a_h=a$, $a_\square=\pi_\square(s_\square)$ and $s_{\square+1}\sim P(\cdot|s_\square, a_\square)$



Finite-Horizon MDP: Bellman Equation

— — —

$$Q_h^\pi(s, a) = r(s, a) + \mathbb{E}_{s' \sim p(\cdot | s, a)} [V_{h+1}^\pi(s')]$$



Finding the Optimal Policy

$$\pi^* = \{\pi_0^*, \pi_1^*, \pi_2^* \dots \pi_{H-1}^*\}$$

Let's reason backwards in time and apply dynamic programming:

$$Q_{H-1}^*(s, a) = r(s, a)$$

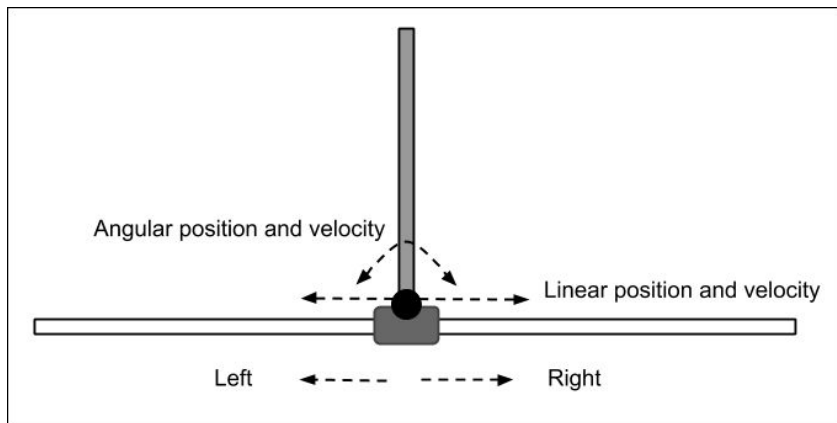
$$\pi_{H-1}^*(s) = \operatorname{argmax}_a Q_{H-1}^*(s, a)$$

$$V_{H-1}^*(s) = \max_a Q_{H-1}^*(s, a) = Q_{H-1}^*(s, \pi_{H-1}^*(s))$$



Control Problems

So far, we assumed discrete state and action spaces, but what about cartpole:



- **state:** angular pos & vel, linear pos & vel
- **action/control:** force applied on the cart
- **goal:** find the control policy which minimizes the long term cost c



Optimal Control

Given a dynamical system with a non-linear transition function f , state x in \mathbb{R}^d and control u in \mathbb{R}^k , we want to find a control policy π such that

$$\text{minimize } \mathbb{E}_{\pi}[c_H(x_H) + \sum_{h=0}^{H-1} c_h(x_h, u_h)]$$

$$\text{where } u_h = \pi(x_h) \text{ and } x_0 \sim \mu_0$$

Now this seems very familiar! Can we treat it as a
Finite-Horizon MDP



Bellman's Curse of Dimensionality

- n -dimensional (discrete) state space
- The number of states grows exponentially in n

In practice discretization is useful, but it is only computationally feasible up to 5 or 6 dimensional state spaces

Let's try to work directly in continuous space, starting from simplified problems



Linear Systems

Consider a system of this kind:

$$x_{t+1} = Ax_t + Bu_t$$

This is our
transition function!

- x_t state at time t
- u_t control (i.e., action) at time t

A in $\mathbb{R}^{d \times d}$, B in $\mathbb{R}^{d \times k}$



Quadratic Cost Function

Consider a cost function of this kind

$$c(x_t, u_t) = x_t^T Q x_t + u_t^T R u_t$$

alternative notation
 $g(x_t, u_t)$

- Q in $\mathbb{R}^{d \times d}$ and R in $\mathbb{R}^{k \times k}$ square matrices
- Q and R positive definite

As a result, there is a non-zero cost for any non-zero state with all-zero control



LQR algorithm

- Initialize P_H (at 0 or Q)
- Starting from $h = H-1$, backwards
 - Set $K_h^* = -(R+B^T P_{h+1} B)^{-1} B^T P_{h+1} A$
 - Compute $u = \pi_h^*(x_t) = K_h^* x_t$
 - Set $P_h = (Q + K_h^{*T} R K_h^* + (A + B K_h^*)^T P_{h+1} (A + B K_h^*))$
 - Set $J_h^* = x_t^T P_h x_t$

Riccati
Equation

This is the Value Iteration update done in closed form: it is always the same and solves this particular continuous-state system with a quadratic cost



LQR Extensions

Extensions to the LQR make it more generally applicable to:

- Affine systems
- Systems with stochasticity
- Regulation around non-zero fixed point for non-linear systems
- Trajectory following for non-linear systems
- ...



End Recap



SAPIENZA
UNIVERSITÀ DI ROMA

Policy Iteration

— — —

- Outputs policies at every iteration: $\{\pi_0, \pi_1, \pi_2 \dots \pi_T\}$
- Different from Value Iteration that was outputting values

Procedure:

1. Start with a random guess π_0 (can be deterministic or stochastic)
2. For $t=0, \dots, T$:

$$Q^\pi(s_t, a) = r_t + \gamma \mathbb{E}_{s' \sim p(\cdot | s, a)} [V^\pi(s')]$$

 - a. Do **policy evaluation** and compute Q^{π^t} for all s, a
 - b. Do **policy improvement** as $\pi_{t+1} = \operatorname{argmax}_a Q^{\pi^t}(s, a)$ for all s

This algorithm only makes progress, and the performance progress of the policy is monotonic



State Visitation Probability

What's the probability of visiting state s , a at time t according to π starting at s_0 ?

$$d_{s_0}^{\pi}(s, a) = (1-\gamma) \sum_{h=0}^{\infty} \gamma^h \mathbb{P}^{\pi}_h(s, a; s_0)$$

$$\mathbb{P}^{\pi}_t(s, a; s_0) = \sum_{a_0, s_1, a_1, \dots, s_{t-1}, a_{t-1}} \mathbb{P}^{\pi}(s_0, a_0, \dots, s_t=s, a_t=a)$$

$$\mathbb{P}^{\pi}(s_0, a_0, \dots, s_t, a_t) = \pi(a_0 | s_0) p(s_1 | s_0, a_0) \pi(a_1 | s_1) p(s_2 | s_1, a_1) \dots p(s_t | s_{t-1}, a_{t-1}) \pi(a_t | s_t)$$



State Visitation Probability

What's the probability of visiting state s , a at time t according to π starting at s_0 ?

$$d_{s_0}^{\pi}(s, a) = (1-\gamma) \sum_{h=0}^{\infty} \gamma^h \mathbb{P}^{\pi}_h(s, a; s_0)$$

Note that $d_{s_0}^{\pi}$ is an infinite mixture

$$\mathbb{P}_t^{\pi}(s, a; s_0) = \sum_{a_0, s_1, a_1, \dots, s_{t-1}, a_{t-1}} \mathbb{P}^{\pi}(s_0, a_0, \dots, s_t=s, a_t=a)$$

$$\mathbb{P}^{\pi}(s_0, a_0, \dots, s_t, a_t) = \pi(a_0 | s_0) p(s_1 | s_0, a_0) \pi(a_1 | s_1) p(s_2 | s_1, a_1) \dots p(s_t | s_{t-1}, a_{t-1}) \pi(a_t | s_t)$$

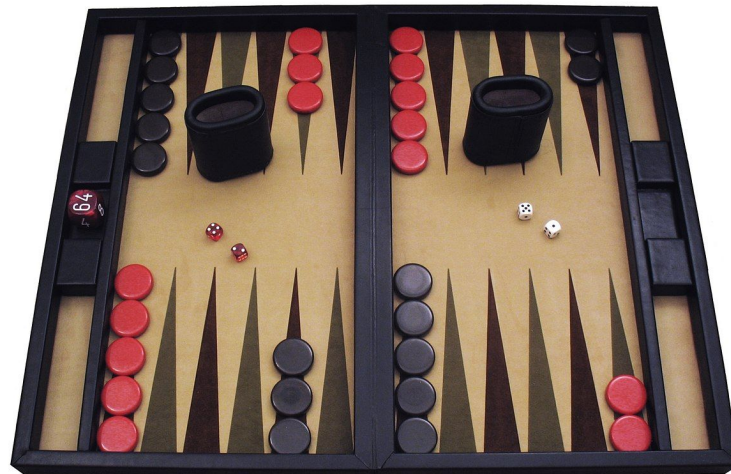
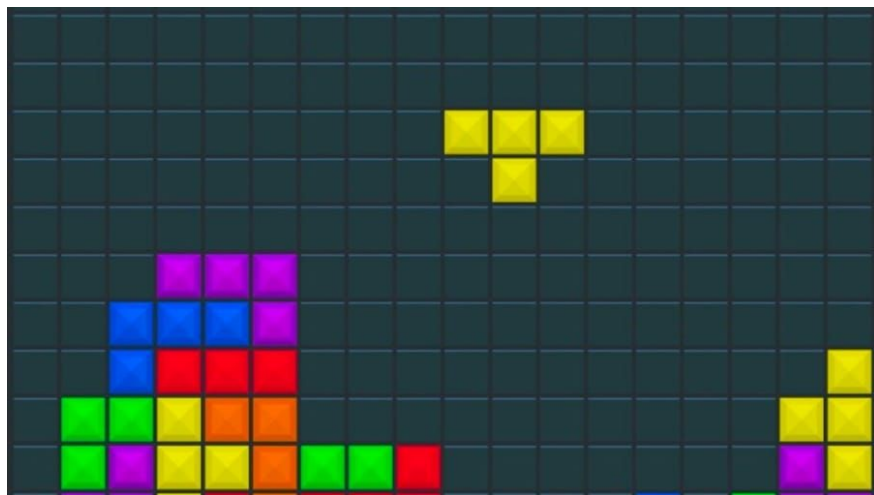


Approximate Policy Iteration

What if the state-space is large and we cannot do exact or iterative policy evaluation for all states?

Approximate Policy Iteration

What if the state-space is large and we cannot do exact or iterative policy evaluation for all states?



Approximate Policy Iteration

What if the state-space is large **or continuous** and we cannot do exact or iterative policy evaluation for all states?



Approximate Policy Iteration

What if the state-space is large and we cannot do exact or iterative policy evaluation for all states?

Assumptions: the (infinite-horizon) MDP is still known, but the state-space is too large to just enumerate all states and compute $V^\pi(s)$



Approximate Policy Iteration

What if the state-space is large and we cannot do exact or iterative policy evaluation for all states?

Assumptions:

$(S, A, R, T, \gamma, \mu_\theta)$ is given

Q is in $[0, 1/(1-\gamma)]$



Approximate Policy Iteration

— — —

- Outputs policies at every iteration: $\{\pi_0, \pi_1, \pi_2 \dots \pi_T\}$

Procedure:

1. Start with a random guess π_0
2. For $t=0, \dots, T$:
 - a. Do **policy evaluation** and compute Q^{π_t} ~~for all s, a~~

$$Q^{\pi}(s_t, a) = r_t + \gamma \mathbb{E}_{s' \sim p(\cdot | s, a)} [V^{\pi}(s')]$$



Approximate Policy Iteration

- Outputs policies at every iteration: $\{\pi_0, \pi_1, \pi_2 \dots \pi_T\}$

Procedure:

1. Start with a random guess π_0
2. For $t=0, \dots, T$:
 - a. Do **policy evaluation** and compute Q^{π^t} ~~for all s, a~~
$$Q^{\pi}(s_t, a) = r_t + \gamma \mathbb{E}_{s' \sim p(\cdot | s, a)} [V^{\pi}(s')]$$
 - b. Do **policy improvement** as $\pi_{t+1} = \operatorname{argmax}_a Q^{\pi^t}(s, a)$ for all s
~~argmax is still doable, we can still enumerate actions or discretize them~~



Approximate Policy Evaluation

We build an **approximation** V^{π} of the true value function V^{π}

If the approximation is close to the true value, then the optimal policy will be close-to-optimal



Approximate Policy Evaluation

We build an **approximation** V^π of the true value function V^π

If the approximation is close to the true value, then the optimal policy will be close-to-optimal

Error bounds exist, but we will skip them for your happiness :)



Approximate Policy Evaluation

We build an **approximation** V^π of the true value function V^π

If the approximation is close to the true value, then the optimal policy will be close-to-optimal

Approximation for large state-spaces is needed to generalize among states and avoid looking at the whole S



Approximate Policy Evaluation

We build an **approximation** V^{π} of the true value function V^{π}

If the approximation is close to the true value, then the optimal policy will be close-to-optimal

Approximation for large state-spaces is needed to generalize among states and avoid looking at the whole S

We use a function approximator



Approximate Policy Evaluation

We build an **approximation** V^π of the true value function V^π

If the approximation is close to the true value, then the optimal policy will be close-to-optimal

Approximation for large state-spaces is needed to generalize among states and avoid looking at the whole S

We use a function approximator

e.g., linear approximators, neural nets, non-parametric, etc.



Approximate Policy Evaluation

— — —

To be fair, we can directly approximate Q , so let's do that



Approximate Policy Evaluation

— — —

To be fair, we can directly approximate Q , so let's do that

Note that this also means that we can also get rid of the assumption of knowing the MDP



Approximate Policy Evaluation

— — —

To be fair, we can directly approximate Q , so let's do that

What do we need?



Data and Least Square Regression

To be fair, we can directly approximate Q , so let's do that

What do we need?

DATA $D = \{s_i, a_i, y_i\}_{i=1}^N$ with y being our label!

with those we can then use least-square regression to extract a function Q in the family of functions

$$S \times A \rightarrow [0, 1/(1-\gamma)]$$



Data and Least Square Regression

To be fair, we can directly approximate Q , so let's do that

What do we need?

DATA $D = \{s_i, a_i, y_i\}_{i=1}^N$ with y being our label!

with those we can then use least-square regression to extract a function Q in the family of functions

$$Q: S \times A \rightarrow [0, 1/(1-\gamma)]$$

$$\operatorname{argmin}_{Q \in \mathcal{Q}} \sum_{i=1}^N (Q(s_i, a_i) - y_i)^2$$



Data and Least Square Regression

$$\operatorname{argmin}_{\mathbf{Q}} \sum_{i=1}^N (\mathbf{Q}(\mathbf{s}_i, \mathbf{a}_i) - y_i)^2$$

This is just a regression problem, which

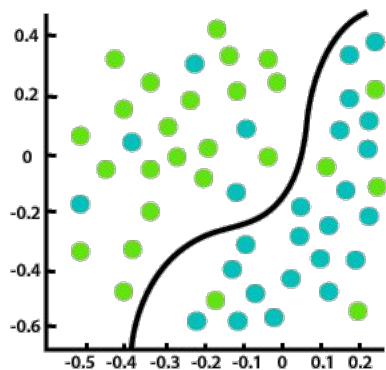
- is numerically tractable
- has generalization bounds



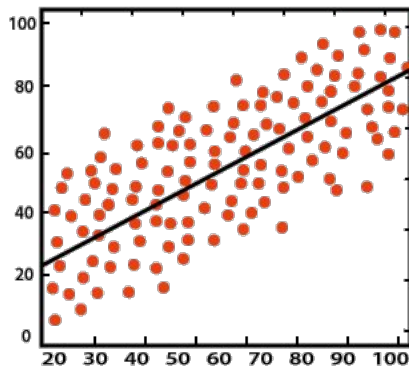
Supervised Learning Digression



Supervised Learning



Classification

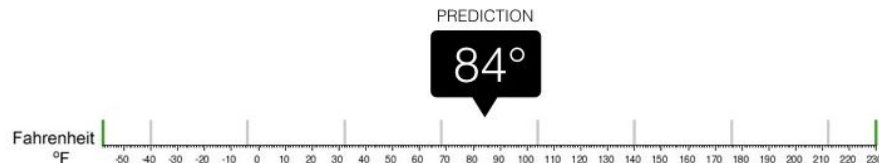


Regression



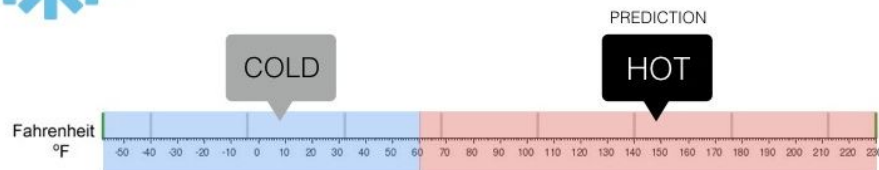
Regression

What is the temperature going to be tomorrow?



Classification

Will it be Cold or Hot tomorrow?



Supervised Learning: Regression

Given a **data distribution** from which we sample points x_i and labels $y_i = f(x_i) + \epsilon_i$, with $\mathbb{E}[\epsilon_i] = 0$ and $|\epsilon_i| \leq c$, we want to approximate f using a finite set of data (dataset):

$$f^\wedge = \operatorname{argmin}_{f^\wedge \in F} \sum_{i=1}^N (f^\wedge(x_i) - y_i)^2$$

$$\text{with } F = \{f^\wedge: X \rightarrow \mathbb{R}\}$$



Supervised Learning: Regression

Given a **data distribution** D from which we sample points x_i and labels $y_i = f(x_i) + \epsilon_i$, with $\mathbb{E}[\epsilon_i] = 0$ and $|\epsilon_i| \leq c$, we want to approximate f using a finite set of data (dataset):

Empirical
Risk
Minimizer

$$f^\wedge = \operatorname{argmin}_{f^\wedge \in F} \sum_{i=1}^N (f^\wedge(x_i) - y_i)^2$$

with $F = \{f^\wedge: X \rightarrow \mathbb{R}\}$

We can generalize under the same data distribution

$$\mathbb{E}_{x \sim D} (f^\wedge(x) - f(x))^2 \leq \delta \text{ with } \delta \text{ small}$$



Supervised Learning: Distribution Mismatch

— — —

$\mathbb{E}_{x \sim D'} (f^{\wedge}(x) - f(x))^2$ can be huge!

If $D' \neq D$

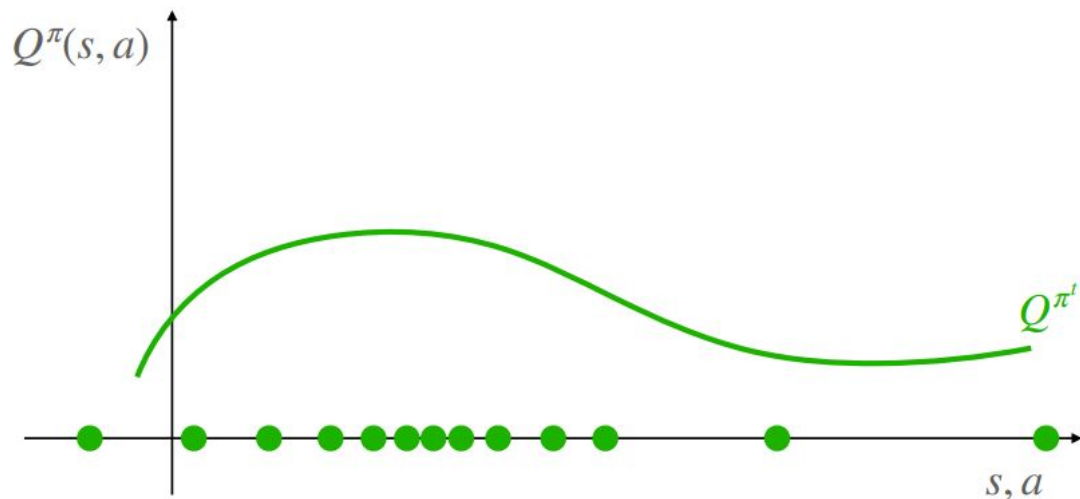


End

Supervised Learning Digression



Oscillation from Distribution Change

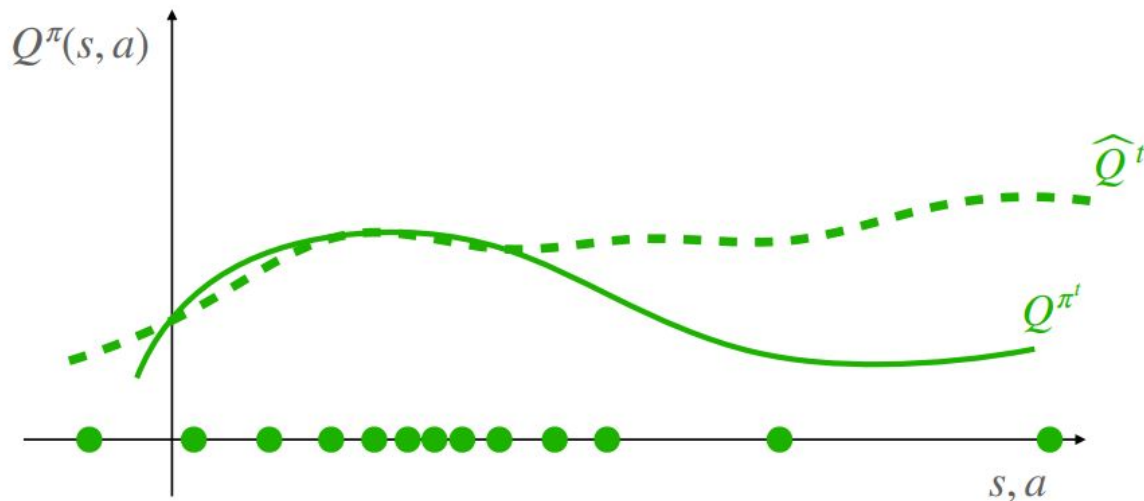


Credits: Wen Sun

Green dots: (s, a) from π^t
Red dots: (s, a) from π^{t+1}



Oscillation from Distribution Change

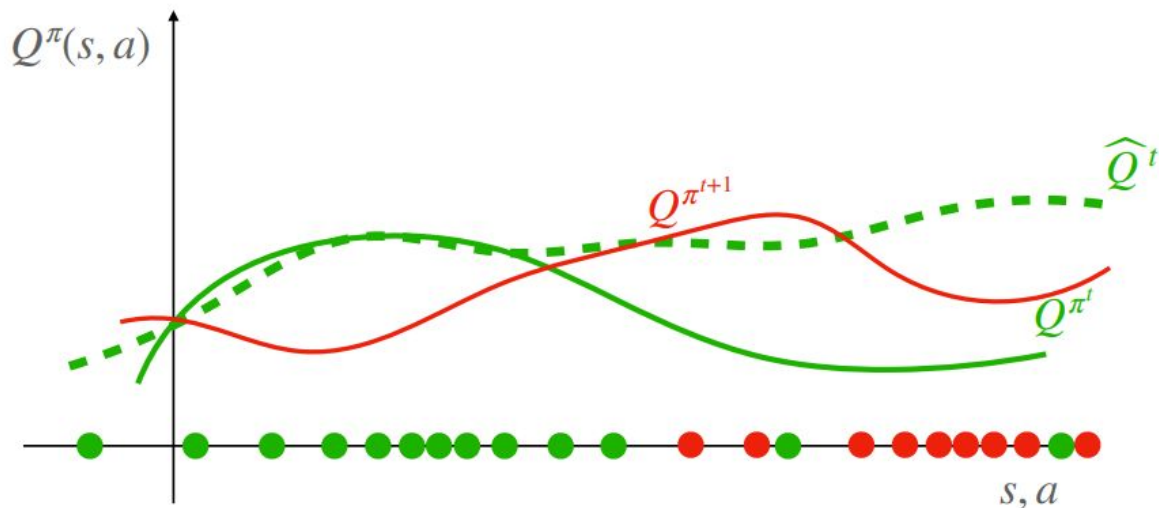


Credits: Wen Sun

Green dots: (s, a) from π^t
Red dots: (s, a) from π^{t+1}



Oscillation from Distribution Change



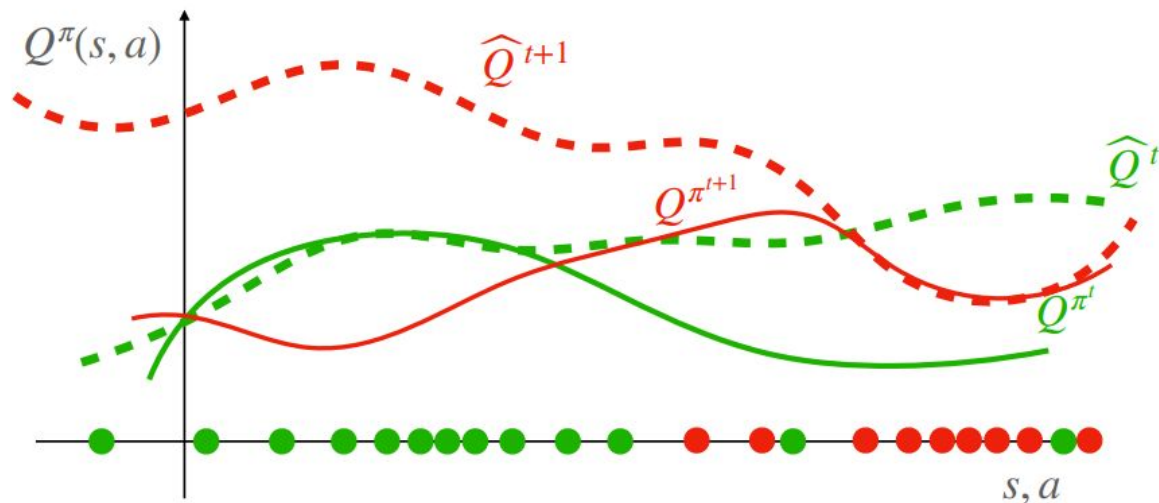
Credits: Wen Sun

Green dots: (s, a) from π^t

Red dots: (s, a) from π^{t+1}



Oscillation from Distribution Change

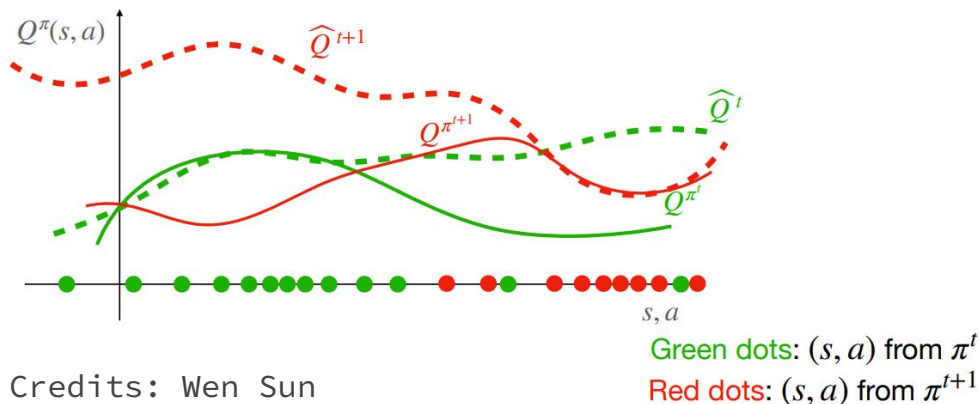


Credits: Wen Sun

Green dots: (s, a) from π^t
Red dots: (s, a) from π^{t+1}



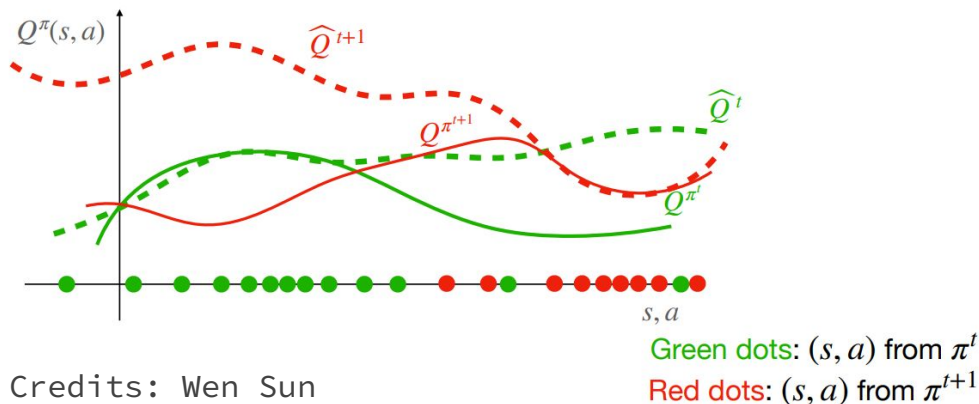
Oscillation from Distribution Change



We cannot guarantee anymore monotonic improvement!



Oscillation from Distribution Change



Our estimation is only good under $d_{\mu^0}^{\pi}$ and to make sure we have monotonic improvement we need a strong coverage assumption

Data Generation

— — —

2 steps:

1. Roll-in
2. Roll-out & compute supervision targets



Data Generation

2 steps:

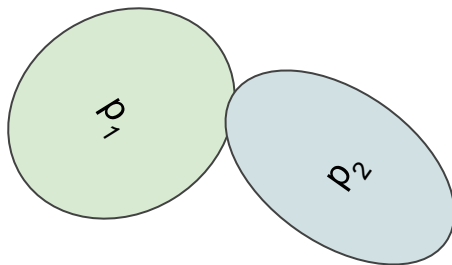
1. Roll-in

2. Roll-out & compute supervision targets

We want to sample our $(s,a) \sim d^{\pi}_{s_0}(s,a) = (1-\gamma) \sum_{h=0}^{\infty} \gamma^h \mathbb{P}^{\pi}_h(s,a;s_0)$



Sampling From Mixtures



$$p = (1-\alpha)p_1 + \alpha p_2$$

- Flip a coin with probability $[\alpha, 1-\alpha]$
- Commit to a specific p_i based on that and sample from p_i



Data Generation

2 steps:

1. **Roll-in**
2. Roll-out & compute supervision targets

We want to sample our $(s,a) \sim d^{\pi}_{s_0}(s,a) = (1-\gamma) \sum_{h=0}^{\infty} \gamma^h \mathbb{P}^{\pi_h}(s,a;s_0)$

- Sample h from $\gamma^h/(1-\gamma)$, thus committing to a specific $\mathbb{P}^{\pi_h}(s,a;s_0)$



Data Generation

2 steps:

1. Roll-in

2. Roll-out & compute supervision targets

We want to sample our $(s,a) \sim d^{\pi}_{s_0}(s,a) = (1-\gamma) \sum_{h=0}^{\infty} \gamma^h \mathbb{P}^{\pi}_h(s,a;s_0)$

- Sample h from $\gamma^h/(1-\gamma)$, thus committing to a specific $\mathbb{P}^{\pi}_h(s,a;s_0)$
- Follow π for h timesteps starting from $s_0 \sim \mu_0$ and get s_h, a_h



Data Generation

2 steps:

1. Roll-in

2. Roll-out & compute supervision targets

Given s , a , how do we estimate $Q^\pi(s,a)$?



Data Generation

2 steps:

1. Roll-in

2. Roll-out & compute supervision targets

Given s , a , how do we estimate $Q^\pi(s,a)$?

$$Q^\pi(s_t, a_t) = \mathbb{E}[\sum_{h=0}^{\infty} \gamma^h r_h \mid (s_0, a_0) = (s_t, a_t), a_{h+1} = \pi(s_h), s_{h+1} \sim p(\cdot \mid s_h, a_h)]$$

$$Q^\pi(s_t, a) = r_t + \gamma \mathbb{E}_{s' \sim p(\cdot \mid s, a)} [V^\pi(s')]$$



Data Generation

2 steps:

1. Roll-in
- 2. Roll-out & compute supervision targets**

Given s , a , how do we estimate $Q^\pi(s,a)$?

$$Q^\pi(s_t, a_t) = \mathbb{E}[\sum_{h=0}^{\infty} \gamma^h r_h \mid (s_0, a_0) = (s_t, a_t), a_{h+1} = \pi(s_h), s_{h+1} \sim p(\cdot \mid s_h, a_h)]$$

How do we get an unbiased
estimate of this?



Data Generation

2 steps:

1. Roll-in
- 2. Roll-out & compute supervision targets**

Given s , a , how do we estimate $Q^\pi(s,a)$?

$$Q^\pi(s_t, a_t) = \mathbb{E}[\sum_{h=0}^{\infty} \gamma^h r_h | (s_0, a_0) = (s_t, a_t), a_{h+1} = \pi(s_h), s_{h+1} \sim p(\cdot | s_h, a_h)]$$

Sample many times and average!



Data Generation

2 steps:

1. Roll-in

2. Roll-out & compute supervision targets

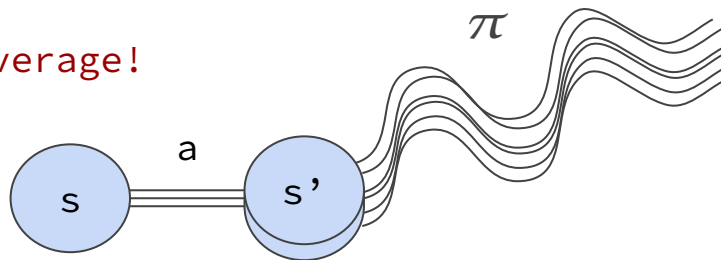
Given s , a , how do we estimate $Q^\pi(s,a)$?

$$Q^\pi(s_t, a_t) = \mathbb{E}[\sum_{h=0}^{\infty} \gamma^h r_h | (s_0, a_0) = (s_t, a_t), a_{h+1} = \pi(s_h), s_{h+1} \sim p(\cdot | s_h, a_h)]$$

Sample many times and average!



SAPIENZA
UNIVERSITÀ DI ROMA



Data Generation

2 steps:

1. Roll-in
2. **Roll-out & compute supervision targets**

Given s , a , how do we estimate $Q^\pi(s,a)$?

$$Q^\pi(s_t, a_t) = \mathbb{E}[\sum_{h=0}^{\infty} \gamma^h r_h | (s_0, a_0) = (s_t, a_t), a_{h+1} = \pi(s_h), s_{h+1} \sim p(\cdot | s_h, a_h)]$$

Easier said than done :(

Infinite horizon!



SAPIENZA
UNIVERSITÀ DI ROMA

Data Generation

2 steps:

1. Roll-in
- 2. Roll-out & compute supervision targets**

Given s , a , how do we estimate $Q^\pi(s,a)$?

$$Q^\pi(s_t, a_t) = \mathbb{E}[\sum_{h=0}^{\infty} \gamma^h r_h | (s_0, a_0) = (s_t, a_t), a_{h+1} = \pi(s_h), s_{h+1} \sim p(\cdot | s_h, a_h)]$$

Use γ as a sampling factor
again to choose an horizon



Data Generation

2 steps:

1. Roll-in

2. Roll-out & compute supervision targets

Given s , a , how do we estimate $Q^\pi(s,a)$?

- Start at s,a
- Repeat:
 - Get $r(s,a)$
 - With probability $1-\gamma$ terminate and return $y=\sum \gamma^h r_h$
 - Execute action and get in s'



Data Generation

2 steps:

1. Roll-in
- 2. Roll-out & compute supervision targets**

Given s , a , how do we estimate $Q^\pi(s,a)$?

- Start at s,a
- Repeat:
 - Get $r(s,a)$
 - With probability $1-\gamma$ terminate and return $y=\sum \gamma^h r_h$
 - Execute action and get in s'

$$D = \{s_i, a_i, y_i\}_{i=1}^N$$

