Approximate Dynamic Programming meets Statistical Learning Theory

A. Lazaric (*INRIA Lille – Team SequeL*)

Journées "Méthodologies pour le contrôle de systèmes complexes"

INRIA Lille – Team SequeL

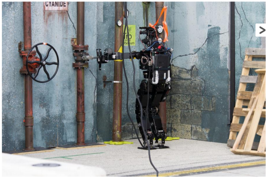
November 17, 2014

Summary of the Talk

Approximate dynamic programming can/could ***successfully*** solve a wide range of problems

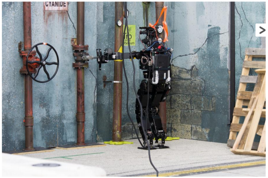
Summary of the Talk

Approximate dynamic programming can/could ***successfully*** solve a wide range of problems



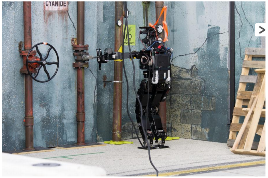
Summary of the Talk

Approximate dynamic programming can/could ***successfully*** solve a wide range of problems

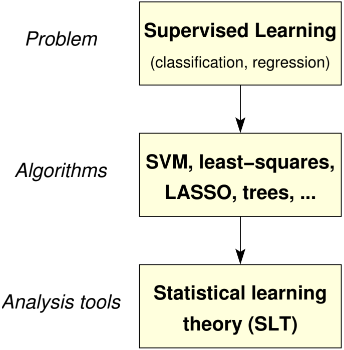


Summary of the Talk

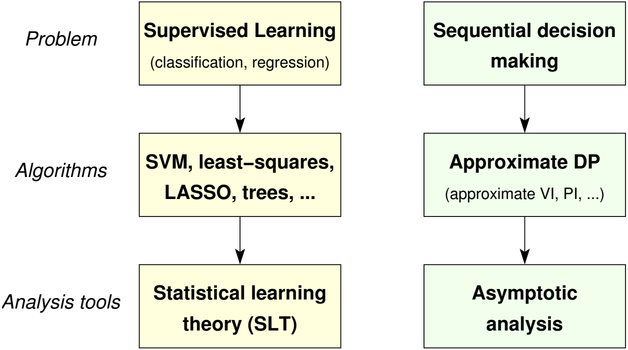
Approximate dynamic programming can/could ***successfully*** solve a wide range of problems



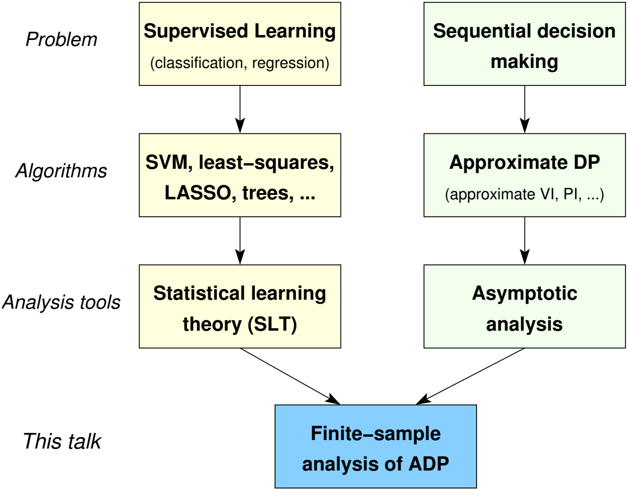

Summary of the Talk



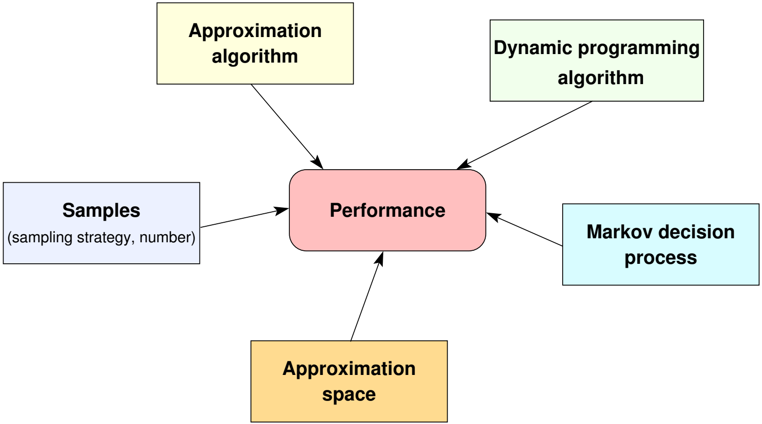
Summary of the Talk



Summary of the Talk



Summary of the Talk



Outline

Outline

Preliminaries

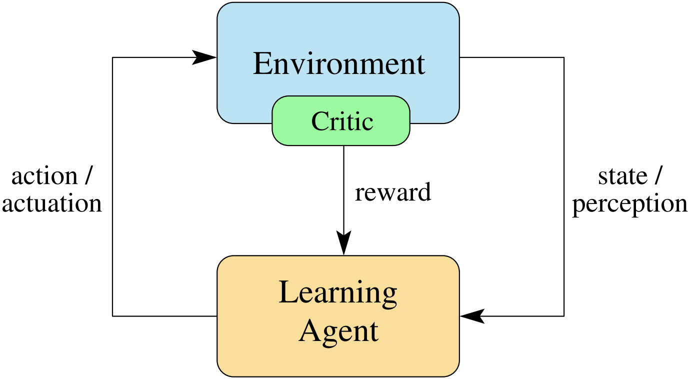
Approximate Dynamic Programming

Linear Fitted Q-Iteration

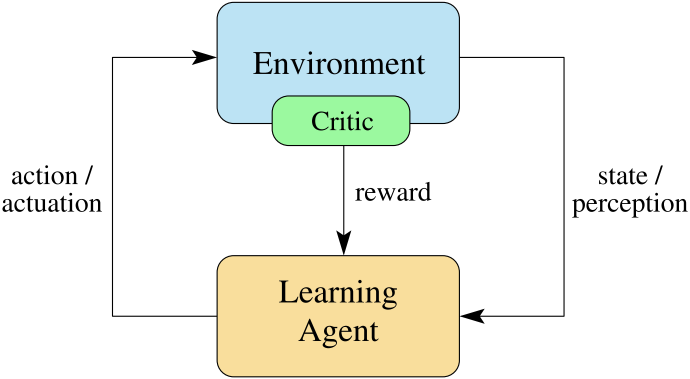
Least-Squares Policy Iteration (LSPI)

Discussion

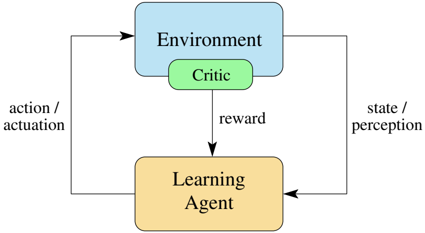
Markov Decision Process



The Reinforcement Learning Model



The Reinforcement Learning Model (Glossary)



- *Environment* = system to control
- *Agent* = controller
- *State* = fully observable state
- *Action* = control
- *Reward* = cost

Markov Decision Process

Definition (Markov decision process)

A ***discounted Markov decision process*** *is a tuple* $M = (X, A, p, r, \gamma)$*:*

Markov Decision Process

Definition (Markov decision process)

A ***discounted Markov decision process*** *is a tuple* $M = (X, A, p, r, \gamma)$*:*

- X *is the* *state* *space,*

Markov Decision Process

Definition (Markov decision process)

A ***discounted Markov decision process*** *is a tuple* $M = (X, A, p, r, \gamma)$*:*

- X *is the state space,*
- A *is the action space,*

Markov Decision Process

Definition (Markov decision process)

*A **discounted Markov decision process** is a tuple $M = (X, A, p, r, \gamma)$:*

- *X is the state space,*
- *A is the action space,*
- *$p(y|x, a)$ is the (stationary and Markov) transition probability*

$$p(y|x, a) = \mathbb{P}(x_{t+1} = y | x_t = x, a_t = a),$$

Markov Decision Process

Definition (Markov decision process)

A ***discounted Markov decision process*** *is a tuple* $M = (X, A, p, r, \gamma)$*:*

- X *is the state space,*
- A *is the action space,*
- $p(y|x, a)$ *is the (stationary and Markov) transition probability*

$$p(y|x, a) = \mathbb{P}(x_{t+1} = y | x_t = x, a_t = a),$$

- $r(x, a, y)$ *is the reward of transition* (x, a, y).

Markov Decision Process

Definition (Markov decision process)

*A **discounted Markov decision process** is a tuple $M = (X, A, p, r, \gamma)$:*

- X *is the state space,*
- A *is the action space,*
- $p(y|x,a)$ *is the (stationary and Markov) transition probability*

$$p(y|x,a) = \mathbb{P}(x_{t+1} = y | x_t = x, a_t = a),$$

- $r(x,a,y)$ *is the reward of transition* (x,a,y)*.*
- $\gamma \in (0,1)$ *is the discount factor*

Markov Decision Process

Definition (Markov decision process)

*A **discounted Markov decision process** is a tuple $M = (X, A, p, r, \gamma)$:*

- X *is the state space,*
- A *is the action space,*
- $p(y|x,a)$ *is the (stationary and Markov) transition probability*

$$p(y|x,a) = \mathbb{P}(x_{t+1} = y | x_t = x, a_t = a),$$

- $r(x,a,y)$ *is the reward of transition* (x,a,y).
- $\gamma \in (0,1)$ *is the discount factor*

Definition (Policy)

*A **policy** is a (stationary and deterministic) mapping*

$$\pi : X \to A$$

Infinite Time Horizon with Discount

Definition (Value functions)

For any policy π, the (action-) state value function $V^\pi : X \mapsto \mathbb{R}$ ($Q^\pi : X \times A \mapsto \mathbb{R}$) is

$$V^\pi(x) = \mathbb{E}\Big[\sum_{t=0}^{\infty} \gamma^t r(x_t, \pi(x_t)) \,|\, x_0 = x; \pi\Big]$$

$$Q^\pi(x, a) = \mathbb{E}\Big[\sum_{t=0}^{\infty} \gamma^t r(x_t, a_t) | x_0 = x, a_0 = a, a_t = \pi(x_t), \forall t \geq 1\Big]$$

Infinite Time Horizon with Discount

Definition (Value functions)

For any policy π, the (action-) state value function $V^\pi : X \mapsto \mathbb{R}$ ($Q^\pi : X \times A \mapsto \mathbb{R}$) is

$$V^\pi(x) = \mathbb{E}\Big[\sum_{t=0}^{\infty} \gamma^t r(x_t, \pi(x_t)) \,|\, x_0 = x; \pi\Big]$$

$$Q^\pi(x,a) = \mathbb{E}\Big[\sum_{t=0}^{\infty} \gamma^t r(x_t, a_t) | x_0 = x, a_0 = a, a_t = \pi(x_t), \forall t \geq 1\Big]$$

Definition (Optimal policy and optimal value function)

The solution to an MDP is an optimal policy π^* satisfying

$$\pi^* \in \arg\max_{\pi \in \Pi} V^\pi$$

and its value function is the optimal value function $V^* = V^{\pi^*}$.

The Bellman Operators

Notation. w.l.o.g. a discrete state space $|X| = N$ and $V^\pi \in \mathbb{R}^N$.

Definition

For any $W \in \mathbb{R}^N$, the Bellman operator $\mathcal{T}^\pi : \mathbb{R}^N \to \mathbb{R}^N$ is

$$\mathcal{T}^\pi W(x) = r(x, \pi(x)) + \gamma \sum_y p(y|x, \pi(x)) W(y),$$

The Bellman Operators

Notation. w.l.o.g. a discrete state space $|X| = N$ and $V^\pi \in \mathbb{R}^N$.

Definition

For any $W \in \mathbb{R}^N$, the Bellman operator $\mathcal{T}^\pi : \mathbb{R}^N \to \mathbb{R}^N$ is

$$\mathcal{T}^\pi W(x) = r(x, \pi(x)) + \gamma \sum_y p(y|x, \pi(x)) W(y),$$

and the optimal Bellman operator is

$$\mathcal{T} W(x) = \max_{a \in A} \Big[r(x, a) + \gamma \sum_y p(y|x, a) W(y) \Big].$$

The Bellman Operators

Notation. w.l.o.g. a discrete state space $|X| = N$ and $V^\pi \in \mathbb{R}^N$.

Definition

For any $W \in \mathbb{R}^N$, the Bellman operator $\mathcal{T}^\pi : \mathbb{R}^N \to \mathbb{R}^N$ is

$$\mathcal{T}^\pi W(x) = r(x, \pi(x)) + \gamma \sum_y p(y|x, \pi(x)) W(y),$$

and the optimal Bellman operator is

$$\mathcal{T} W(x) = \max_{a \in A} \Big[r(x, a) + \gamma \sum_y p(y|x, a) W(y) \Big].$$

With abuse of notation $\mathcal{T}^\pi$ and $\mathcal{T}$ will be used for Q-functions as well.

The Bellman Operators

The Bellman operators are *γ-Contraction in L_∞-norm*: for any $W_1, W_2 \in \mathbb{R}^N$

$$\begin{aligned} \|\mathcal{T}^\pi W_1 - \mathcal{T}^\pi W_2\|_\infty &\leq \gamma \|W_1 - W_2\|_\infty, \\ \|\mathcal{T} W_1 - \mathcal{T} W_2\|_\infty &\leq \gamma \|W_1 - W_2\|_\infty. \end{aligned}$$

Thus

V^π is the *unique fixed point* of $\mathcal{T}^\pi$,

V^* is the *unique fixed point* of $\mathcal{T}$.

Outline

Exact Value Iteration

1. Let Q_0 be any Q-function

Exact Value Iteration

1. Let Q_0 be any Q-function
2. At each iteration $k = 1, 2, \ldots, K$

Exact Value Iteration

1. Let Q_0 be any Q-function
2. At each iteration $k = 1, 2, \ldots, K$
   - Compute $Q_{k+1} = \mathcal{T} Q_k$

Exact Value Iteration

1. Let Q_0 be any Q-function
2. At each iteration $k = 1, 2, \ldots, K$
    - Compute $Q_{k+1} = \mathcal{T} Q_k$
3. Return the greedy policy

$$\pi_K(x) \in \arg\max_{a \in A} Q_K(x, a)$$

Exact Value Iteration

1. Let Q_0 be any Q-function
2. At each iteration $k = 1, 2, \dots, K$
   - Compute $Q_{k+1} = \mathcal{T} Q_k$
3. Return the greedy policy

$$\pi_K(x) \in \arg\max_{a \in A} Q_K(x, a)$$

From the *contraction* property of $\mathcal{T}$

$$\begin{aligned} ||Q_{k+1} - Q^*||_\infty = ||\mathcal{T} Q_k - \mathcal{T} Q^*||_\infty &\leq \gamma ||Q_k - Q^*||_\infty \\ &\leq \gamma^{k+1} ||Q_0 - Q^*||_\infty \rightarrow 0 \end{aligned}$$

Exact Policy Iteration

1. Let π_0 be *any* stationary policy

Exact Policy Iteration

1. Let π_0 be *any* stationary policy
2. At each iteration $k = 1, 2, \ldots, K$

Exact Policy Iteration

1. Let π_0 be *any* stationary policy
2. At each iteration $k = 1, 2, \ldots, K$
   - *Policy evaluation*: given π_k, compute Q^{π_k}.

Exact Policy Iteration

1. Let π_0 be *any* stationary policy
2. At each iteration $k = 1, 2, \ldots, K$
   - *Policy evaluation*: given π_k, compute Q^{π_k}.
   - *Policy improvement*: compute the *greedy* policy

$$\pi_{k+1}(x) \in \arg\max_{a \in A} Q^{\pi_k}(x, a).$$

Exact Policy Iteration

1. Let π_0 be *any* stationary policy
2. At each iteration $k = 1, 2, \dots, K$
   - *Policy evaluation*: given π_k, compute Q^{π_k}.
   - *Policy improvement*: compute the *greedy* policy

$$\pi_{k+1}(x) \in \arg\max_{a \in A} Q^{\pi_k}(x, a).$$

3. Return the last policy π_K

Exact Policy Iteration

1. Let π_0 be *any* stationary policy
2. At each iteration $k = 1, 2, \ldots, K$
   - *Policy evaluation*: given π_k, compute Q^{π_k}.
   - *Policy improvement*: compute the *greedy* policy

$$\pi_{k+1}(x) \in \arg\max_{a \in A} Q^{\pi_k}(x, a).$$

3. Return the last policy π_K

From the Bellman operators

$$V^{\pi_{k+1}} \geq V^{\pi_k}$$

Limitation of Exact Dynamic Programming

Dynamic programming algorithms require

- ***Explicit*** definition of transition probabilities $p(\cdot|x, a)$ and reward function $r(x, a)$,
- ***Exact*** representation of action-value functions Q in $X \times A$.

Limitation of Exact Dynamic Programming

Dynamic programming algorithms require

- ***Explicit*** definition of transition probabilities $p(\cdot|x,a)$ and reward function $r(x,a)$,
- ***Exact*** representation of action-value functions Q in $X \times A$.

Approximate DP relaxes these requirements by using

- ***Samples*** $\{x_i, a_i, x'_i, r_i\}_i$ obtained from a generative model (e.g., a simulator) of the MDP,
- An ***approximation space*** $\mathcal{F} = \{f : X \times A \to \mathbb{R}\}$ to approximate Q-functions.

Approximate Value Iteration

Input: *samples* $\mathcal{D} = \{x_i, a_i, x'_i, r_i\}_{i=1}^n$, *approximation space* $\mathcal{F}$

1. Let Q_0 be any Q-function
2. At each iteration $k = 1, 2, \ldots, K$
   - Compute $\widehat{Q}_{k+1} \approx \mathcal{T} Q_k$ (using $\mathcal{D}$ and $\mathcal{F}$)
3. Return the greedy policy

$$\pi_K(x) \in \arg\max_{a \in A} \widehat{Q}_K(x, a)$$

Approximate Value Iteration

Input: *samples* $\mathcal{D} = \{x_i, a_i, x'_i, r_i\}_{i=1}^n$, *approximation space* $\mathcal{F}$

1. Let Q_0 be any Q-function
2. At each iteration $k = 1, 2, \ldots, K$
   - Compute $\widehat{Q}_{k+1} \approx \mathcal{T} Q_k$ (using $\mathcal{D}$ and $\mathcal{F}$)
3. Return the greedy policy

$$\pi_K(x) \in \arg\max_{a \in A} \widehat{Q}_K(x, a)$$

Problem: $||Q^* - \widehat{Q}_{k+1}|| \overset{?}{\leq} \gamma ||Q^* - Q_k||$

Approximate Policy Iteration

Input: *samples* $\mathcal{D} = \{x_i, a_i, x'_i, r_i\}_{i=1}^n$, *approximation space* $\mathcal{F}$

1. Let π_0 be *any* stationary policy
2. At each iteration $k = 1, 2, \ldots, K$
   - *Policy evaluation* given π_k, compute $\widehat{Q}^{\pi_k} \approx Q^{\pi_k}$ (using $\mathcal{D}$ and $\mathcal{F}$).
   - *Policy improvement*: compute the *greedy* policy

$$\pi_{k+1}(x) \in \arg\max_{a \in A} \widehat{Q}^{\pi_k}(x, a).$$

3. Return the last policy π_K

Approximate Policy Iteration

Input: *samples* $\mathcal{D} = \{x_i, a_i, x'_i, r_i\}_{i=1}^n$, *approximation space* $\mathcal{F}$

1. Let π_0 be *any* stationary policy
2. At each iteration $k = 1, 2, \ldots, K$
   - *Policy evaluation* given π_k, compute $\widehat{Q}^{\pi_k} \approx Q^{\pi_k}$ (using $\mathcal{D}$ and $\mathcal{F}$).
   - *Policy improvement*: compute the *greedy* policy

$$\pi_{k+1}(x) \in \arg\max_{a \in A} \widehat{Q}^{\pi_k}(x, a).$$

3. Return the last policy π_K

Problem: $V^{\pi_k} \overset{?}{\geq} V^{\pi_{k-1}}$

Performance Bounds

Question: what is the performance of the policy π_K returned by an ADP algorithm?

Performance Bounds

Question: what is the performance of the policy π_K returned by an ADP algorithm?

$$||V^* - V^{\pi_K}||_{p,\mu}$$

Performance Bounds

Question: what is the performance of the policy π_K returned by an ADP algorithm?

$$||V^* - V^{\pi_K}||_{p,\mu} \leq \text{bound}$$

Performance Bounds

Question: what is the performance of the policy π_K returned by an ADP algorithm?

$$||V^* - V^{\pi_K}||_{p,\mu} \leq \text{bound}(\text{MDP}, \text{alg}, \mathcal{D}, \mathcal{F})$$

Performance Bounds

Question: what is the performance of the policy π_K returned by an ADP algorithm?

$$||V^* - V^{\pi_K}||_{p,\mu} \leq \text{bound}(\text{MDP}, \text{alg}, \mathcal{D}, \mathcal{F}) \quad \textit{w.h.p.}$$

Statistical Learning Theory in ADP

Solution:

- supervised learning methods (regression, classification) appear in the inner-loop of ADP algorithms
- SLT tools used to analyze supervised learning methods can be used in ADP!

Statistical Learning Theory in ADP

Solution:

- supervised learning methods (regression, classification) appear in the inner-loop of ADP algorithms
- SLT tools used to analyze supervised learning methods can be used in ADP!

The specific nature of ADP makes things not so straightforward...

Outline

Linear Fitted Q-iteration

Linear space (used to approximate action–value functions)

$$\mathcal{F} = \Big\{ f(x,a) = \sum_{j=1}^{d} \alpha_j \varphi_j(x,a), \ \ \alpha \in \mathbb{R}^d \Big\}$$

Linear Fitted Q-iteration

Linear space (used to approximate action–value functions)

$$\mathcal{F} = \Big\{ f(x,a) = \sum_{j=1}^{d} \alpha_j \varphi_j(x,a), \quad \alpha \in \mathbb{R}^d \Big\}$$

with features

$$\varphi_j : \mathcal{X} \times \mathcal{A} \to [0, L] \qquad \phi(x,a) = [\varphi_1(x,a) \ldots \varphi_d(x,a)]^\top$$

Linear Fitted Q-iteration

Input: space $\mathcal{F}$, iterations K, sampling distribution ρ, num of samples n

Linear Fitted Q-iteration

Input: space $\mathcal{F}$, iterations K, sampling distribution ρ, num of samples n

Initial function $\widehat{Q}_0 \in \mathcal{F}$

Linear Fitted Q-iteration

Input: space $\mathcal{F}$, iterations K, sampling distribution ρ, num of samples n

Initial function $\widehat{Q}_0 \in \mathcal{F}$
For $k = 1, \dots, K$

Linear Fitted Q-iteration

Input: space $\mathcal{F}$, iterations K, sampling distribution ρ, num of samples n

Initial function $\widehat{Q}_0 \in \mathcal{F}$
For $k = 1, \ldots, K$

- Draw n samples $(x_i, a_i) \overset{\text{i.i.d}}{\sim} \rho$

Linear Fitted Q-iteration

Input: space $\mathcal{F}$, iterations K, sampling distribution ρ, num of samples n

Initial function $\widehat{Q}_0 \in \mathcal{F}$
For $k = 1, \dots, K$

- Draw n samples $(x_i, a_i) \overset{\text{i.i.d}}{\sim} \rho$
- Sample $x'_i \sim p(\cdot|x_i, a_i)$ and $r_i = r(x_i, a_i)$

Linear Fitted Q-iteration

Input: space $\mathcal{F}$, iterations K, sampling distribution ρ, num of samples n

Initial function $\widehat{Q}_0 \in \mathcal{F}$
For $k = 1, \ldots, K$

- Draw n samples $(x_i, a_i) \stackrel{\text{i.i.d}}{\sim} \rho$
- Sample $x_i' \sim p(\cdot|x_i, a_i)$ and $r_i = r(x_i, a_i)$
- Compute $y_i = r_i + \gamma \max_a \widehat{Q}_{k-1}(x_i', a)$

Linear Fitted Q-iteration

Input: space $\mathcal{F}$, iterations K, sampling distribution ρ, num of samples n

Initial function $\widehat{Q}_0 \in \mathcal{F}$
For $k = 1, \ldots, K$

- Draw n samples $(x_i, a_i) \overset{\text{i.i.d}}{\sim} \rho$
- Sample $x_i' \sim p(\cdot|x_i, a_i)$ and $r_i = r(x_i, a_i)$
- Compute $y_i = r_i + \gamma \max_a \widehat{Q}_{k-1}(x_i', a)$
- Build training set $\big\{\big((x_i, a_i), y_i\big)\big\}_{i=1}^n$

Linear Fitted Q-iteration

Input: space $\mathcal{F}$, iterations K, sampling distribution ρ, num of samples n

Initial function $\widehat{Q}_0 \in \mathcal{F}$
For $k = 1, \dots, K$

- Draw n samples $(x_i, a_i) \overset{\text{i.i.d}}{\sim} \rho$
- Sample $x_i' \sim p(\cdot|x_i, a_i)$ and $r_i = r(x_i, a_i)$
- Compute $y_i = r_i + \gamma \max_a \widehat{Q}_{k-1}(x_i', a)$
- Build training set $\big\{\big((x_i, a_i), y_i\big)\big\}_{i=1}^n$
- Solve the *least squares problem*

$$f_{\widehat{\alpha}_k} = \arg\min_{f_\alpha \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^n \big(f_\alpha(x_i, a_i) - y_i\big)^2$$

Linear Fitted Q-iteration

Input: space $\mathcal{F}$, iterations K, sampling distribution ρ, num of samples n

Initial function $\widehat{Q}_0 \in \mathcal{F}$
For $k = 1, \ldots, K$

- Draw n samples $(x_i, a_i) \overset{\text{i.i.d}}{\sim} \rho$
- Sample $x_i' \sim p(\cdot|x_i, a_i)$ and $r_i = r(x_i, a_i)$
- Compute $y_i = r_i + \gamma \max_a \widehat{Q}_{k-1}(x_i', a)$
- Build training set $\Big\{\big((x_i, a_i), y_i\big)\Big\}_{i=1}^n$
- Solve the *least squares problem*

$$f_{\widehat{\alpha}_k} = \arg\min_{f_\alpha \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \big(f_\alpha(x_i, a_i) - y_i\big)^2$$

- Return $\widehat{Q}_k = f_{\widehat{\alpha}_k}$ *(truncation may be needed)*

Linear Fitted Q-iteration

Input: space $\mathcal{F}$, iterations K, sampling distribution ρ, num of samples n

Initial function $\widehat{Q}_0 \in \mathcal{F}$
For $k = 1, \ldots, K$

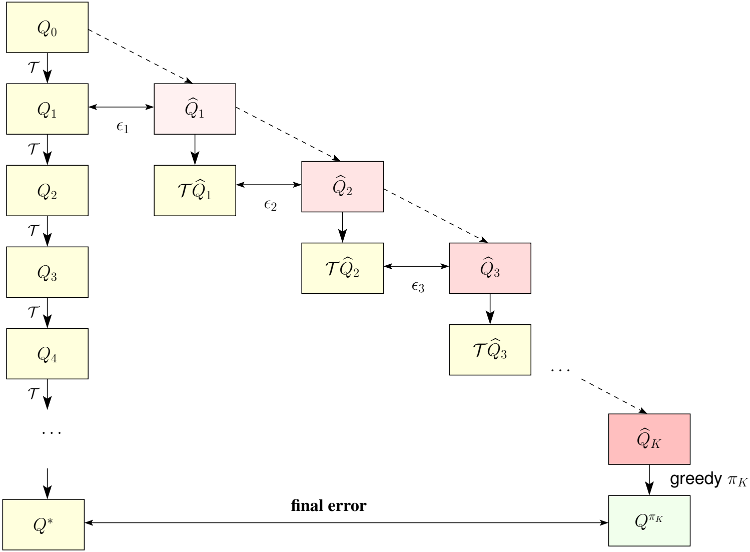
- Draw n samples $(x_i, a_i) \overset{\text{i.i.d}}{\sim} \rho$
- Sample $x_i' \sim p(\cdot|x_i, a_i)$ and $r_i = r(x_i, a_i)$
- Compute $y_i = r_i + \gamma \max_a \widehat{Q}_{k-1}(x_i', a)$
- Build training set $\big\{\big((x_i, a_i), y_i\big)\big\}_{i=1}^n$
- Solve the *least squares problem*

$$f_{\widehat{\alpha}_k} = \arg\min_{f_\alpha \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^n \big(f_\alpha(x_i, a_i) - y_i\big)^2$$

- Return $\widehat{Q}_k = f_{\widehat{\alpha}_k}$ *(truncation may be needed)*

Return $\pi_K(\cdot) = \arg\max_a \widehat{Q}_K(\cdot, a)$ (*greedy policy*)

Sketch of the Analysis



Theoretical Objectives

Objective: derive a bound on the performance (*quadratic*) loss w.r.t. a *testing* distribution μ

$$||Q^* - Q^{\pi_K}||_\mu \leq \ ???$$

Theoretical Objectives

Objective: derive a bound on the performance (*quadratic*) loss w.r.t. a *testing* distribution μ

$$||Q^* - Q^{\pi_K}||_\mu \leq \; ???$$

Sub-Objective 1: derive an *intermediate* bound on the prediction error at *any* iteration k w.r.t. to the *sampling* distribution ρ

$$||\mathcal{T}\widehat{Q}_{k-1} - \widehat{Q}_k||_\rho \leq \; ???$$

Theoretical Objectives

Objective: derive a bound on the performance (*quadratic*) loss w.r.t. a *testing* distribution μ

$$||Q^* - Q^{\pi_K}||_\mu \leq \ ???$$

Sub-Objective 1: derive an *intermediate* bound on the prediction error at *any* iteration k w.r.t. to the *sampling* distribution ρ

$$||\mathcal{T}\widehat{Q}_{k-1} - \widehat{Q}_k||_\rho \leq \ ???$$

Sub-Objective 2: analyze how the *error at each iteration* is *propagated* through iterations

$$||Q^* - Q^{\pi_K}||_\mu \leq \textit{propagation}(||\mathcal{T}\widehat{Q}_{k-1} - \widehat{Q}_k||_\rho)$$

The Sources of Error

- *Desired* solution

$$Q_k = \mathcal{T}\widehat{Q}_{k-1}$$

The Sources of Error

- *Desired* solution

$$Q_k = \mathcal{T}\widehat{Q}_{k-1}$$

- *Best* solution (wrt sampling distribution ρ)

$$f_{\alpha_k^*} = \arg \inf_{f_\alpha \in \mathcal{F}} ||f_\alpha - Q_k||_\rho$$

The Sources of Error

- *Desired* solution

$$Q_k = \mathcal{T}\widehat{Q}_{k-1}$$

- *Best* solution (wrt sampling distribution ρ)

$$f_{\alpha_k^*} = \arg \inf_{f_\alpha \in \mathcal{F}} ||f_\alpha - Q_k||_\rho$$

$\Rightarrow$ ***Error*** from the approximation space $\mathcal{F}$

The Sources of Error

- *Desired* solution

$$Q_k = \mathcal{T}\widehat{Q}_{k-1}$$

- *Best* solution (wrt sampling distribution ρ)

$$f_{\alpha_k^*} = \arg\inf_{f_\alpha \in \mathcal{F}} ||f_\alpha - Q_k||_\rho$$

$\Rightarrow$ ***Error*** from the approximation space $\mathcal{F}$

- *Returned* solution

$$f_{\widehat{\alpha}_k} = \arg\min_{f_\alpha \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \big(f_\alpha(x_i, a_i) - y_i\big)^2$$

The Sources of Error

- *Desired* solution

$$Q_k = \mathcal{T}\widehat{Q}_{k-1}$$

- *Best* solution (wrt sampling distribution ρ)

$$f_{\alpha_k^*} = \arg\inf_{f_\alpha \in \mathcal{F}} ||f_\alpha - Q_k||_\rho$$

$\Rightarrow$ ***Error*** from the approximation space $\mathcal{F}$

- *Returned* solution

$$f_{\widehat{\alpha}_k} = \arg\min_{f_\alpha \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \big(f_\alpha(x_i, a_i) - y_i\big)^2$$

$\Rightarrow$ ***Error*** from the (random) samples

Per-Iteration Error

Theorem

*At each iteration k, Linear-FQI returns an approximation $\widehat{Q}_k$ such that (**Sub-Objective 1**)*

$$
\begin{aligned}
||Q_k - \widehat{Q}_k||_\rho \leq 4||Q_k - f_{\alpha_k^*}||_\rho & \\
& + O\left(\left(V_{\max} + L||\alpha_k^*||\right)\sqrt{\frac{\log 1/\delta}{n}}\right) \\
& + O\left(V_{\max}\sqrt{\frac{d \log n/\delta}{n}}\right),
\end{aligned}
$$

with probability $1-\delta$.

Tools: concentration of measure inequalities, covering space, linear algebra, union bounds, special tricks for linear spaces, ...

Per-Iteration Error

$$
\begin{aligned}
||Q_k - \widehat{Q}_k||_\rho \leq 4||Q_k - f_{\alpha_k^*}||_\rho \\
&+ O\left(\left(V_{\max} + L||\alpha_k^*||\right)\sqrt{\frac{\log 1/\delta}{n}}\right) \\
&+ O\left(V_{\max}\sqrt{\frac{d \log n/\delta}{n}}\right)
\end{aligned}
$$

Per-Iteration Error

$$\begin{aligned}||Q_k - \widehat{Q}_k||_\rho \leq\ & 4||Q_k - f_{\alpha_k^*}||_\rho \\ & + O\Big(\big(V_{\max} + L||\alpha_k^*||\big)\sqrt{\frac{\log 1/\delta}{n}}\Big) \\ & + O\Big(V_{\max}\sqrt{\frac{d\log n/\delta}{n}}\Big)\end{aligned}$$

Remarks

- No algorithm can do better
- Constant 4
- Depends on the space $\mathcal{F}$
- Changes with the iteration k

Per-Iteration Error

$$||Q_k - \widehat{Q}_k||_\rho \leq 4||Q_k - f_{\alpha_k^*}||_\rho + O\bigg(\big(V_{\max} + L||\alpha_k^*||\big)\sqrt{\frac{\log 1/\delta}{n}}\bigg) + O\bigg(V_{\max}\sqrt{\frac{d \log n/\delta}{n}}\bigg)$$

Remarks

- Vanishing to zero as $O(n^{-1/2})$
- Depends on the features (L) and on the best solution ($||\alpha_k^*||$)

Per-Iteration Error

$$
\begin{aligned}
||Q_k - \widehat{Q}_k||_\rho \leq 4||Q_k - f_{\alpha_k^*}||_\rho \\
&+ O\bigg((V_{\max} + L||\alpha_k^*||) \sqrt{\frac{\log 1/\delta}{n}} \bigg) \\
&+ O\bigg(V_{\max} \sqrt{\frac{d \log n/\delta}{n}} \bigg)
\end{aligned}
$$

Remarks

- Vanishing to zero as $O(n^{-1/2})$
- Depends on the dimensionality of the space (d) and the number of samples (n)

Error Propagation

Objective

$$||Q^* - Q^{\pi_K}||_\mu$$

Error Propagation

Objective

$$||Q^* - Q^{\pi_K}||_\mu$$

- **Problem 1**: the test norm μ is different from the sampling norm ρ

Error Propagation

Objective

$$||Q^* - Q^{\pi_K}||_\mu$$

- **Problem 1**: the test norm μ is different from the sampling norm ρ
- **Problem 2**: we have bounds for $\widehat{Q}_k$ not for the performance of the corresponding π_k

Error Propagation

Objective

$$||Q^* - Q^{\pi_K}||_\mu$$

- **Problem 1**: the test norm μ is different from the sampling norm ρ
- **Problem 2**: we have bounds for $\widehat{Q}_k$ not for the performance of the corresponding π_k
- **Problem 3**: we have bounds for one single iteration

Error Propagation

Transition kernel for a fixed policy P^π.

- m-step (worst-case) concentration of future state distribution

$$c(m) = \sup_{\pi_1 \ldots \pi_m} \left\| \frac{d(\mu P^{\pi_1} \ldots P^{\pi_m})}{d\rho} \right\|_\infty < \infty$$

Error Propagation

Transition kernel for a fixed policy P^π.

- m-step (worst-case) concentration of future state distribution

$$c(m) = \sup_{\pi_1 \dots \pi_m} \left\| \frac{d(\mu P^{\pi_1} \dots P^{\pi_m})}{d\rho} \right\|_\infty < \infty$$

- Average (discounted) concentration

$$C_{\mu,\rho} = (1-\gamma)^2 \sum_{m \geq 1} m \gamma^{m-1} c(m) < +\infty$$

Error Propagation

Remark: relationship to top-Lyapunov exponent

$$L^+ = \sup_\pi \lim \sup_{m \to \infty} \frac{1}{m} \log^+ \left(||\rho P^{\pi_1} P^{\pi_2} \cdots P^{\pi_m}|| \right)$$

If $L^+ \leq 0$ (*stable system*), then $c(m)$ has a growth rate which is polynomial and $C_{\mu,\rho} < \infty$ is ***finite***

Error Propagation

Proposition

Let $\varepsilon_k = Q_k - \widehat{Q}_k$ *be the propagation error at each iteration, then after* K *iteration the* performance loss *of the greedy policy* π_K *is*

$$||Q^* - Q^{\pi_K}||_\mu^2 \leq \left[\frac{2\gamma}{(1-\gamma)^2}\right]^2 C_{\mu,\rho} \max_k ||\varepsilon_k||_\rho^2 + O\left(\frac{\gamma^K}{(1-\gamma)^3} V_{\max}^2\right)$$

The Final Bound

Bringing everything together...

$$||Q^* - Q^{\pi_K}||_\mu^2 \leq \left[\frac{2\gamma}{(1-\gamma)^2}\right]^2 C_{\mu,\rho} \max_k ||\varepsilon_k||_\rho^2 + O\left(\frac{\gamma^K}{(1-\gamma)^3} V_{\max}^2\right)$$

The Final Bound

Bringing everything together...

$$||Q^* - Q^{\pi_K}||_\mu^2 \leq \left[\frac{2\gamma}{(1-\gamma)^2}\right]^2 C_{\mu,\rho} \max_k ||\varepsilon_k||_\rho^2 + O\left(\frac{\gamma^K}{(1-\gamma)^3} V_{\max}^2\right)$$

$$\begin{aligned} ||\varepsilon_k||_\rho = ||Q_k - \widehat{Q}_k||_\rho \leq 4||Q_k - f_{\alpha_k^*}||_\rho & \\ + O\left(\left(V_{\max} + L||\alpha_k^*||\right)\sqrt{\frac{\log 1/\delta}{n}}\right) & \\ + O\left(V_{\max}\sqrt{\frac{d \log n/\delta}{n}}\right) & \end{aligned}$$

The Final Bound

Theorem (see e.g., Munos,'03)

LinearFQI with a space $\mathcal{F}$ of d features, with n samples at each iteration returns a policy π_K after K iterations such that

$$||Q^* - Q^{\pi_K}||_\mu \leq \frac{2\gamma}{(1-\gamma)^2}\sqrt{C_{\mu,\rho}}\left(4d(\mathcal{F},\mathcal{T}\mathcal{F}) + O\left(V_{\max}\big(1+\frac{L}{\sqrt{\omega}}\big)\sqrt{\frac{d\log n/\delta}{n}}\right)\right) + O\left(\frac{\gamma^K}{(1-\gamma)^3}V_{\max}^2\right)$$

The Final Bound

Theorem

LinearFQI with a space $\mathcal{F}$ of d features, with n samples at each iteration returns a policy π_K after K iterations such that

$$||Q^* - Q^{\pi_K}||_\mu \leq \frac{2\gamma}{(1-\gamma)^2}\sqrt{C_{\mu,\rho}}\left(4d(\mathcal{F},\mathcal{T}\mathcal{F}) + O\left(V_{\max}\left(1+\frac{L}{\sqrt{\omega}}\right)\sqrt{\frac{d\log n/\delta}{n}}\right)\right) + O\left(\frac{\gamma^K}{(1-\gamma)^3}V_{\max}^2\right)$$

The *propagation* (and different norms) makes the problem *more complex*
$\Rightarrow$ how do we choose the *sampling distribution*?

The Final Bound

Theorem

LinearFQI with a space $\mathcal{F}$ of d features, with n samples at each iteration returns a policy π_K after K iterations such that

$$||Q^* - Q^{\pi_K}||_\mu \leq \frac{2\gamma}{(1-\gamma)^2}\sqrt{C_{\mu,\rho}}\left(4d(\mathcal{F},\mathcal{T}\mathcal{F}) + O\left(V_{\max}\big(1+\frac{L}{\sqrt{\omega}}\big)\sqrt{\frac{d\log n/\delta}{n}}\right)\right) + O\left(\frac{\gamma^K}{(1-\gamma)^3}V_{\max}^2\right)$$

The *approximation* error is *worse* than in regression

The Final Bound

The inherent Bellman error

$$
\begin{aligned}
||Q_k - f_{\alpha_k^*}||_\rho &= \inf_{f \in \mathcal{F}} ||Q_k - f||_\rho \\
&= \inf_{f \in \mathcal{F}} ||\mathcal{T}\widehat{Q}_{k-1} - f||_\rho \\
&\leq \inf_{f \in \mathcal{F}} ||\mathcal{T} f_{\alpha_{k-1}} - f||_\rho \\
&\leq \sup_{g \in \mathcal{F}} \inf_{f \in \mathcal{F}} ||\mathcal{T} g - f||_\rho = d(\mathcal{F}, \mathcal{T}\mathcal{F})
\end{aligned}
$$

Question: how to design $\mathcal{F}$ to make it "compatible" with the Bellman operator?

The Final Bound

Theorem

LinearFQI with a space $\mathcal{F}$ of d features, with n samples at each iteration returns a policy π_K after K iterations such that

$$||Q^* - Q^{\pi_K}||_\mu \leq \frac{2\gamma}{(1-\gamma)^2}\sqrt{C_{\mu,\rho}}\left(4d(\mathcal{F},\mathcal{T}\mathcal{F}) + O\left(V_{\max}\left(1+\frac{L}{\sqrt{\omega}}\right)\sqrt{\frac{d\log n/\delta}{n}}\right)\right) + O\left(\frac{\gamma^K}{(1-\gamma)^3}V_{\max}^2\right)$$

The dependency on γ is worse than at each iteration

$\Rightarrow$ is it possible to *avoid* it?

The Final Bound

Theorem

LinearFQI with a space $\mathcal{F}$ of d features, with n samples at each iteration returns a policy π_K after K iterations such that

$$||Q^* - Q^{\pi_K}||_\mu \leq \frac{2\gamma}{(1-\gamma)^2}\sqrt{C_{\mu,\rho}}\left(4d(\mathcal{F},\mathcal{T}\mathcal{F}) + O\left(V_{\max}\left(1+\frac{L}{\sqrt{\omega}}\right)\sqrt{\frac{d\log n/\delta}{n}}\right)\right) + O\left(\frac{\gamma^K}{(1-\gamma)^3}V_{\max}^2\right)$$

The error decreases exponentially in K

$\Rightarrow K \approx \varepsilon/(1-\gamma)$

The Final Bound

Theorem

LinearFQI with a space $\mathcal{F}$ of d features, with n samples at each iteration returns a policy π_K after K iterations such that

$$||Q^* - Q^{\pi_K}||_\mu \leq \frac{2\gamma}{(1-\gamma)^2}\sqrt{C_{\mu,\rho}}\left(4d(\mathcal{F},\mathcal{T}\mathcal{F}) + O\left(V_{\max}\left(1+\frac{L}{\sqrt{\omega}}\right)\sqrt{\frac{d\log n/\delta}{n}}\right)\right) + O\left(\frac{\gamma^K}{(1-\gamma)^3}V_{\max}^2\right)$$

The smallest eigenvalue of the Gram matrix

$\Rightarrow$ design the features so as to be *orthogonal* w.r.t. ρ

The Final Bound

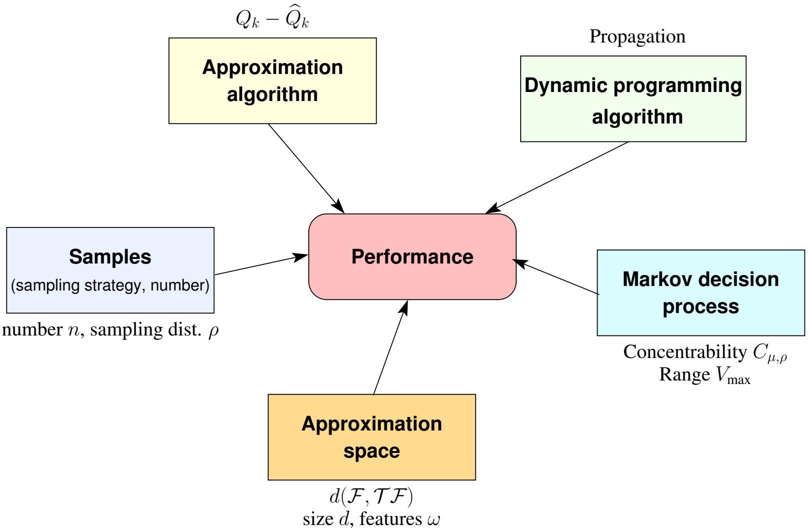
Theorem

LinearFQI with a space $\mathcal{F}$ of d features, with n samples at each iteration returns a policy π_K after K iterations such that

$$||Q^* - Q^{\pi_K}||_\mu \leq \frac{2\gamma}{(1-\gamma)^2}\sqrt{C_{\mu,\rho}}\bigg(4d(\mathcal{F}, \mathcal{T}\mathcal{F}) + O\Big(V_{\max}\big(1+\frac{L}{\sqrt{\omega}}\big)\sqrt{\frac{d\log n/\delta}{n}}\Big)\bigg) + O\Big(\frac{\gamma^K}{(1-\gamma)^3}V_{\max}^2\Big)$$

The asymptotic rate $O(d/n)$ is the same as for regression

Summary


$Q_k - \widehat{Q}_k$
Approximation algorithm
Propagation
Dynamic programming algorithm
Samples
(sampling strategy, number)
number n, sampling dist. ρ
Performance
Markov decision process
Concentrability $C_{\mu,\rho}$
Range $V_{\max}$
Approximation space
$d(\mathcal{F}, \mathcal{T}\mathcal{F})$
size d, features ω

Summary

The STL recipe for ADP:

1. Take your favorite learning algorithm.

Summary

The STL recipe for ADP:

1. Take your favorite learning algorithm.
2. Integrate it into approximate value iteration.

Summary

The STL recipe for ADP:

1. Take your favorite learning algorithm.
2. Integrate it into approximate value iteration.
3. Find a paper proving bounds for the learning algorithm (or prove it yourself)

Summary

The STL recipe for ADP:

1. Take your favorite learning algorithm.
2. Integrate it into approximate value iteration.
3. Find a paper proving bounds for the learning algorithm (or prove it yourself)
4. Plug it in the error propagation bound.

Summary

The STL recipe for ADP:

1. Take your favorite learning algorithm.
2. Integrate it into approximate value iteration.
3. Find a paper proving bounds for the learning algorithm (or prove it yourself)
4. Plug it in the error propagation bound.
5. Enjoy your final bound!

Summary

The STL recipe for ADP:

1. Take your favorite learning algorithm.
2. Integrate it into approximate value iteration.
3. Find a paper proving bounds for the learning algorithm (or prove it yourself)
4. Plug it in the error propagation bound.
5. Enjoy your final bound!

Not always so easy...

Outline

Least-Squares Policy Iteration (LSPI)

LSPI uses

- Linear space to approximate value functions

$$\mathcal{F} = \Big\{ f(x) = \sum_{j=1}^{d} \alpha_j \varphi_j(x), \;\; \alpha \in \mathbb{R}^d \Big\}$$

Least-Squares Policy Iteration (LSPI)

LSPI uses

- Linear space to approximate value functions

$$\mathcal{F} = \Big\{ f(x) = \sum_{j=1}^{d} \alpha_j \varphi_j(x), \ \ \alpha \in \mathbb{R}^d \Big\}$$

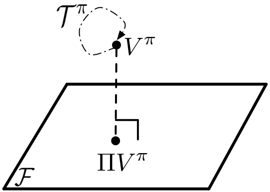
- Least-Squares Temporal Difference (LSTD) algorithm for *policy evaluation*.

Least-Squares Temporal-Difference Learning (LSTD)

- V^π is the fixed-point of $\mathcal{T}^\pi$ $\qquad V^\pi = \mathcal{T}^\pi V^\pi$
- V^π may not belong to $\mathcal{F}$ $\qquad V^\pi \notin \mathcal{F}$
- Best approximation of V^π in $\mathcal{F}$ is

$$\Pi V^\pi = \arg\min_{f \in \mathcal{F}} ||V^\pi - f||$$

(Π is the projection onto $\mathcal{F}$)



Least-Squares Temporal-Difference Learning (LSTD)

- LSTD searches for the fixed-point of $\Pi_? \mathcal{T}^\pi$ instead ($\Pi_?$ is a projection into $\mathcal{F}$ w.r.t. $L_?$-norm)

Least-Squares Temporal-Difference Learning (LSTD)

- LSTD searches for the fixed-point of $\Pi_? \mathcal{T}^\pi$ instead ($\Pi_?$ is a projection into $\mathcal{F}$ w.r.t. $L_?$-norm)

- $\Pi_\infty \mathcal{T}^\pi$ is a contraction in L_∞-norm
  - L_∞-projection is numerically expensive when the number of states is large or infinite

Least-Squares Temporal-Difference Learning (LSTD)

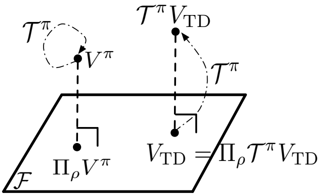
- LSTD searches for the fixed-point of $\Pi_? \mathcal{T}^\pi$ instead ($\Pi_?$ is a projection into $\mathcal{F}$ w.r.t. $L_?$-norm)
- $\Pi_\infty \mathcal{T}^\pi$ is a contraction in L_∞-norm
  - L_∞-projection is numerically expensive when the number of states is large or infinite
- LSTD searches for the fixed-point of $\Pi_{2,\rho} \mathcal{T}^\pi$

$$\Pi_{2,\rho}\, g = \arg\min_{f \in \mathcal{F}} ||g - f||_{2,\rho}$$

Least-Squares Temporal-Difference Learning (LSTD)

When the fixed-point of $\Pi_\rho \mathcal{T}^\pi$ exists, we call it the LSTD solution

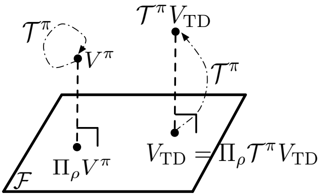
$$V_{\mathsf{TD}} = \Pi_\rho \mathcal{T}^\pi V_{\mathsf{TD}}$$



Least-Squares Temporal-Difference Learning (LSTD)

When the fixed-point of $\Pi_\rho \mathcal{T}^\pi$ exists, we call it the LSTD solution

$$V_{\mathsf{TD}} = \Pi_\rho \mathcal{T}^\pi V_{\mathsf{TD}}$$



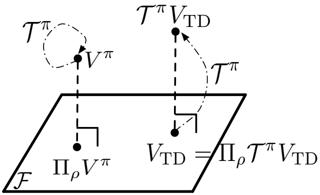
$$\langle \mathcal{T}^\pi V_{\mathsf{TD}} - V_{\mathsf{TD}}, \varphi_i \rangle_\rho = 0, \qquad i = 1, \ldots, d$$

$$\underbrace{\langle r^\pi, \varphi_i \rangle_\rho}_{b_i} - \sum_{i=1}^{d} \underbrace{\langle \varphi_j - \gamma P^\pi \varphi_j, \varphi_i \rangle_\rho}_{A_{ij}} \cdot \alpha_{\mathsf{TD}}^{(j)} = 0 \quad \Longrightarrow \quad A\, \alpha_{\mathsf{TD}} = b$$

Least-Squares Temporal-Difference Learning (LSTD)

When the fixed-point of $\Pi_\rho \mathcal{T}^\pi$ exists, we call it the LSTD solution

$$V_{\mathsf{TD}} = \Pi_\rho \mathcal{T}^\pi V_{\mathsf{TD}}$$

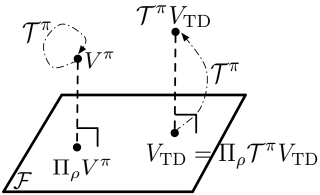


- **Problem:** In general, $\Pi_\rho \mathcal{T}^\pi$ is ***not a contraction*** and does not have a fixed-point.
- **Solution:** If $\rho = \rho^\pi$ (*stationary dist. of π*) then $\Pi_{\rho^\pi} \mathcal{T}^\pi$ has a unique fixed-point.

Least-Squares Temporal-Difference Learning (LSTD)

When the fixed-point of $\Pi_\rho \mathcal{T}^\pi$ exists, we call it the LSTD solution

$$V_{\mathsf{TD}} = \Pi_\rho \mathcal{T}^\pi V_{\mathsf{TD}}$$



- **Problem:** In general, $\Pi_\rho \mathcal{T}^\pi$ cannot be computed (because ***unknown***)
- **Solution:** Use *samples* coming from a "trajectory" of π.

Least-Squares Policy Iteration (LSPI)

Input: space $\mathcal{F}$, iterations K, sampling distribution ρ, num of samples n

Least-Squares Policy Iteration (LSPI)

Input: space $\mathcal{F}$, iterations K, sampling distribution ρ, num of samples n

Initial policy π_0

Least-Squares Policy Iteration (LSPI)

Input: space $\mathcal{F}$, iterations K, sampling distribution ρ, num of samples n

Initial policy π_0
For $k = 1, \ldots, K$

Least-Squares Policy Iteration (LSPI)

Input: space $\mathcal{F}$, iterations K, sampling distribution ρ, num of samples n

Initial policy π_0
For $k = 1, \ldots, K$

- Generate a trajectory of length n from the stationary dist. ρ^{π_k}

$$(x_1, \pi_k(x_1), r_1, x_2, \pi_k(x_2), r_2, \ldots, x_{n-1}, \pi_k(x_{n-1}), r_{n-1}, x_n)$$

Least-Squares Policy Iteration (LSPI)

Input: space $\mathcal{F}$, iterations K, sampling distribution ρ, num of samples n

Initial policy π_0
For $k = 1, \ldots, K$

- Generate a trajectory of length n from the stationary dist. ρ^{π_k}

$$(x_1, \pi_k(x_1), r_1, x_2, \pi_k(x_2), r_2, \ldots, x_{n-1}, \pi_k(x_{n-1}), r_{n-1}, x_n)$$

- Compute the empirical matrix $\widehat{A}_k$ and the vector $\widehat{b}_k$ and solve the linear system $\alpha_k = \widehat{A}_k^{-1}\widehat{b}_k$

Least-Squares Policy Iteration (LSPI)

Input: space $\mathcal{F}$, iterations K, sampling distribution ρ, num of samples n

Initial policy π_0
For $k = 1, \ldots, K$

- Generate a trajectory of length n from the stationary dist. ρ^{π_k}

$$(x_1, \pi_k(x_1), r_1, x_2, \pi_k(x_2), r_2, \ldots, x_{n-1}, \pi_k(x_{n-1}), r_{n-1}, x_n)$$

- Compute the empirical matrix $\widehat{A}_k$ and the vector $\widehat{b}_k$ and solve the linear system $\alpha_k = \widehat{A}_k^{-1}\widehat{b}_k$
- Compute the greedy policy π_{k+1} w.r.t. $\widehat{V}_k = f_{\alpha_k}$

Least-Squares Policy Iteration (LSPI)

Input: space $\mathcal{F}$, iterations K, sampling distribution ρ, num of samples n

Initial policy π_0
For $k = 1, \ldots, K$

- Generate a trajectory of length n from the stationary dist. ρ^{π_k}

$$(x_1, \pi_k(x_1), r_1, x_2, \pi_k(x_2), r_2, \ldots, x_{n-1}, \pi_k(x_{n-1}), r_{n-1}, x_n)$$

- Compute the empirical matrix $\widehat{A}_k$ and the vector $\widehat{b}_k$ and solve the linear system $\alpha_k = \widehat{A}_k^{-1}\widehat{b}_k$
- Compute the greedy policy π_{k+1} w.r.t. $\widehat{V}_k = f_{\alpha_k}$

Return the last policy π_K

LSTD Algorithm

When $n \to \infty$ then $\widehat{A} \to A$ and $\widehat{b} \to b$, and thus,

$$\widehat{\alpha}_{\mathsf{TD}} \to \alpha_{\mathsf{TD}} \text{ and } \widehat{V}_{\mathsf{TD}} \to V_{\mathsf{TD}}$$

Proposition (LSTD Performance)

If LSTD is used to estimate the value of π with an ***infinite*** number of samples drawn from the stationary distribution ρ^π then

$$||V^\pi - V_{\mathsf{TD}}||_{\rho^\pi} \leq \frac{1}{\sqrt{1-\gamma^2}} \inf_{V \in \mathcal{F}} ||V^\pi - V||_{\rho^\pi}$$

LSTD Algorithm

When $n \to \infty$ then $\widehat{A} \to A$ and $\widehat{b} \to b$, and thus,

$$\widehat{\alpha}_{\text{TD}} \to \alpha_{\text{TD}} \text{ and } \widehat{V}_{\text{TD}} \to V_{\text{TD}}$$

Proposition (LSTD Performance)

If LSTD is used to estimate the value of π with an ***infinite*** number of samples drawn from the stationary distribution ρ^π then

$$||V^\pi - V_{\text{TD}}||_{\rho^\pi} \leq \frac{1}{\sqrt{1-\gamma^2}} \inf_{V \in \mathcal{F}} ||V^\pi - V||_{\rho^\pi}$$

Problem: we don't have an infinite number of samples...

LSTD Algorithm

When $n \to \infty$ then $\widehat{A} \to A$ and $\widehat{b} \to b$, and thus,

$$\widehat{\alpha}_{\mathsf{TD}} \to \alpha_{\mathsf{TD}} \text{ and } \widehat{V}_{\mathsf{TD}} \to V_{\mathsf{TD}}$$

Proposition (LSTD Performance)

If LSTD is used to estimate the value of π with an ***infinite*** number of samples drawn from the stationary distribution ρ^π then

$$||V^\pi - V_{\mathsf{TD}}||_{\rho^\pi} \leq \frac{1}{\sqrt{1-\gamma^2}} \inf_{V \in \mathcal{F}} ||V^\pi - V||_{\rho^\pi}$$

Problem: we don't have an infinite number of samples...
Problem 2: V_{TD} is a fixed point solution and not a standard machine learning problem...

LSTD Error Bound

Assumption: The Markov chain induced by the policy π_k has a stationary distribution ρ^{π_k} and it is ergodic and β-mixing.

LSTD Error Bound

Assumption: The Markov chain induced by the policy π_k has a stationary distribution ρ^{π_k} and it is ergodic and β-mixing.

Theorem (LSTD Error Bound)

At any iteration k, if LSTD uses n samples obtained from a single trajectory of π and a d-dimensional space, then with probability $1-\delta$

$$||V^{\pi_k} - \widehat{V}_k||_{\rho^{\pi_k}} \leq \frac{c}{\sqrt{1-\gamma^2}} \inf_{f\in\mathcal{F}} ||V^{\pi_k} - f||_{\rho^{\pi_k}} + O\left(\sqrt{\frac{d\log(d/\delta)}{n\,\nu}}\right)$$

LSTD Error Bound

$$||V^\pi - \widehat{V}||_{\rho^\pi} \leq \frac{c}{\sqrt{1-\gamma^2}} \underbrace{\inf_{f \in \mathcal{F}} ||V^\pi - f||_{\rho^\pi}}_{\text{approximation error}} + \underbrace{O\left(\sqrt{\frac{d \log(d/\delta)}{n\,\nu}}\right)}_{\text{estimation error}}$$

- **Approximation error:** it depends on how well the function space $\mathcal{F}$ can approximate the value function V^π
- **Estimation error:** it depends on the number of samples n, the dim of the function space d, the smallest eigenvalue of the Gram matrix ν, the mixing properties of the Markov chain (hidden in O)

LSTD Error Bound

$$||V^{\pi_k} - \widehat{V}_k||_{\rho^{\pi_k}} \leq \frac{c}{\sqrt{1-\gamma^2}} \underbrace{\inf_{f \in \mathcal{F}} ||V^{\pi_k} - f||_{\rho^{\pi_k}}}_{\text{approximation error}} + \underbrace{O\left(\sqrt{\frac{d \log(d/\delta)}{n\, \nu_k}}\right)}_{\text{estimation error}}$$

- n number of samples and d dimensionality

LSTD Error Bound

$$||V^{\pi_k} - \widehat{V}_k||_{\rho^{\pi_k}} \leq \frac{c}{\sqrt{1-\gamma^2}} \underbrace{\inf_{f \in \mathcal{F}} ||V^{\pi_k} - f||_{\rho^{\pi_k}}}_{\text{approximation error}} + \underbrace{O\left(\sqrt{\frac{d \log(d/\delta)}{n \, \nu_k}}\right)}_{\text{estimation error}}$$

- $\nu_k =$ the smallest eigenvalue of the Gram matrix $(\int \varphi_i \, \varphi_j \, d\rho^{\pi_k})_{i,j}$ (**Assumption:** *eigenvalues of the Gram matrix are strictly positive - existence of the model-based LSTD solution*)
- β-mixing coefficients are hidden in the $O(\cdot)$ notation

LSPI Error Bound

Theorem (LSPI Error Bound)

If LSPI is run over K iterations, then the performance loss policy π_K is

$$||V^* - V^{\pi_K}||_\mu \leq \frac{4\gamma}{(1-\gamma)^2} \left\{ \sqrt{CC_{\mu,\rho}} \left[E_0(\mathcal{F}) + O\left(\sqrt{\frac{d \log(dK/\delta)}{n\,\nu_\rho}} \right) \right] + \gamma^K R_{\max} \right\}$$

with probability $1 - \delta$.

LSPI Error Bound

Theorem (LSPI Error Bound)

If LSPI is run over K iterations, then the performance loss policy π_K is

$$||V^* - V^{\pi_K}||_\mu \leq \frac{4\gamma}{(1-\gamma)^2} \left\{ \sqrt{CC_{\mu,\rho}} \left[cE_0(\mathcal{F}) + O\left(\sqrt{\frac{d \log(dK/\delta)}{n\, \nu_\rho}} \right) \right] + \gamma^K R_{\max} \right\}$$

with probability $1 - \delta$.

- **Approximation error:** $E_0(\mathcal{F}) = \sup_{\pi \in \mathcal{G}(\widetilde{\mathcal{F}})} \inf_{f \in \mathcal{F}} ||V^\pi - f||_{\rho^\pi}$

LSPI Error Bound

Theorem (LSPI Error Bound)

If LSPI is run over K iterations, then the performance loss policy π_K is

$$||V^* - V^{\pi_K}||_\mu \leq \frac{4\gamma}{(1-\gamma)^2} \left\{ \sqrt{CC_{\mu,\rho}} \left[cE_0(\mathcal{F}) + O\left(\sqrt{\frac{d \log(dK/\delta)}{n\ \nu_\rho}} \right) \right] + \gamma^K R_{\max} \right\}$$

with probability $1 - \delta$.

- **Approximation error:** $E_0(\mathcal{F}) = \sup_{\pi \in \mathcal{G}(\widetilde{\mathcal{F}})} \inf_{f \in \mathcal{F}} ||V^\pi - f||_{\rho^\pi}$
- **Estimation error:** depends on n, d, ν_ρ, K

LSPI Error Bound

Theorem (LSPI Error Bound)

If LSPI is run over K iterations, then the performance loss policy π_K is

$$||V^* - V^{\pi_K}||_\mu \leq \frac{4\gamma}{(1-\gamma)^2} \left\{ \sqrt{CC_{\mu,\rho}} \left[cE_0(\mathcal{F}) + O\left(\sqrt{\frac{d \log(dK/\delta)}{n\, \nu_\rho}} \right) \right] + \gamma^K R_{\max} \right\}$$

with probability $1-\delta$.

- **Approximation error:** $E_0(\mathcal{F}) = \sup_{\pi \in \mathcal{G}(\widetilde{\mathcal{F}})} \inf_{f \in \mathcal{F}} ||V^\pi - f||_{\rho^\pi}$
- **Estimation error:** depends on n, d, ν_ρ, K
- **Initialization error:** error due to the choice of the initial value function or initial policy $|V^* - V^{\pi_0}|$

LSPI Error Bound

LSPI Error Bound

$$||V^* - V^{\pi_K}||_\mu \leq \frac{4\gamma}{(1-\gamma)^2}\left\{\sqrt{CC_{\mu,\rho}}\left[cE_0(\mathcal{F}) + O\left(\sqrt{\frac{d\log(dK/\delta)}{n\,\nu_\rho}}\right)\right] + \gamma^K R_{\max}\right\}$$

Lower-Bounding Distribution

There exists a distribution ρ such that for any policy $\pi \in \mathcal{G}(\widetilde{\mathcal{F}})$, we have $\rho \leq C\rho^\pi$, where $C < \infty$ is a constant and ρ^π is the stationary distribution of π. Furthermore, we can define the concentrability coefficient $C_{\mu,\rho}$ as before.

LSPI Error Bound

LSPI Error Bound

$$||V^* - V^{\pi_K}||_\mu \leq \frac{4\gamma}{(1-\gamma)^2} \left\{ \sqrt{CC_{\mu,\rho}} \left[cE_0(\mathcal{F}) + O\left(\sqrt{\frac{d \log(dK/\delta)}{n\, \nu_\rho}} \right) \right] + \gamma^K R_{\max} \right\}$$

Lower-Bounding Distribution

There exists a distribution ρ such that for any policy $\pi \in \mathcal{G}(\widetilde{\mathcal{F}})$, we have $\rho \leq C\rho^\pi$, where $C < \infty$ is a constant and ρ^π is the stationary distribution of π. Furthermore, we can define the concentrability coefficient $C_{\mu,\rho}$ as before.

- $\nu_\rho =$ the smallest eigenvalue of the Gram matrix $(\int \varphi_i\, \varphi_j\, d\rho)_{i,j}$

Outline

Other Finite-Sample Analysis Results in ADP

Approximate Value Iteration

- Fitted value iteration ***(Munos & Szepesvari 2008)***
- L_2-Regularized Fitted Q-Iteration ***(Farahmand et al. 2009)***
- Transfer of samples in Fitted Q-Iteration ***(L, Restelli, 2010)***
- Multi-task Sparse Fitted Q-Iteration ***(Calandriello, L, Restelli, 2014)***

Other Finite-Sample Analysis Results in ADP

Approximate Policy Iteration

- LSTD and LSPI ***(L, Ghavamzadeh, Munos 2010, 2012)***
- Bellman Residual Minimization ***(Maillard, Munos, L, Ghavamzadeh 2010)***
- Modified Bellman Residual Minimization ***(Antos et al. 2008)***
- Classification-based Policy Iteration ***(Fern et al. 2006; L, Ghavamzadeh, Munos et al. 2010; Gabillon, L, Ghavamzadeh, Scherrer 2011)***
- Conservative Policy Iteration ***(Kakade & Langford 2002; Kakade 2003)***
- ℓ_1-regularize LSTD ***(Ghavamzadeh, L, Munos, 2011, Hoffman, L, Ghavamzadeh, Munos, 2012, Geist, Scherrer, L, Ghavamzadeh, 2012)***
- LSTD (LSPI) with Random Projections ***(Ghavamzadeh, L, Maillard, Munos, 2010)***

Comparison to Supervised Learning

Similarity: The convergence rate is the same (optimal) rate of statistical learning theory.

Difference

- dependency on $1/(1-\gamma)$ (sequential nature of the problem)
- the approximation error is more complex (iterative nature of the algorithms)
- the propagation of error (concentrability) (control problem)
- the sampling problem (how to choose ρ – exploration problem)

Practical Lessons

- Tuning the parameters (given a fixed accuracy ϵ)
  - number of samples (inverting the bound) $n \geq \widetilde{\Omega}(\frac{d}{\epsilon})$
  - number of iterations (inverting the bound) $K \approx \epsilon/(1-\gamma)$
- choice of function $\mathcal{F}$ and/or policy space Π
  - features $\{\varphi_i\}_{i=1}^d$ to be linearly independent given the sampling distribution ρ (on-policy – off-policy sampling)
- tradeoff between approximation and estimation errors

Open Problems

Control the propagation of error

- Improve the *sampling* distribution
- Refine the analysis of *concentrability* terms
- Off-policy learning
- No-regret algorithms
- Find “easier” MDPs

Control the approximation error

- Non-parametric approaches
- Smooth MDPs
- Automatic construction of basis functions
- Representation learning

Approximate Dynamic Programming meets Statistical Learning Theory

A. Lazaric
{alessandro.lazaric}@inria.fr
sequel.lille.inria.fr