Approximate Dynamic Programming

A. LAZARIC (*SequeL Team @INRIA-Lille*)
ENS Cachan - Master 2 MVA

SequeL – INRIA Lille

Approximate Dynamic Programming

(a.k.a. Batch Reinforcement Learning)

Approximate Dynamic Programming

(a.k.a. Batch Reinforcement Learning)

Approximate Value Iteration

Approximate Policy Iteration

From DP to ADP

- Dynamic programming algorithms require an *explicit* definition of
  - transition probabilities $p(\cdot|x, a)$
  - reward function $r(x, a)$

From DP to ADP

- Dynamic programming algorithms require an *explicit* definition of
  - transition probabilities $p(\cdot|x, a)$
  - reward function $r(x, a)$
- This knowledge is often *unavailable* (i.e., wind intensity, human-computer-interaction).

From DP to ADP

- Dynamic programming algorithms require an *explicit* definition of
  - transition probabilities $p(\cdot|x, a)$
  - reward function $r(x, a)$
- This knowledge is often *unavailable* (i.e., wind intensity, human-computer-interaction).
- ***Can we rely on samples?***

From DP to ADP

- Dynamic programming algorithms require an *exact* representation of value functions and policies

From DP to ADP

- Dynamic programming algorithms require an *exact* representation of value functions and policies
- This is often *impossible* since their shape is too “complicated” (e.g., large or continuous state space).

From DP to ADP

- Dynamic programming algorithms require an *exact* representation of value functions and policies
- This is often *impossible* since their shape is too “complicated” (e.g., large or continuous state space).
- ***Can we use approximations?***

The Objective

Find a policy π such that

the *performance loss* $||V^* - V^\pi||$ is as small as possible

From Approximation Error to Performance Loss

Question: if V is an approximation of the optimal value function V^* with an error

$$\text{error} = \|V - V^*\|$$

From Approximation Error to Performance Loss

Question: if V is an approximation of the optimal value function V^* with an error

$$\text{error} = \|V - V^*\|$$

how does it translate to the (loss of) performance of the *greedy policy*

$$\pi(x) \in \arg\max_{a \in A} \sum_y p(y|x,a)\big[r(x,a,y) + \gamma V(y)\big]$$

From Approximation Error to Performance Loss

Question: if V is an approximation of the optimal value function V^* with an error

$$\text{error} = \| V - V^* \|$$

how does it translate to the (loss of) performance of the *greedy policy*

$$\pi(x) \in \arg\max_{a \in A} \sum_y p(y|x,a)\big[r(x,a,y) + \gamma V(y)\big]$$

i.e.

$$\text{performance loss} = \| V^* - V^{\pi} \|$$

From Approximation Error to Performance Loss

Proposition

Let $V \in \mathbb{R}^N$ be an approximation of V^* and π its corresponding greedy policy, then

$$\underbrace{\|V^* - V^\pi\|_\infty}_{\textit{performance loss}} \leq \frac{2\gamma}{1-\gamma} \underbrace{\|V^* - V\|_\infty}_{\textit{approx. error}}.$$

Furthermore, there exists $\epsilon > 0$ such that if $\|V - V^*\|_\infty \leq \epsilon$, then π is *optimal*.

From Approximation Error to Performance Loss

Proof.

$$\begin{aligned}
\|V^* - V^\pi\|_\infty &\leq \|\mathcal{T}V^* - \mathcal{T}^\pi V\|_\infty + \|\mathcal{T}^\pi V - \mathcal{T}^\pi V^\pi\|_\infty \\
&\leq \|\mathcal{T}V^* - \mathcal{T}V\|_\infty + \gamma\|V - V^\pi\|_\infty \\
&\leq \gamma\|V^* - V\|_\infty + \gamma(\|V - V^*\|_\infty + \|V^* - V^\pi\|_\infty) \\
&\leq \frac{2\gamma}{1-\gamma}\|V^* - V\|_\infty.
\end{aligned}$$

■

Approximate Dynamic Programming

(a.k.a. Batch Reinforcement Learning)

Approximate Value Iteration

Approximate Policy Iteration

From Approximation Error to Performance Loss

Question: how do we compute a *good* V?

From Approximation Error to Performance Loss

Question: how do we compute a *good* V?

Problem: unlike in standard approximation scenarios (see supervised learning), we have a *limited access* to the target function, i.e. V^*.

From Approximation Error to Performance Loss

Question: how do we compute a *good* V?

Problem: unlike in standard approximation scenarios (see supervised learning), we have a *limited access* to the target function, i.e. V^*.

Solution: value iteration tends to learn functions which are *close to the optimal value function* V^*.

Value Iteration: the Idea

1. Let Q_0 be *any* action-value function
2. At each iteration $k = 1, 2, \ldots, K$
   - Compute
     $$Q_{k+1}(x,a) = \mathcal{T} Q_k(x,a) = r(x,a) + \sum_y p(y|x,a)\gamma \max_b Q_k(y,b)$$
3. Return the *greedy* policy

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

Value Iteration: the Idea

1. Let Q_0 be *any* action-value function
2. At each iteration $k = 1, 2, \ldots, K$
   - Compute
     $$Q_{k+1}(x,a) = \mathcal{T} Q_k(x,a) = r(x,a) + \sum_y p(y|x,a)\gamma \max_b Q_k(y,b)$$
3. Return the *greedy* policy

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

- ***Problem***: how can we approximate $\mathcal{T} Q_k$?
- ***Problem***: if $Q_{k+1} \neq \mathcal{T} Q_k$, does (approx.) value iteration still work?

Linear Fitted Q-iteration: the Approximation Space

Linear space (used to approximate action–value functions)

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

Linear Fitted Q-iteration: the Approximation Space

Linear space (used to approximate action–value functions)

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

with features

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

Linear Fitted Q-iteration: the Samples

Assumption: access to a ***generative model***, that is a black-box simulator *sim()* of the environment is available.
Given (x, a),

$$\mathrm{sim}(x, a) = \{y, r\}, \quad \text{with } y \sim p(\cdot|x, a), r = r(x, a)$$

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 $k = 1, \dots, K$

1. 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 $k = 1, \ldots, K$
    1. Draw $n$ samples $(x_i, a_i) \overset{\text{i.i.d}}{\sim} \rho$
    2. 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 $k = 1, \dots, K$
  1. Draw $n$ samples $(x_i, a_i) \overset{\text{i.i.d}}{\sim} \rho$
  2. Sample $x_i' \sim p(\cdot|x_i, a_i)$ and $r_i = r(x_i, a_i)$
  3. 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 $k = 1, \ldots, K$
  1. Draw $n$ samples $(x_i, a_i) \overset{\text{i.i.d}}{\sim} \rho$
  2. Sample $x'_i \sim p(\cdot|x_i, a_i)$ and $r_i = r(x_i, a_i)$
  3. Compute $y_i = r_i + \gamma \max_a \widehat{Q}_{k-1}(x'_i, a)$
  4. 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 $k = 1, \ldots, K$

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

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

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 $k = 1, \ldots, K$
    1. Draw $n$ samples $(x_i, a_i) \overset{\text{i.i.d}}{\sim} \rho$
    2. Sample $x_i' \sim p(\cdot|x_i, a_i)$ and $r_i = r(x_i, a_i)$
    3. Compute $y_i = r_i + \gamma \max_a \widehat{Q}_{k-1}(x_i', a)$
    4. Build training set $\{((x_i, a_i), y_i)\}_{i=1}^n$
    5. Solve the *least squares problem*
       $f_{\hat{\alpha}_k} = \arg\min_{f_\alpha \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} (f_\alpha(x_i, a_i) - y_i)^2$
    6. Return $\widehat{Q}_k = f_{\hat{\alpha}_k}$ *(truncation may be needed)*
```

Linear Fitted Q-iteration

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

Initial function $\widehat{Q}_0 \in \mathcal{F}$
For $k = 1, \dots, K$
    1. Draw $n$ samples $(x_i, a_i) \overset{\text{i.i.d}}{\sim} \rho$
    2. Sample $x'_i \sim p(\cdot|x_i, a_i)$ and $r_i = r(x_i, a_i)$
    3. Compute $y_i = r_i + \gamma \max_a \widehat{Q}_{k-1}(x'_i, a)$
    4. Build training set $\big\{\big((x_i, a_i), y_i\big)\big\}_{i=1}^n$
    5. Solve the *least squares problem*
       $$f_{\hat\alpha_k} = \arg\min_{f_\alpha \in \mathcal{F}} \frac{1}{n}\sum_{i=1}^{n}\big(f_\alpha(x_i, a_i) - y_i\big)^2$$
    6. Return $\widehat{Q}_k = f_{\hat\alpha_k}$ *(truncation may be needed)*
**Return** $\pi_K(\cdot) = \arg\max_a \widehat{Q}_K(\cdot, a)$ (*greedy policy*)
```

Linear Fitted Q-iteration: Sampling

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

Linear Fitted Q-iteration: Sampling

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

- In practice it can be done *once* before running the algorithm
- The sampling distribution ρ should cover the state-action space in all *relevant* regions
- If not possible to choose ρ, a *database* of samples can be used

Linear Fitted Q-iteration: The Training Set

4. Compute $y_i = r_i + \gamma \max_a \widehat{Q}_{k-1}(x'_i, a)$
5. Build training set $\big\{\big((x_i, a_i), y_i\big)\big\}_{i=1}^n$

Linear Fitted Q-iteration: The Training Set

4. Compute $y_i = r_i + \gamma \max_a \widehat{Q}_{k-1}(x_i', a)$
5. Build training set $\big\{\big((x_i, a_i), y_i\big)\big\}_{i=1}^n$

- Each sample y_i is an unbiased sample, since

$$\begin{aligned}\mathbb{E}[y_i|x_i, a_i] &= \mathbb{E}[r_i + \gamma \max_a \widehat{Q}_{k-1}(x_i', a)] = r(x_i, a_i) + \gamma\mathbb{E}[\max_a \widehat{Q}_{k-1}(x_i', a)] \\ &= r(x_i, a_i) + \gamma \int_X \max_a \widehat{Q}_{k-1}(x', a) p(dy|x, a) = \mathcal{T}\widehat{Q}_{k-1}(x_i, a_i)\end{aligned}$$

- The problem “reduces” to standard *regression*
- It should be recomputed at each iteration

Linear Fitted Q-iteration: The Regression Problem

6. Solve the *least squares problem*

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

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

Linear Fitted Q-iteration: The Regression Problem

6. Solve the *least squares problem*

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

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

- Thanks to the linear space we can solve it as
  - Build matrix $\Phi = \big[\phi(x_1, a_1)^\top \dots \phi(x_n, a_n)^\top\big]$
  - Compute $\hat{\alpha}^k = (\Phi^\top \Phi)^{-1} \Phi^\top y$ (*least–squares solution*)
- Truncation to $[-V_{\max}; V_{\max}]$ (with $V_{\max} = R_{\max}/(1-\gamma)$)

Sketch of the Analysis

$$\begin{tikzcd}
Q_0 \arrow[d, "\mathcal{T}"'] \arrow[dr, dashed] & & & & & \\
Q_1 \arrow[d, "\mathcal{T}"'] \arrow[r, leftrightarrow, "\epsilon_1"'] & \widehat{Q}_1 \arrow[d] \arrow[dr, dashed] & & & & \\
Q_2 \arrow[d, "\mathcal{T}"'] & \mathcal{T}\widehat{Q}_1 \arrow[r, leftrightarrow, "\epsilon_2"'] & \widehat{Q}_2 \arrow[d] \arrow[dr, dashed] & & & \\
Q_3 \arrow[d, "\mathcal{T}"'] & & \mathcal{T}\widehat{Q}_2 \arrow[r, leftrightarrow, "\epsilon_3"'] & \widehat{Q}_3 \arrow[d] & & \\
Q_4 \arrow[d, "\mathcal{T}"'] & & & \mathcal{T}\widehat{Q}_3 & \cdots \arrow[dr, dashed] & \\
\cdots \arrow[d] & & & & & \widehat{Q}_K \arrow[d, "\text{greedy } \pi_K"] \\
Q^* \arrow[rrrrr, leftrightarrow, "\textbf{final error}"] & & & & & Q^{\pi_K}
\end{tikzcd}$$

▸ Skip Theory

Theoretical Objectives

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

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

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

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

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

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

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

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

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

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

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

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

$\Rightarrow$ ***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**)*

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

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

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

Remarks

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

Per-Iteration Error

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

Remarks

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

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

The Final Bound

Bringing everything together...

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

The Final Bound

Bringing everything together...

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

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

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

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

The 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

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

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

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

The error decreases exponentially in K

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

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

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

The smallest eigenvalue of the Gram matrix

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

The Final Bound

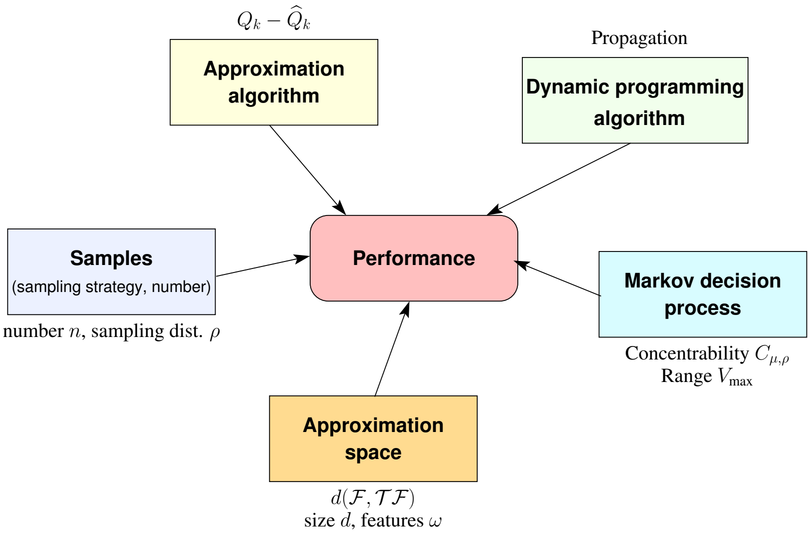
Theorem

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

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

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

Summary



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

Other implementations

Replace the *regression* step with

- K-nearest neighbour
- Regularized linear regression with L_1 or L_2 regularisation
- Neural network
- Support vector regression
- ...

Example: the Optimal Replacement Problem

State: level of wear of an object (e.g., a car).

Example: the Optimal Replacement Problem

State: level of wear of an object (e.g., a car).
Action: $\{(R)\text{eplace}, (K)\text{eep}\}$.

Example: the Optimal Replacement Problem

State: level of wear of an object (e.g., a car).
Action: $\{(R)\text{eplace}, (K)\text{eep}\}$.
Cost:

- $c(x, R) = C$
- $c(x, K) = c(x)$ maintenance plus extra costs.

Example: the Optimal Replacement Problem

State: level of wear of an object (e.g., a car).
Action: $\{(R)\text{eplace}, (K)\text{eep}\}$.
Cost:

- $c(x, R) = C$
- $c(x, K) = c(x)$ maintenance plus extra costs.

Dynamics:

- $p(\cdot|x, R) = \exp(\beta)$ with density $d(y) = \beta \exp^{-\beta y} \mathbb{I}\{y \geq 0\}$,
- $p(\cdot|x, K) = x + \exp(\beta)$ with density $d(y - x)$.

Example: the Optimal Replacement Problem

State: level of wear of an object (e.g., a car).
Action: $\{(R)\text{eplace}, (K)\text{eep}\}$.
Cost:

- $c(x, R) = C$
- $c(x, K) = c(x)$ maintenance plus extra costs.

Dynamics:

- $p(\cdot|x, R) = \exp(\beta)$ with density $d(y) = \beta \exp^{-\beta y} \mathbb{I}\{y \geq 0\}$,
- $p(\cdot|x, K) = x + \exp(\beta)$ with density $d(y - x)$.

Problem: Minimize the discounted expected cost over an infinite horizon.

Example: the Optimal Replacement Problem

Optimal value function

$$V^*(x) = \min\left\{c(x) + \gamma \int_0^\infty d(y-x)V^*(y)dy,\ C + \gamma \int_0^\infty d(y)V^*(y)dy\right\}$$

Example: the Optimal Replacement Problem

Optimal value function

$$V^*(x) = \min\left\{c(x)+\gamma\int_0^\infty d(y-x)V^*(y)dy,\ C+\gamma\int_0^\infty d(y)V^*(y)dy\right\}$$

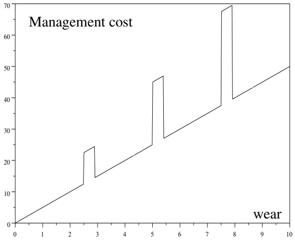
Optimal policy: action that attains the minimum

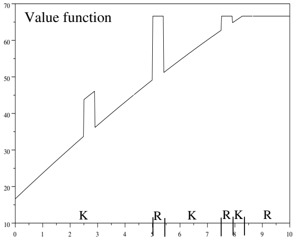
Example: the Optimal Replacement Problem

Optimal value function

$$V^*(x) = \min\left\{c(x) + \gamma \int_0^\infty d(y-x)V^*(y)dy,\ C + \gamma \int_0^\infty d(y)V^*(y)dy\right\}$$

Optimal policy: action that attains the minimum



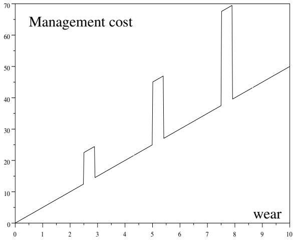


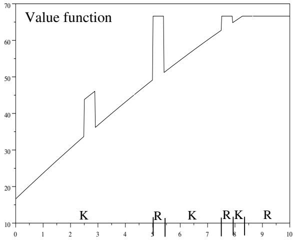
Example: the Optimal Replacement Problem

Optimal value function

$$V^*(x) = \min\left\{c(x)+\gamma\int_0^\infty d(y-x)V^*(y)dy,\ C+\gamma\int_0^\infty d(y)V^*(y)dy\right\}$$

Optimal policy: action that attains the minimum

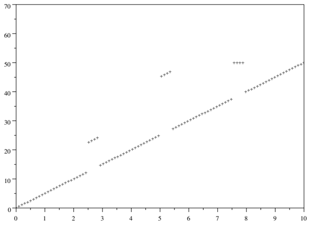




Linear approximation space $\mathcal{F} := \left\{V_n(x) = \sum_{k=1}^{20}\alpha_k \cos(k\pi\frac{x}{x_{\max}})\right\}$.

Example: the Optimal Replacement Problem

Collect N sample on a uniform grid.



Example: the Optimal Replacement Problem

Collect N sample on a uniform grid.

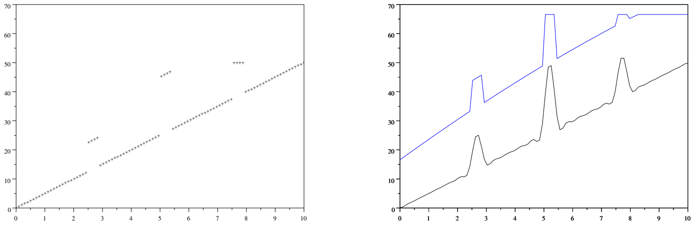
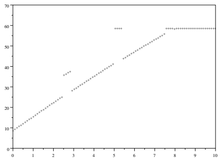
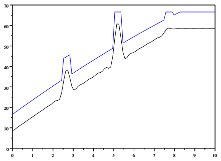


Figure: Left: the *target* values computed as $\{\mathcal{T}V_0(x_n)\}_{1\leq n\leq N}$. Right: the approximation $V_1 \in \mathcal{F}$ of the target function $\mathcal{T}V_0$.

Example: the Optimal Replacement Problem





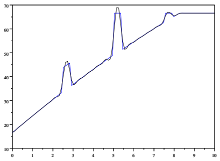


Figure: Left: the *target* values computed as $\{\mathcal{T}V_1(x_n)\}_{1\leq n\leq N}$. Center: the approximation $V_2 \in \mathcal{F}$ of $\mathcal{T}V_1$. Right: the approximation $V_n \in \mathcal{F}$ after n iterations.

Example: the Optimal Replacement Problem

Simulation

Approximate Dynamic Programming

(a.k.a. Batch Reinforcement Learning)

Approximate Value Iteration

Approximate Policy Iteration

Policy Iteration: the Idea

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

$$\pi_{k+1}(x) \in \arg\max_{a \in A} \Big[r(x,a) + \gamma \sum_y p(y|x,a) V^{\pi_k}(y) \Big].$$

3. Return the last policy π_K

Policy Iteration: the Idea

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

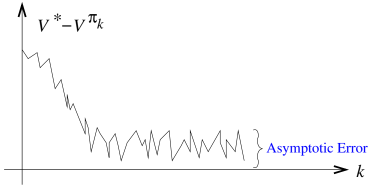
$$\pi_{k+1}(x) \in \arg\max_{a \in A} \Big[r(x,a) + \gamma \sum_y p(y|x,a) V^{\pi_k}(y) \Big].$$

3. Return the last policy π_K

- ***Problem***: how can we approximate V^{π_k}?
- ***Problem***: if $V_k \neq V^{\pi_k}$, does (approx.) policy iteration still work?

Approximate Policy Iteration: performance loss

Problem: the algorithm is no longer guaranteed to converge.



Proposition

The asymptotic performance of the policies π_k generated by the API algorithm is related to the approximation error as:

$$\limsup_{k\to\infty} \underbrace{\|V^* - V^{\pi_k}\|_\infty}_{\textit{performance loss}} \leq \frac{2\gamma}{(1-\gamma)^2} \limsup_{k\to\infty} \underbrace{\|V_k - V^{\pi_k}\|_\infty}_{\textit{approximation error}}$$

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*.

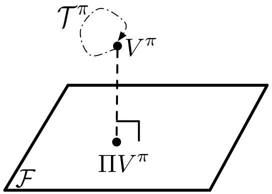
*In practice we use approximations of action-value functions.

Least-Squares Temporal-Difference Learning (LSTD)

- 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)

- V^π is the fixed-point of $\mathcal{T}^\pi$

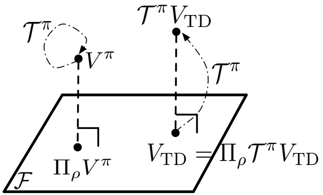
$$V^\pi = \mathcal{T}^\pi V^\pi = r^\pi + \gamma P^\pi V^\pi$$

- LSTD searches for the fixed-point of $\Pi_{2,\rho}\mathcal{T}^\pi$

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

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

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


$\mathcal{T}^\pi$
V^π
$\mathcal{T}^\pi V_{\text{TD}}$
$\mathcal{T}^\pi$
$\Pi_\rho V^\pi$
$V_{\text{TD}} = \Pi_\rho \mathcal{T}^\pi V_{\text{TD}}$
$\mathcal{F}$

Least-Squares Temporal-Difference Learning (LSTD)

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

- The projection Π_ρ is orthogonal *in expectation* w.r.t. the space $\mathcal{F}$ *spanned* by the features $\varphi_1, \dots, \varphi_d$

$$\mathbb{E}_{x\sim\rho}\big[(\mathcal{T}^\pi V_{TD}(x) - V_{TD}(x))\varphi_i(x)\big] = 0, \;\; \forall i \in [1, d]$$

$$\langle \mathcal{T}^\pi V_{TD} - V_{TD}, \varphi_i \rangle_\rho = 0$$

Least-Squares Temporal-Difference Learning (LSTD)

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

- The projection Π_ρ is orthogonal *in expectation* w.r.t. the space $\mathcal{F}$ *spanned* by the features $\varphi_1, \ldots, \varphi_d$

$$\mathbb{E}_{x\sim\rho}\big[(\mathcal{T}^\pi V_{TD}(x) - V_{TD}(x))\varphi_i(x)\big] = 0, \;\; \forall i \in [1, d]$$

$$\langle \mathcal{T}^\pi V_{TD} - V_{TD}, \varphi_i \rangle_\rho = 0$$

- By definition of Bellman operator

$$\langle r^\pi + \gamma P^\pi V_{TD} - V_{TD}, \varphi_i \rangle_\rho = 0$$

$$\langle r^\pi, \varphi_i \rangle_\rho - \langle (I - \gamma P^\pi) V_{TD}, \varphi_i \rangle_\rho = 0$$

Least-Squares Temporal-Difference Learning (LSTD)

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

- The projection Π_ρ is orthogonal *in expectation* w.r.t. the space $\mathcal{F}$ *spanned* by the features $\varphi_1, \dots, \varphi_d$

$$\mathbb{E}_{x\sim\rho}\big[(\mathcal{T}^\pi V_{TD}(x) - V_{TD}(x))\varphi_i(x)\big] = 0, \ \ \forall i \in [1, d]$$

$$\langle \mathcal{T}^\pi V_{TD} - V_{TD}, \varphi_i \rangle_\rho = 0$$

- By definition of Bellman operator

$$\langle r^\pi + \gamma P^\pi V_{TD} - V_{TD}, \varphi_i \rangle_\rho = 0$$

$$\langle r^\pi, \varphi_i \rangle_\rho - \langle (I - \gamma P^\pi) V_{TD}, \varphi_i \rangle_\rho = 0$$

- Since $V_{TD} \in \mathcal{F}$, there exists α_{TD} such that $V_{TD}(x) = \phi(x)^\top \alpha_{TD}$

$$\langle r^\pi, \varphi_i \rangle_\rho - \sum_{j=1}^d \langle (I - \gamma P^\pi)\varphi_j \alpha_{TD,j}, \varphi_i \rangle_\rho = 0$$

$$\langle r^\pi, \varphi_i \rangle_\rho - \sum_{j=1}^d \langle (I - \gamma P^\pi)\varphi_j, \varphi_i \rangle_\rho \alpha_{TD,j} = 0$$

Least-Squares Temporal-Difference Learning (LSTD)

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

$$\Downarrow$$

$$\underbrace{\langle r^\pi, \varphi_i \rangle_\rho}_{b_i} - \sum_{j=1}^{d} \underbrace{\langle (I - \gamma P^\pi)\varphi_j, \varphi_i \rangle_\rho}_{A_{i,j}} \alpha_{TD,j} = 0$$

$$\Downarrow$$

$$A\alpha_{TD} = b$$

Least-Squares Temporal-Difference Learning (LSTD)

- **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)

- **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.

- **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 $\pi_0$
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 $\pi_0$
For $k = 1, \dots, K$
    1. Generate a trajectory of length $n$ from the stationary dist. $\rho^{\pi_k}$
           $(x_1, \pi_k(x_1), r_1, x_2, \pi_k(x_2), r_2, \dots, x_{n-1}, \pi_k(x_{n-1}), r_{n-1}, x_n)$
```

Least-Squares Policy Iteration (LSPI)

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

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

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

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

2. Compute the empirical matrix $\widehat{A}_k$ and the vector $\widehat{b}_k$

$$[\widehat{A}_k]_{i,j} = \frac{1}{n}\sum_{t=1}^{n}(\varphi_j(x_t) - \gamma\varphi_j(x_{t+1})\varphi_i(x_t) \approx \langle (I - \gamma P^{\pi})\varphi_j, \varphi_i \rangle_{\rho^{\pi_k}}$$

$$[\widehat{b}_k]_i = \frac{1}{n}\sum_{t=1}^{n}\varphi_i(x_t) r_t \approx \langle r^{\pi}, \varphi_i \rangle_{\rho^{\pi_k}}$$

3. 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 $k = 1, \dots, K$

1. Generate a trajectory of length n from the stationary dist. ρ^{π_k}
$$(x_1, \pi_k(x_1), r_1, x_2, \pi_k(x_2), r_2, \dots, x_{n-1}, \pi_k(x_{n-1}), r_{n-1}, x_n)$$
2. Compute the empirical matrix $\widehat{A}_k$ and the vector $\widehat{b}_k$
$$[\widehat{A}_k]_{i,j} = \frac{1}{n}\sum_{t=1}^{n}(\varphi_j(x_t) - \gamma\varphi_j(x_{t+1})\varphi_i(x_t) \approx \langle (I - \gamma P^{\pi})\varphi_j, \varphi_i \rangle_{\rho^{\pi_k}}$$
$$[\widehat{b}_k]_i = \frac{1}{n}\sum_{t=1}^{n}\varphi_i(x_t) r_t \approx \langle r^{\pi}, \varphi_i \rangle_{\rho^{\pi_k}}$$
3. Solve the linear system $\alpha_k = \widehat{A}_k^{-1}\widehat{b}_k$
4. 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 $\pi_0$
For $k = 1, \dots, K$
    1. Generate a trajectory of length $n$ from the stationary dist. $\rho^{\pi_k}$
       $(x_1, \pi_k(x_1), r_1, x_2, \pi_k(x_2), r_2, \dots, x_{n-1}, \pi_k(x_{n-1}), r_{n-1}, x_n)$
    2. Compute the empirical matrix $\widehat{A}_k$ and the vector $\widehat{b}_k$
       $[\widehat{A}_k]_{i,j} = \frac{1}{n} \sum_{t=1}^{n} (\varphi_j(x_t) - \gamma \varphi_j(x_{t+1}) \varphi_i(x_t) \approx \langle (I - \gamma P^{\pi}) \varphi_j, \varphi_i \rangle_{\rho^{\pi_k}}$
       $[\widehat{b}_k]_i = \frac{1}{n} \sum_{t=1}^{n} \varphi_i(x_t) r_t \approx \langle r^{\pi}, \varphi_i \rangle_{\rho^{\pi_k}}$
    3. Solve the linear system $\alpha_k = \widehat{A}_k^{-1} \widehat{b}_k$
    4. Compute the greedy policy $\pi_{k+1}$ w.r.t. $\widehat{V}_k = f_{\alpha_k}$
Return the last policy $\pi_K$
```

Least-Squares Policy Iteration (LSPI)

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

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

- The first few samples may be *discarded* because not actually drawn from the *stationary* distribution ρ^{π_k}
- *Off-policy* samples could be used with *importance weighting*
- In practice i.i.d. states drawn from an arbitrary distribution (but with actions π_k) may be used

Least-Squares Policy Iteration (LSPI)

4. Compute the greedy policy π_{k+1} w.r.t. $\widehat{V}_k = f_{\alpha_k}$

- Computing the greedy policy from $\widehat{V}_k$ is difficult, so move to LSTD-Q and compute

$$\pi_{k+1}(x) = \arg\max_a \widehat{Q}_k(x, a)$$

Least-Squares Policy Iteration (LSPI)

For $k = 1, \ldots, K$

Least-Squares Policy Iteration (LSPI)

For $k = 1, \ldots, K$

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

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

...

4. Compute the greedy policy π_{k+1} w.r.t. $\widehat{V}_k = f_{\alpha_k}$

Problem: This process may be unstable because π_k ***does not cover*** the state space *properly*

▸ Skip Theory

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

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

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,

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

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...
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

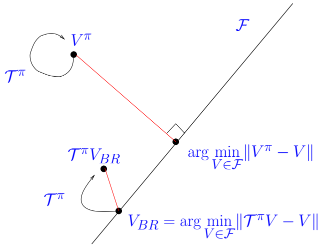
$$||V^* - V^{\pi_K}||_\mu \leq \frac{4\gamma}{(1-\gamma)^2}\left\{\sqrt{CC_{\mu,\rho}}\left[cE_0(\mathcal{F}) + O\left(\sqrt{\frac{d\log(dK/\delta)}{n\,\nu_\rho}}\right)\right] + \gamma^K R_{\max}\right\}$$

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 $(\int \varphi_i\,\varphi_j\,d\rho)_{i,j}$

Bellman Residual Minimization (BRM): the idea


$\mathcal{F}$
V^π
$\mathcal{T}^\pi$
$\arg\min_{V\in\mathcal{F}} \|V^\pi - V\|$
$\mathcal{T}^\pi V_{BR}$
$\mathcal{T}^\pi$
$V_{BR} = \arg\min_{V\in\mathcal{F}} \|\mathcal{T}^\pi V - V\|$

Let μ be a distribution over X, V_{BR} is the minimum *Bellman residual w.r.t.* $\mathcal{T}^\pi$

$$V_{BR} = \arg\min_{V\in\mathcal{F}} \| T^\pi V - V\|_{2,\mu}$$

Bellman Residual Minimization (BRM): the idea

The mapping $\alpha \to \mathcal{T}^\pi V_\alpha - V_\alpha$ is affine
The function $\alpha \to \|\mathcal{T}^\pi V_\alpha - V_\alpha\|_\mu^2$ is quadratic
$\Rightarrow$ The minimum is obtained by computing the *gradient and setting it to zero*

$$\langle r^\pi + (\gamma P^\pi - I)\sum_{j=1}^{d} \phi_j \alpha_j, (\gamma P^\pi - I)\phi_i \rangle_\mu = 0,$$

which can be rewritten as $A\alpha = b$, with

$$\begin{cases} A_{i,j} &= \langle \phi_i - \gamma P^\pi \phi_i, \phi_j - \gamma P^\pi \phi_j \rangle_\mu, \\ b_i &= \langle \phi_i - \gamma P^\pi \phi_i, r^\pi \rangle_\mu, \end{cases}$$

Bellman Residual Minimization (BRM): the idea

Remark: the system admits a solution whenever the features ϕ_i are *linearly independent* w.r.t. μ

Bellman Residual Minimization (BRM): the idea

Remark: the system admits a solution whenever the features ϕ_i are *linearly independent* w.r.t. μ

Remark: let $\{\psi_i = \phi_i - \gamma P^\pi \phi_i\}_{i=1\dots d}$, then the previous system can be interpreted as a linear regression problem

$$\|\alpha \cdot \psi - r^\pi\|_\mu$$

BRM: the approximation error

Proposition

We have

$$\|V^\pi - V_{BR}\| \leq \|(I - \gamma P^\pi)^{-1}\|(1 + \gamma\|P^\pi\|) \inf_{V \in \mathcal{F}} \|V^\pi - V\|.$$

If μ_π is the *stationary policy* of π, then $\|P^\pi\|_{\mu_\pi} = 1$ and $\|(I - \gamma P^\pi)^{-1}\|_{\mu_\pi} = \frac{1}{1-\gamma}$, thus

$$\|V^\pi - V_{BR}\|_{\mu_\pi} \leq \frac{1+\gamma}{1-\gamma} \inf_{V \in \mathcal{F}} \|V^\pi - V\|_{\mu_\pi}.$$

BRM: the implementation

Assumption. A generative model is available.

- Drawn n states $X_t \sim \mu$
- Call generative model on (X_t, A_t) (with $A_t = \pi(X_t)$) and obtain $R_t = r(X_t, A_t)$, $Y_t \sim p(\cdot|X_t, A_t)$
- Compute

$$\hat{\mathcal{B}}(V) = \frac{1}{n}\sum_{t=1}^{n}\big[V(X_t) - \underbrace{\big(R_t + \gamma V(Y_t)\big)}_{\hat{\mathcal{T}}V(X_t)}\big]^2.$$

BRM: the implementation

Problem: this estimator is *biased and not consistent*! In fact,

$$\begin{aligned}\mathbb{E}[\hat{\mathcal{B}}(V)] &= \mathbb{E}\Big[\big[V(X_t) - \mathcal{T}^\pi V(X_t) + \mathcal{T}^\pi V(X_t) - \hat{\mathcal{T}} V(X_t)\big]^2\Big] \\ &= \|\mathcal{T}^\pi V - V\|_\mu^2 + \mathbb{E}\Big[\big[\mathcal{T}^\pi V(X_t) - \hat{\mathcal{T}} V(X_t)\big]^2\Big]\end{aligned}$$

$\Rightarrow$ minimizing $\hat{\mathcal{B}}(V)$ *does not* correspond to minimizing $\mathcal{B}(V)$ (even when $n \to \infty$).

BRM: the implementation

Solution. In each state X_t, generate *two independent samples* Y_t et $Y'_t \sim p(\cdot|X_t, A_t)$
Define

$$\hat{\mathcal{B}}(V) = \frac{1}{n}\sum_{t=1}^{n}\big[V(X_t) - \big(R_t + \gamma V(Y_t)\big)\big]\big[V(X_t) - \big(R_t + \gamma V(Y'_t)\big)\big].$$

$\Rightarrow \hat{\mathcal{B}} \to \mathcal{B}$ for $n \to \infty$.

BRM: the implementation

The function $\alpha \to \hat{\mathcal{B}}(V_\alpha)$ is quadratic and we obtain the linear system

$$\begin{aligned} \widehat{A}_{i,j} &= \frac{1}{n}\sum_{t=1}^{n}\big[\phi_i(X_t) - \gamma\phi_i(Y_t)\big]\big[\phi_j(X_t) - \gamma\phi_j(Y'_t)\big], \\ \widehat{b}_i &= \frac{1}{n}\sum_{t=1}^{n}\big[\phi_i(X_t) - \gamma\frac{\phi_i(Y_t) + \phi_i(Y'_t)}{2}\big] R_t. \end{aligned}$$

BRM: the approximation error

Proof. We relate the Bellman residual to the approximation error as

$$\begin{aligned} V^\pi - V &= V^\pi - T^\pi V + T^\pi V - V = \gamma P^\pi (V^\pi - V) + T^\pi V - \\ (I - \gamma P^\pi)(V^\pi - V) &= T^\pi V - V, \end{aligned}$$

taking the norm both sides we obtain

$$\|V^\pi - V_{BR}\| \leq \|(I - \gamma P^\pi)^{-1}\| \|\mathcal{T}^\pi V_{BR} - V_{BR}\|$$

and

$$\|\mathcal{T}^\pi V_{BR} - V_{BR}\| = \inf_{V \in \mathcal{F}} \|\mathcal{T}^\pi V - V\| \leq (1 + \gamma \|P^\pi\|) \inf_{V \in \mathcal{F}} \|V^\pi - V\|.$$

BRM: the approximation error

Proof. If we consider the stationary distribution μ_π, then $\|P^\pi\|_{\mu_\pi} = 1$. The matrix $(I - \gamma P^\pi)$ can be written as the power series $\sum_t \gamma (P^\pi)^t$. Applying the norm we obtain

$$\|(I - \gamma P^\pi)^{-1}\|_{\mu_\pi} \leq \sum_{t \geq 0} \gamma^t \|P^\pi\|_{\mu_\pi}^t \leq \frac{1}{1-\gamma}$$

■

LSTD vs BRM

- **Different assumptions:** BRM requires a *generative model*, LSTD requires a *single trajectory*.
- **The performance is evaluated differently:** BRM *any* distribution, LSTD *stationary* distribution μ^π.

How to solve *approximately* an MDP

Approximate Dynamic Programming

(a.k.a. Batch Reinforcement Learning)

Approximate Value Iteration

Neural Q-learning (aka DQN)

Q-learning with Function Approximation

Exact Q-learning

- Compute the temporal difference on $\langle x_t, a_t, r_t, x_{t+1} \rangle$

$$\delta_t = r_t + \gamma \max_{a'} Q(x_{t+1}, a') - Q(x_t, a_t)$$

- Update the estimate of Q as

$$Q(x_t, a_t) = Q(x_t, a_t) + \alpha(x_t, a_t)\delta_t$$

Q-learning with Function Approximation

Approximate Q-learning

- Parameterize the Q-function $Q(x, a; \theta)$ using a NN architecture
- Define the error

$$L(\theta) = \mathbb{E}\big[r(x,a) + \gamma \max Q(y, a'; \theta') - Q(x, a; \theta)^2\big]$$

- Compute the gradient

$$\nabla_\theta L(\theta) = \mathbb{E}\big[(r(x,a) + \gamma \max Q(y, a'; \theta') - Q(x, a; \theta))\nabla_\theta Q(x, a; \theta)\big]$$

- Update the parameter

$$\theta_{t+1} = \theta_t + \alpha \nabla_\theta L(\theta_t)$$

Q-learning with Function Approximation

Approximate Q-learning

- Parameterize the Q-function $Q(x, a; \theta)$ using a NN architecture
- Define the error

$$L(\theta) = \mathbb{E}\Big[r(x,a) + \gamma \max Q(y, a'; \theta') - Q(x, a; \theta)^2\Big]$$

- Compute the gradient

$$\nabla_\theta L(\theta) = \mathbb{E}\Big[(r(x,a) + \gamma \max Q(y, a'; \theta') - Q(x, a; \theta))\nabla_\theta Q(x, a; \theta)\Big]$$

- Update the parameter

$$\theta_{t+1} = \theta_t + \alpha \nabla_\theta L(\theta_t)$$

Main issues

- $\nabla_\theta L(\theta)$ cannot be computed (no expectation)
- Strong correlations between approximation, policy, and data
- Since data are then fed back into the approximation, this may lead to instability and divergence

Q-learning with Function Approximation

```
For $i = 1, \dots, n$
    1. Set $t = 0$
    2. Set initial state $x_0$
    3. While ($x_t$ not terminal)
        3.1 Take action $a_t$ with $\varepsilon$-greedy strategy using $Q(x_t, a; \theta_i)$
        3.2 Observe next state $x_{t+1}$ and reward $r_t$
        3.3 Store transition $x_t, a_t, x_{t+1}, r_t$ in $\mathcal{D}$
        3.4 Sample a random transition $x, a, x', r$ from $\mathcal{D}$ [action reply]
        3.5 Compute target [batch updates]
                $y = r + \gamma \max_b Q(x', b; \theta_i)$
        3.6 Perform gradient descent on $\big(y - Q(x, a; \theta_i)\big)^2$ and update $\theta_{i+1}$
    EndWhile
EndFor
```

Q-learning with Function Approximation

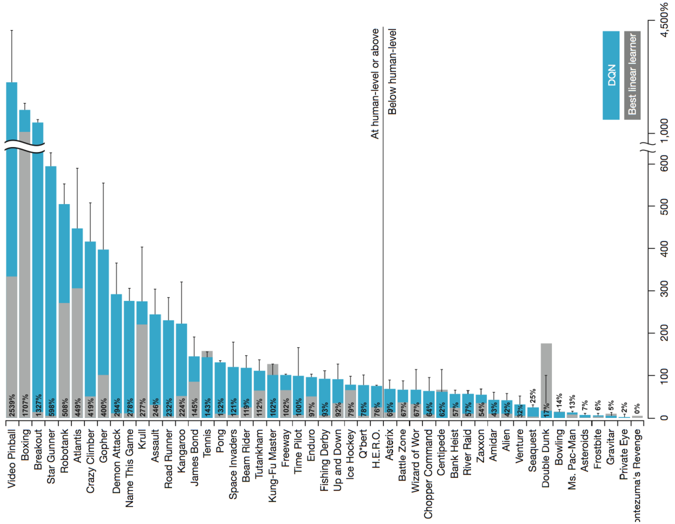
Why it works:

- *Action reply*: de-correlate changes to θ to the current policy
- *One-sample update*: similar to stochastic gradient descent
- *Batch updates*: "freeze" the policy for a while

$\Rightarrow$ increase the *stability* by reducing the (fast) loops on changing approximation, policy and data

Q-learning with Function Approximation

Super-human performance



Q-learning with Function Approximation

Why it works in Atari games:

- Based on images: ConNets work well on images
- Almost deterministic environment
- Massive amount of data

Q-learning with Function Approximation

Why it works in Atari games:

- Based on images: ConNets work well on images
- Almost deterministic environment
- Massive amount of data

$\Rightarrow$ would it still work in, eg, financial applications?

Bibliography I

Reinforcement Learning

Alessandro Lazaric
alessandro.lazaric@inria.fr
sequel.lille.inria.fr