Approximate Dynamic Programming

A. LAZARIC (SequeL Team @INRIA-Lille)
ENS Cachan - Master 2 MVA
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 $r(x, a)$

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 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 $\pi$ such that

the *performance loss* $\|V^* - V^\pi\|$ is as small as possible
**Question**: if $V$ is an approximation of the optimal value function $V^*$ with an error

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

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

$$\pi(x) \in \arg \max \sum_a \sum_y p(y|x,a) \left[ r(x,a,y) + \gamma V(y) \right]$$

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 $\pi$ its corresponding greedy policy, then

\[
\|V^* - V^\pi\|_\infty \leq \frac{2\gamma}{1 - \gamma} \|V^* - V\|_\infty.
\]

**performance loss** \hspace{0.5cm} \text{approx. error}

Furthermore, there exists $\epsilon > 0$ such that if $\|V - V^*\|_\infty \leq \epsilon$, then $\pi$ is **optimal**.
From Approximation Error to Performance Loss

Proof.

\[
\| V^* - V^\pi \|_\infty \leq \| TV^* - T^\pi V \|_\infty + \| T^\pi V - T^\pi V^\pi \|_\infty \\
\leq \| TV^* - TV \|_\infty + \gamma \| V - V^\pi \|_\infty \\
\leq \gamma \| V^* - V \|_\infty + \gamma(\| V - V^* \|_\infty + \| V^* - V^\pi \|_\infty) \\
\leq \frac{2\gamma}{1 - \gamma} \| V^* - V \|_\infty.
\]
Approximate Dynamic Programming
(a.k.a. Batch Reinforcement Learning)

Approximate Value Iteration

Approximate Policy Iteration
Question: how do we compute a good $V$?
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
     \[
     Q_{k+1}(x, a) = T Q_k(x, a) = r(x, a) + \sum_y p(y|x, a) \gamma \max_b Q_k(y, b)
     \]

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
     
     $$Q_{k+1}(x, a) = T Q_k(x, a) = r(x, a) + \sum_y p(y|x, a) \gamma \max_b Q_k(y, b)$$

3. Return the greedy policy
   
   $$\pi_K(x) \in \arg \max_{a \in A} Q_K(x, a).$$

- **Problem**: how can we approximate $T Q_k$?
- **Problem**: if $Q_{k+1} \neq T Q_k$, 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)

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

with features

\[ \varphi_j : X \times A \rightarrow [0, L], \quad \phi(x, a) = [\varphi_1(x, a) \ldots \varphi_d(x, a)]^\top \]
Linear Fitted Q-iteration: the Samples

**Assumption**: access to a *generative model*, that is a black-box simulator $\text{sim}()$ of the environment is available.

Given $(x, a)$,

$$\text{sim}(x, a) = \{y, r\}, \quad \text{with } y \sim p(\cdot | x, a), \quad r = r(x, a)$$
Linear Fitted Q-iteration

**Input**: space $\mathcal{F}$, iterations $K$, sampling distribution $\rho$, num of samples $n$
Linear Fitted Q-iteration

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

Initial function $\tilde{Q}_0 \in \mathcal{F}$
Linear Fitted Q-iteration

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

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

For $k = 1, \ldots, K$

1. Draw $n$ samples $(x_i, a_i)$ 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 $y_i = r_i + \gamma \max_a \hat{Q}_{k-1}(x_i', a)$
4. Build training set $\{(x_i, a_i, y_i)\}_{i=1}^n$
5. Solve the least squares problem $\hat{\alpha}_k = \arg \min_{\alpha \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^n (f_{\alpha}(x_i, a_i) - y_i)^2$
6. Return $\hat{Q}_k = f_{\hat{\alpha}_k}$ (truncation may be needed)

Return $\pi_K(\cdot) = \arg \max_a \hat{Q}_K(\cdot, a)$ (greedy policy)
Linear Fitted Q-iteration

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

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

1. Draw $n$ samples $(x_i, a_i) \sim \rho$
Linear Fitted Q-iteration

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

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

For $k = 1, \ldots, K$

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

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 $\rho$, num of samples $n$

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

For $k = 1, \ldots, K$

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

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

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

For $k = 1, \ldots, K$

1. Draw $n$ samples $(x_i, a_i)^{i.i.d.} \sim \rho$

2. Sample $x'_i \sim p(\cdot| x_i, a_i)$ and $r_i = r(x_i, a_i)$

3. Compute $y_i = r_i + \gamma \max_a \tilde{Q}_{k-1}(x'_i, a)$

4. Build training set $\{(x_i, a_i), y_i\}_{i=1}^n$

5. Solve the least squares problem $\hat{\alpha}_k = \arg\min_{\alpha \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^n (f_{\alpha}(x_i, a_i) - y_i)^2$

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

Return $\pi_K(\cdot) = \arg\max_a \hat{Q}_K(\cdot, a)$ (greedy policy)
Linear Fitted Q-iteration

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

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

For $k = 1, \ldots, K$

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

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

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

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

For $k = 1, \ldots, K$

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

2. Sample $x'_i \sim p(\cdot|x_i, a_i)$ and $r_i = r(x_i, a_i)$

3. Compute $y_i = r_i + \gamma \max_a \hat{Q}_{k-1}(x'_i, a)$

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 $\hat{Q}_k = f_{\hat{\alpha}_k}$ (truncation may be needed)
Linear Fitted Q-iteration

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

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

For $k = 1, \ldots, K$

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

2. Sample $x_i' \sim p(\cdot|x_i, a_i)$ and $r_i = r(x_i, a_i)$

3. Compute $y_i = r_i + \gamma \max_a \hat{Q}_{k-1}(x_i', a)$

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

**Return** $\pi_K(\cdot) = \arg\max_a \hat{Q}_K(\cdot, a)$ (*greedy policy*)
Linear Fitted Q-iteration: Sampling

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

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 $(x_i, a_i) \overset{i.i.d}{\sim} \rho$

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 $\rho$ should cover the state-action space in all relevant regions
- If not possible to choose $\rho$, a database of samples can be used
Linear Fitted Q-iteration: The Training Set

4. Compute \( y_i = r_i + \gamma \max_a \tilde{Q}_{k-1}(x'_i, a) \)
5. Build training set \( \{(x_i, a_i), y_i\}_{i=1}^n \)
Linear Fitted Q-iteration: The Training Set

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

- Each sample $y_i$ is an unbiased sample, since

$$
E[y_i|x_i, a_i] = E[r_i + \gamma \max_a \hat{Q}_{k-1}(x'_i, a)] = r(x_i, a_i) + \gamma E[\max_a \hat{Q}_{k-1}(x'_i, a)]
$$

$$
= r(x_i, a_i) + \gamma \int_X \max_a \hat{Q}_{k-1}(x', a) p(dy|x, a) = T\hat{Q}_{k-1}(x_i, a_i)
$$

- 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

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

7. Return \( \hat{Q}_k = f_{\hat{\alpha}_k} \) (truncation may be needed)
Linear Fitted Q-iteration: The Regression Problem

6. Solve the least squares problem

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

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

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

\[ Q_0 \xrightarrow{\tau} Q_1 \xrightarrow{\epsilon_1} \hat{Q}_1 \xrightarrow{\tau} \hat{Q}_2 \xrightarrow{\epsilon_2} \hat{Q}_2 \xrightarrow{\tau} \hat{Q}_3 \xrightarrow{\epsilon_3} \hat{Q}_3 \xrightarrow{\tau} \hat{Q}_K \xrightarrow{\text{greedy } \pi_K} Q^\pi_K \]

\[
\begin{align*}
Q^* &\xleftarrow{\text{final error}} Q^\pi_K \\
\end{align*}
\]
Theoretical Objectives

Objective: derive a bound on the performance (quadratic) loss w.r.t. a testing distribution $\mu$

$$||Q^* - Q^{\pi K}||_\mu \leq ???$$
Theoretical Objectives

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

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

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

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

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

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

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

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

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

$$\|Q^* - Q^{\pi_K}\|_\mu \leq \text{propagation}(\|\mathcal{T}\hat{Q}_{k-1} - \hat{Q}_k\|_\rho)$$
The Sources of Error

- Desired solution

\[ Q_k = T \hat{Q}_{k-1} \]
The Sources of Error

- **Desired solution**

\[ Q_k = T \hat{Q}_{k-1} \]

- **Best solution (wrt sampling distribution \( \rho \))**

\[ f_{\alpha_k}^* = \arg \inf_{f_{\alpha} \in \mathcal{F}} \| f_{\alpha} - Q_k \|_{\rho} \]
The Sources of Error

- **Desired** solution

\[ Q_k = \mathcal{T} \hat{Q}_{k-1} \]

- **Best** solution (wrt sampling distribution \( \rho \))

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

\[ \Rightarrow Error \text{ from the approximation space } \mathcal{F} \]
The Sources of Error

- **Desired** solution

\[ Q_k = \mathcal{T} \hat{Q}_{k-1} \]

- **Best** solution (wrt sampling distribution \( \rho \))

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

\[ \Rightarrow \text{Error} \text{ from the approximation space } \mathcal{F} \]

- **Returned** solution

\[ f_{\hat{\alpha}} = \arg \min_{f_{\alpha} \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} (f_{\alpha}(x_i, a_i) - y_i)^2 \]
The Sources of Error

- **Desired** solution

\[ Q_k = T \hat{Q}_{k-1} \]

- **Best** solution (wrt sampling distribution \( \rho \))

\[ f_{\alpha^*} = \arg \inf_{f_\alpha \in F} \left\| f_\alpha - Q_k \right\|_\rho \]

\[ \Rightarrow \text{Error from the approximation space } F \]

- **Returned** solution

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

\[ \Rightarrow \text{Error from the (random) samples} \]
Per-Iteration Error

**Theorem**

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

$$||Q_k - \hat{Q}_k||_\rho \leq 4||Q_k - f_{\alpha^*_k}||_\rho$$

$$+ O\left( (V_{\text{max}} + L||\alpha^*_k||) \sqrt{\frac{\log 1/\delta}{n}} \right)$$

$$+ O\left( V_{\text{max}} \sqrt{\frac{d \log n/\delta}{n}} \right),$$

with probability $1 - \delta$.

Tools: concentration of measure inequalities, covering space, linear algebra, union bounds, special tricks for linear spaces, ...
\[ \left\| Q_k - \hat{Q}_k \right\|_\rho \leq 4\left\| Q_k - f_{\alpha_k^*} \right\|_\rho \]

\[ + O\left( \left( V_{\text{max}} + L\|\alpha_k^*\| \right) \sqrt{\frac{\log 1/\delta}{n}} \right) \]

\[ + O\left( V_{\text{max}} \sqrt{\frac{d \log n/\delta}{n}} \right) \]
Per-Iteration Error

\[ \| Q_k - \hat{Q}_k \|_\rho \leq 4 \| Q_k - f_{\alpha_k^*} \|_\rho \]

\[ + O \left( (V_{\text{max}} + L \| \alpha_k^* \|) \sqrt{\frac{\log 1/\delta}{n}} \right) \]

\[ + O \left( V_{\text{max}} \sqrt{\frac{d \log n/\delta}{n}} \right) \]

Remarks

- No algorithm can do better
- Constant 4
- Depends on the space \( \mathcal{F} \)
- Changes with the iteration \( k \)
Per-Iteration Error

\[ \| Q_k - \hat{Q}_k \|_\rho \leq 4 \| Q_k - f_{\alpha^*_k} \|_\rho \]

\[ + O \left( (V_{\text{max}} + L \| \alpha^*_k \|) \sqrt{\frac{\log 1/\delta}{n}} \right) \]

\[ + O \left( V_{\text{max}} \sqrt{\frac{d \log n/\delta}{n}} \right) \]

Remarks

- Vanishing to zero as \( O(n^{-1/2}) \)
- Depends on the features (\( L \)) and on the best solution (\( \| \alpha^*_k \| \))
Per-Iteration Error

\[ \| Q_k - \hat{Q}_k \|_\rho \leq 4 \| Q_k - f_{\alpha^*_k} \|_\rho \]

\[ + O\left( (V_{\text{max}} + L \| \alpha^*_k \| ) \sqrt{\frac{\log 1/\delta}{n}} \right) \]

\[ + O\left( V_{\text{max}} \sqrt{\frac{d \log n/\delta}{n}} \right) \]

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 \( \mu \) is different from the sampling norm \( \rho \)
Error Propagation

Objective

\[ \| Q^* - Q^{\pi_K} \|_\mu \]

- **Problem 1:** the test norm \( \mu \) is different from the sampling norm \( \rho \)
- **Problem 2:** we have bounds for \( \hat{Q}_k \) not for the performance of the corresponding \( \pi_k \)
Error Propagation

Objective

\[ \| Q^* - Q^{\pi_k} \|_\mu \]

- **Problem 1**: the test norm \( \mu \) is different from the sampling norm \( \rho \)
- **Problem 2**: we have bounds for \( \hat{Q}_k \) not for the performance of the corresponding \( \pi_k \)
- **Problem 3**: we have bounds for one single iteration
Error Propagation

Transition kernel for a fixed policy $P^\pi$.

- $m$-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 $P^\pi$.

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

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

- 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

\[ L^+ = \sup_{\pi} \lim_{m \to \infty} \sup_{\rho} \frac{1}{m} \log^+ (||\rho P_{\pi_1} P_{\pi_2} \ldots P_{\pi_m}||) \]

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 \( \epsilon_k = Q_k - \hat{Q}_k \) be the propagation error at each iteration, then after \( K \) iteration the *performance loss* of the greedy policy \( \pi_K \) 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...

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

Bringing everything together...

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

\[
\|\epsilon_k\|_{\rho} = \|Q_k - \hat{Q}_k\|_{\rho} \leq 4\|Q_k - f_{\alpha_k^*}\|_{\rho}
\]

\[
+ O\left((V_{\text{max}} + L\|\alpha_k^*\|) \sqrt{\log \frac{1}{\delta}} \right)
\]

\[
+ O\left(V_{\text{max}} \sqrt{\frac{d \log n/\delta}{n}} \right)
\]
The Final Bound

Theorem (see e.g., Munos,’03)

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

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

$$+ O \left( \frac{\gamma^K}{(1 - \gamma)^3} V_{\max}^2 \right)$$
LinearFQI with a space $\mathcal{F}$ of $d$ features, with $n$ samples at each iteration returns a policy $\pi_K$ 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
⇒ 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 \( \pi_K \) after \( K \) iterations such that

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

The **approximation** error is **worse** than in regression.
The Final Bound

The inherent Bellman error

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

$$= \inf_{f \in \mathcal{F}} \|T\hat{Q}_{k-1} - f\|_\rho$$

$$\leq \inf_{f \in \mathcal{F}} \|Tf_{\alpha_{k-1}} - f\|_\rho$$

$$\leq \sup_{g \in \mathcal{F}} \inf_{f \in \mathcal{F}} \|Tg - f\|_\rho = d(\mathcal{F}, T\mathcal{F})$$

**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 $\pi_K$ after $K$ iterations such that

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

$$+ \ O \left(\frac{\gamma^K}{(1 - \gamma)^3} V_{\text{max}}^2 \right)$$

The dependency on $\gamma$ 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 $\pi_K$ after $K$ iterations such that

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

$$+ O\left( \frac{\gamma^K}{(1 - \gamma)^3} V_{\text{max}}^2 \right)$$

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 $\pi_K$ after $K$ iterations such that

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

$$+ O\left(\frac{\gamma^K}{(1 - \gamma)^3} V_{\max}^2 \right)$$

The smallest eigenvalue of the Gram matrix

$\Rightarrow$ design the features so as to be **orthogonal** w.r.t. $\rho$
The Final Bound

Theorem

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

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

$$+ O\left(\frac{\gamma^K}{(1 - \gamma)^3} V_{\text{max}}^2\right)$$

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

Approximation space

Samples
(sampling strategy, number)

Performance

Propagation

Dynamic programming algorithm

Markov decision process

Concentrability $C_{\mu,\rho}$
Range $V_{\text{max}}$

$Q_k - \hat{Q}_k$

Approximation algorithm

number $n$, sampling dist. $\rho$

d($F, T F$
size $d$, features $\omega$

$\hat{Q}_k$

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)eplace, (K)eep\}.

**Cost:** $c(x, R) = c(x, K) = c(x)$ maintenance plus extra costs.

**Dynamics:** $p(\cdot | x, R) = \exp(\beta)$ with density $d(y) = \beta \exp(-\beta y) I\{y \geq 0\},$

$p(\cdot | x, K) = x + \exp(\beta)$ with density $d(y-x) = \beta \exp(-\beta y)$. 

**Problem:** Minimize the discounted expected cost over an infinite horizon.
**Example: the Optimal Replacement Problem**

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

**Action:** \{\text{(R)eplace}, \text{(K)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: \{\text{(R)eplace, (K)eep}\}.
Cost:
  - \(c(x, R) = C\)
  - \(c(x, K) = c(x)\) maintenance plus extra costs.
Dynamics:
  - \(p(\cdot|x, R) = \exp(\beta)\) with density \(d(y) = \beta \exp^{-\beta y} \mathbb{1}\{y \geq 0\}\),
  - \(p(\cdot|x, K) = x + \exp(\beta)\) with density \(d(y - x)\).
Example: the Optimal Replacement Problem

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

**Action:** \{("R")eplace, ("K")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

- Management cost
- Value function
Example: the Optimal Replacement Problem

Optimal value function

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

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 $\{TV_0(x_n)\}_{1 \leq n \leq N}$. Right: the approximation $V_1 \in \mathcal{F}$ of the target function $TV_0$. 
Example: the Optimal Replacement Problem

Figure: Left: the target values computed as $\{TV_1(x_n)\}_{1 \leq n \leq N}$. Center: the approximation $V_2 \in F$ of $TV_1$. Right: the approximation $V_n \in F$ 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 $\pi_0$ be any stationary policy

2. At each iteration $k = 1, 2, \ldots, K$
   - **Policy evaluation** given $\pi_k$, compute $V_k = V^{\pi_k}$.
   - **Policy improvement**: compute the greedy policy
     \[
     \pi_{k+1}(x) \in \arg\max_{a \in A} \left[ r(x, a) + \gamma \sum_y p(y|x, a) V^{\pi_k}(y) \right].
     \]

3. Return the last policy $\pi_K$
Policy Iteration: the Idea

1. Let $\pi_0$ be any stationary policy
2. At each iteration $k = 1, 2, \ldots, K$
   - **Policy evaluation** given $\pi_k$, compute $V_k = V^{\pi_k}$.
   - **Policy improvement**: compute the greedy policy
     \[
     \pi_{k+1}(x) \in \arg\max_{a \in A} \left[ r(x, a) + \gamma \sum_y p(y|x, a) V^{\pi_k}(y) \right].
     \]
3. Return the last policy $\pi_K$

- **Problem**: how can we approximate $V^{\pi_k}$?
- **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.

\[ V^* - V^{\pi_k} \]

The asymptotic performance of the policies \( \pi_k \) generated by the API algorithm is related to the approximation error as:

\[
\limsup_{k \to \infty} \left\| V^* - V^{\pi_k} \right\|_\infty \leq \frac{2\gamma}{(1 - \gamma)^2} \limsup_{k \to \infty} \left\| V_k - V^{\pi_k} \right\|_\infty
\]

**Proposition**

The asymptotic performance of the policies \( \pi_k \) generated by the API algorithm is related to the approximation error as:

\[
\limsup_{k \to \infty} \left\| V^* - V^{\pi_k} \right\|_\infty \leq \frac{2\gamma}{(1 - \gamma)^2} \limsup_{k \to \infty} \left\| V_k - V^{\pi_k} \right\|_\infty
\]
Least-Squares Policy Iteration (LSPI)

LSPI uses

- Linear space to approximate value functions*

\[
\mathcal{F} = \left\{ f(x) = \sum_{j=1}^{d} \alpha_j \varphi_j(x), \quad \alpha \in \mathbb{R}^d \right\}
\]
Least-Squares Policy Iteration (LSPI)

LSPI uses

- Linear space to approximate value functions

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

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

*In practice we use approximations of action-value functions.
Least-Squares Temporal-Difference Learning (LSTD)

- $V^\pi$ may not belong to $\mathcal{F}$
- Best approximation of $V^\pi$ in $\mathcal{F}$ is

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

($\Pi$ is the projection onto $\mathcal{F}$)
Least-Squares Temporal-Difference Learning (LSTD)

- $V^\pi$ 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

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

"\text{A. LAZARIC – Reinforcement Learning Algorithms Dec 2nd, 2014 - 51/82}"

Least-Squares Temporal-Difference Learning (LSTD)

\[ V_{TD} = \Pi_\rho T^\pi V_{TD} \]

- The projection \( \Pi_\rho \) is orthogonal \textit{in expectation} w.r.t. the space \( \mathcal{F} \) \textit{spanned} by the features \( \varphi_1, \ldots, \varphi_d \)

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

\[ \langle T^\pi V_{TD} - V_{TD}, \varphi_i \rangle_\rho = 0 \]
Least-Squares Temporal-Difference Learning (LSTD)

\[ V_{TD} = \Pi_{\rho} T^\pi V_{TD} \]

- The projection \( \Pi_{\rho} \) is orthogonal \textit{in expectation} w.r.t. the space \( \mathcal{F} \) spanned by the features \( \varphi_1, \ldots, \varphi_d \)

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

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

- 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)

\[ V_{TD} = \Pi_{\rho} T^{\pi} V_{TD} \]

- The projection \( \Pi_{\rho} \) is orthogonal \textit{in expectation} w.r.t. the space \( \mathcal{F} \) spanned by the features \( \varphi_1, \ldots, \varphi_d \)

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

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

- 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 \( \alpha_{TD} \) 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)

\[ V_{TD} = \Pi_\rho T^\pi \cdot V_{TD} \]

\[ \downarrow \]

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

\[ \downarrow \]

\[ A\alpha_{TD} = b \]
Least-Squares Temporal-Difference Learning (LSTD)

- **Problem:** In general, $\Pi_\rho T^\pi$ is *not a contraction* and does not have a fixed-point.

- **Solution:** If $\rho = \rho^\pi$ (*stationary dist. of $\pi$*) then $\Pi_{\rho^\pi} T^\pi$ has a unique fixed-point.
Least-Squares Temporal-Difference Learning (LSTD)

- **Problem:** In general, $\Pi_{\rho} T^\pi$ is *not a contraction* and does not have a fixed-point.

- **Solution:** If $\rho = \rho^\pi$ (*stationary dist. of $\pi$*) then $\Pi_{\rho^\pi} T^\pi$ has a unique fixed-point.

- **Problem:** In general, $\Pi_{\rho} T^\pi$ cannot be computed (because *unknown*)

- **Solution:** Use *samples* coming from a “trajectory” of $\pi$. 
Least-Squares Policy Iteration (LSPI)

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

1. Generate a trajectory of length $n$ from the stationary dist. $\rho_{\pi_k}(x_1, \pi_k(x_1), r_1, x_2, \pi_k(x_2), r_2, \ldots, x_{n-1}, \pi_k(x_{n-1}), r_{n-1}, x_n)$

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

   $\hat{A}_{ik} = \frac{1}{n} \sum_{t=1}^{n} \left( \phi_j(x_t) - \gamma \phi_j(x_{t+1}) \phi_i(x_t) \right) \approx \langle (I - \gamma P_{\pi_k}) \phi_j, \phi_i \rangle_{\rho_{\pi_k}}$

   $\hat{b}_k = \frac{1}{n} \sum_{t=1}^{n} \phi_i(x_t) r_t \approx \langle r_{\pi_k}, \phi_i \rangle_{\rho_{\pi_k}}$

3. Solve the linear system $\alpha_k = \hat{A}_k^{-1} \hat{b}_k$

4. Compute the greedy policy $\pi_{k+1}$ w.r.t. $\hat{V}_k = f(\alpha_k)$

Return the last policy $\pi_K$
Least-Squares Policy Iteration (LSPI)

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

Initial policy $\pi_0$
Least-Squares Policy Iteration (LSPI)

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

Initial policy \( \pi_0 \)

For \( k = 1, \ldots, K \)

1. Generate a trajectory of length \( n \) from the stationary dist. \( \rho_{\pi_k}(x_1, \pi_k(x_1), r_1, x_2, \pi_k(x_2), r_2, \ldots, x_{n-1}, \pi_k(x_{n-1}), r_{n-1}, x_n) \)

2. Compute the empirical matrix \( \hat{A}_k \) and the vector \( \hat{b}_k \):

   \[
   \hat{A}_k[i,j] = \frac{1}{n} \sum_{t=1}^{n} (\phi_j(x_t) - \gamma \phi_j(x_{t+1}) \phi_i(x_t)) \approx \langle (I - \gamma P_{\pi_k}) \phi_j, \phi_i \rangle_{\rho_{\pi_k}}
   \]

   \[
   \hat{b}_k[i] = \frac{1}{n} \sum_{t=1}^{n} \phi_i(x_t) r_t \approx \langle r_{\pi_k}, \phi_i \rangle_{\rho_{\pi_k}}
   \]

3. Solve the linear system

   \[
   \alpha_k = \hat{A}_k^{-1} \hat{b}_k
   \]

4. Compute the greedy policy \( \pi_{k+1} \) w.r.t. \( \hat{V}_k = f(\alpha_k) \)

Return the last policy \( \pi_K \)
Least-Squares Policy Iteration (LSPI)

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

Initial policy $\pi_0$

For $k = 1, \ldots, K$

1. Generate a trajectory of length $n$ from the stationary dist. $\rho^{\pi_k}$

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

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

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

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

3. Solve the linear system $\alpha_k = \hat{A}_k^{-1} \hat{b}_k$

4. Compute the greedy policy $\pi_{k+1}$ w.r.t. $\hat{V}_k = f^{\alpha_k}$

Return the last policy $\pi_K$
Least-Squares Policy Iteration (LSPI)

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

Initial policy $\pi_0$

For $k = 1, \ldots, K$

1. Generate a trajectory of length $n$ from the stationary dist. $\rho^{\pi_k}$

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

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

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

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

3. Solve the linear system $\alpha_k = \hat{A}_k^{-1} \hat{b}_k$


Least-Squares Policy Iteration (LSPI)

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

Initial policy $\pi_0$

For $k = 1, \ldots, K$

1. Generate a trajectory of length $n$ from the stationary dist. $\rho^{\pi_k}$
$$ (x_1, \pi_k(x_1), r_1, x_2, \pi_k(x_2), r_2, \ldots, x_{n-1}, \pi_k(x_{n-1}), r_{n-1}, x_n) $$

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

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

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

3. Solve the linear system $\alpha_k = \hat{A}_k^{-1} \hat{b}_k$

4. Compute the greedy policy $\pi_{k+1}$ w.r.t. $\hat{V}_k = f_{\alpha_k}$
Least-Squares Policy Iteration (LSPI)

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

Initial policy $\pi_0$

For $k = 1, \ldots, K$

1. Generate a trajectory of length $n$ from the stationary dist. $\rho^{\pi_k}$
   \[
   (x_1, \pi_k(x_1), r_1, x_2, \pi_k(x_2), r_2, \ldots, x_{n-1}, \pi_k(x_{n-1}), r_{n-1}, x_n)
   \]

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

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

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

3. Solve the linear system $\alpha_k = \hat{A}_k^{-1} \hat{b}_k$

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

**Return** the last policy $\pi_K$
Least-Squares Policy Iteration (LSPI)

1. Generate a trajectory of length $n$ from the stationary dist. $\rho^{\pi_k}$
   $$(x_1, \pi_k(x_1), r_1, x_2, \pi_k(x_2), r_2, \ldots, x_{n-1}, \pi_k(x_{n-1}), r_{n-1}, x_n)$$

- The first few samples may be discarded because not actually drawn from the stationary distribution $\rho^{\pi_k}$
- Off-policy samples could be used with importance weighting
- In practice i.i.d. states drawn from an arbitrary distribution (but with actions $\pi_k$) may be used
4. Compute the greedy policy $\pi_{k+1}$ w.r.t. $\hat{V}_k = f_{\alpha_k}$

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

$$\pi_{k+1}(x) = \arg\max_a \hat{Q}_k(x, a)$$
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. $\rho^{\pi_k}$
   
   $$(x_1, \pi_k(x_1), r_1, x_2, \pi_k(x_2), r_2, \ldots, x_{n-1}, \pi_k(x_{n-1}), r_{n-1}, x_n)$$

2. ... 

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

**Problem:** This process may be unstable because $\pi_k$ does not cover the state space properly
LSTD Algorithm

When \( n \to \infty \) then \( \hat{A} \to A \) and \( \hat{b} \to b \), and thus,

\[
\hat{\alpha}_{TD} \to \alpha_{TD} \text{ and } \hat{V}_{TD} \to V_{TD}
\]

Proposition (LSTD Performance)

If LSTD is used to estimate the value of \( \pi \) with an \textit{infinite} number of samples drawn from the stationary distribution \( \rho^\pi \) then

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

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 Algorithm

When \( n \to \infty \) then \( \hat{A} \to A \) and \( \hat{b} \to b \), and thus,

\[
\hat{\alpha}_{TD} \to \alpha_{TD} \text{ and } \hat{V}_{TD} \to V_{TD}
\]

**Proposition (LSTD Performance)**

If LSTD is used to estimate the value of \( \pi \) with an *infinite* number of samples drawn from the stationary distribution \( \rho^\pi \) then

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

**Problem:** we don’t have an infinite number of samples...
LSTD Algorithm

When \( n \to \infty \) then \( \hat{A} \to A \) and \( \hat{b} \to b \), and thus,

\[
\hat{\alpha}_\text{TD} \to \alpha_{\text{TD}} \text{ and } \hat{V}_\text{TD} \to V_{\text{TD}}
\]

**Proposition (LSTD Performance)**

If LSTD is used to estimate the value of \( \pi \) with an *infinite* number of samples drawn from the stationary distribution \( \rho^\pi \) 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:** \( V_{\text{TD}} \) is a fixed point solution and not a standard machine learning problem...
LSTD Error Bound

**Assumption:** The Markov chain induced by the policy $\pi_k$ has a stationary distribution $\rho^{\pi_k}$ and it is ergodic and $\beta$-mixing.
LSTD Error Bound

**Assumption:** The Markov chain induced by the policy $\pi_k$ has a stationary distribution $\rho_{\pi_k}$ and it is ergodic and $\beta$-mixing.

**Theorem (LSTD Error Bound)**

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

$$||V_{\pi_k} - \hat{V}_k||_{\rho_{\pi_k}} \leq \frac{c}{\sqrt{1 - \gamma^2}} \inf_{f \in \mathcal{F}} ||V_{\pi_k} - f||_{\rho_{\pi_k}} + O \left( \sqrt{\frac{d \log(d/\delta)}{n \nu}} \right)$$
LSTD Error Bound

\[ \| V^\pi - \hat{V} \|_{\rho^\pi} \leq \frac{c}{\sqrt{1 - \gamma^2}} \inf_{f \in F} \| V^\pi - f \|_{\rho^\pi} + O \left( \sqrt{\frac{d \log(d/\delta)}{n \nu}} \right) \]

- **Approximation error**: it depends on how well the function space \( F \) can approximate the value function \( V^\pi \)

- **Estimation error**: it depends on the number of samples \( n \), the dim of the function space \( d \), the smallest eigenvalue of the Gram matrix \( \nu \), the mixing properties of the Markov chain (hidden in \( O \))
LSTD Error Bound

\[
\|V^{\pi_k} - \hat{V}_k\|_{\rho^{\pi_k}} \leq \frac{c}{\sqrt{1 - \gamma^2}} \inf_{f \in F} \|V^{\pi_k} - f\|_{\rho^{\pi_k}} + O\left(\sqrt{n \nu_k} \log(d/\delta) \right)
\]

- \( n \) number of samples and \( d \) dimensionality
LSTD Error Bound

\[
\|V_{\pi k} - \hat{V}_k\|_{\rho_{\pi k}} \leq \frac{c}{\sqrt{1 - \gamma^2}} \inf_{f \in \mathcal{F}} \|V_{\pi k} - f\|_{\rho_{\pi k}} + O \left( \sqrt{d \log(\frac{d}{\delta})} \right) \frac{n \nu_k}{\nu_k} 
\]

- \( \nu_k = \) 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)

- \( \beta \)-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 $\pi_K$ is

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

with probability $1 - \delta$. 
**Theorem (LSPI Error Bound)**

If LSPI is run over $K$ iterations, then the performance loss policy $\pi_K$ is

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

with probability $1 - \delta$.

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

If LSPI is run over $K$ iterations, then the performance loss policy $\pi_K$ is

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

with probability $1 - \delta$.

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

- **Estimation error:** depends on $n, d, \nu_\rho, K$
LSPI Error Bound

**Theorem (LSPI Error Bound)**

If LSPI is run over $K$ iterations, then the performance loss policy $\pi_K$ 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:** $E_0(\mathcal{F}) = \sup_{\pi \in \mathcal{G}} \inf_{f \in \mathcal{F}} \|V^\pi - f\|_{\rho^\pi}$

- **Estimation error:** depends on $n, d, \nu_\rho, K$

- **Initialization error:** error due to the choice of the initial value function or initial policy $|V^* - V^{\pi_0}|$
LSPI Error Bound

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

Lower-Bounding Distribution

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

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

Lower-Bounding Distribution

There exists a distribution \( \rho \) such that for any policy \( \pi \in \mathcal{G}(\tilde{\mathcal{F}}) \), we have \( \rho \leq C \rho^\pi \), where \( C < \infty \) is a constant and \( \rho^\pi \) is the stationary distribution of \( \pi \). 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

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

$$V_{BR} = \arg \min_{V \in \mathcal{F}} \| T^\pi V - V \|_{2,\mu}$$
Bellman Residual Minimization (BRM): the idea

The mapping $\alpha \rightarrow T^{\pi} V_\alpha - V_\alpha$ is affine

The function $\alpha \rightarrow \|T^{\pi} V_\alpha - V_\alpha\|_\mu^2$ is quadratic

$\Rightarrow$ The minimum is obtained by computing the \textit{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 $\phi_i$ are linearly independent w.r.t. $\mu$
Bellman Residual Minimization (BRM): the idea

*Remark:* the system admits a solution whenever the features $\phi_i$ are *linearly independent* w.r.t. $\mu$

*Remark:* let $\{\psi_i = \phi_i - \gamma P^\pi \phi_i\}_{i=1}^d$, 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 $\mu_\pi$ is the *stationary policy* of $\pi$, 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

\[
\hat{B}(V) = \frac{1}{n} \sum_{t=1}^{n} \left[ V(X_t) - \left( R_t + \gamma \hat{V}(Y_t) \right) \right] \hat{V}(X_t).
\]
Problem: this estimator is \textit{biased and not consistent}! In fact,

\[
\mathbb{E}[\hat{B}(V)] = \mathbb{E}\left[\left(V(X_t) - \mathcal{T}^\pi V(X_t) + \mathcal{T}^\pi V(X_t) - \hat{T} V(X_t)\right)^2\right]
\]

\[
= \left\| \mathcal{T}^\pi V - V \right\|_{\mu}^2 + \mathbb{E}\left[\left(\mathcal{T}^\pi V(X_t) - \hat{T} V(X_t)\right)^2\right]
\]

\[\Rightarrow\] minimizing \( \hat{B}(V) \) \textit{does not} correspond to minimizing \( B(V) \) (even when \( n \rightarrow \infty \)).
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

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

$$\Rightarrow \hat{B} \to B \text{ for } n \to \infty.$$
The function $\alpha \rightarrow \hat{B}(V_\alpha)$ is quadratic and we obtain the linear system

$$
\hat{A}_{i,j} = \frac{1}{n} \sum_{t=1}^{n} \left[ \phi_i(X_t) - \gamma \phi_i(Y_t) \right] \left[ \phi_j(X_t) - \gamma \phi_j(Y'_t) \right],
$$

$$
\hat{b}_i = \frac{1}{n} \sum_{t=1}^{n} \left[ \phi_i(X_t) - \gamma \frac{\phi_i(Y_t) + \phi_i(Y'_t)}{2} \right] R_t.
$$
BRM: the approximation error

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

\[ V^\pi - V = V^\pi - T^\pi V + T^\pi V - V = \gamma P^\pi (V^\pi - V) + T^\pi V - V, \]

\[ (I - \gamma P^\pi)(V^\pi - V) = T^\pi V - V, \]

taking the norm both sides we obtain

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

and

\[ \| T^\pi V_{BR} - V_{BR} \| = \inf_{V \in \mathcal{F}} \| T^\pi V - V \| \leq (1 + \gamma \| P^\pi \|) \inf_{V \in \mathcal{F}} \| V^\pi - V \|. \]
Proof. If we consider the stationary distribution $\mu_\pi$, 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 \( \mu^\pi \).
How to solve \textit{approximately} an MDP

\textbf{Approximate Dynamic Programming}

(a.k.a. Batch Reinforcement Learning)

\textbf{Approximate Value Iteration}

\textbf{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
  $$L(\theta) = \mathbb{E}\left[r(x, a) + \gamma \max Q(y, a'; \theta') - Q(x, a; \theta)^2\right]$$
- Compute the gradient
  $$\nabla_\theta L(\theta) = \mathbb{E}\left[(r(x, a) + \gamma \max Q(y, a'; \theta') - Q(x, a; \theta))\nabla_\theta Q(x, a; \theta)\right]$$
- 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

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

- Compute the gradient

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

- 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 $i = 1, \ldots, n$

1. Set $t = 0$
2. Set initial state $x_0$
3. While ($x_t$ not terminal)
   
   3.1 Take action $a_t$ with $\epsilon$-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 $(y - Q(x, a; \theta_i))^2$ and update $\theta_{i+1}$

   EndWhile

EndFor
Q-learning with Function Approximation

Why it works:

- **Action reply**: de-correlate changes to $\theta$ 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

<table>
<thead>
<tr>
<th>Game</th>
<th>At human-level or above</th>
<th>Below human-level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Video Pinball</td>
<td>29.9%</td>
<td>100%</td>
</tr>
<tr>
<td>Boxing</td>
<td>19.7%</td>
<td>80%</td>
</tr>
<tr>
<td>Breakout</td>
<td>86%</td>
<td>10%</td>
</tr>
<tr>
<td>Robotank</td>
<td>66%</td>
<td>34%</td>
</tr>
<tr>
<td>Atlantis</td>
<td>49%</td>
<td>51%</td>
</tr>
<tr>
<td>Crazy Climber</td>
<td>29%</td>
<td>71%</td>
</tr>
<tr>
<td>Gopher</td>
<td>27%</td>
<td>73%</td>
</tr>
<tr>
<td>Demon Attack</td>
<td>27%</td>
<td>73%</td>
</tr>
<tr>
<td>Name This Game</td>
<td>25%</td>
<td>75%</td>
</tr>
<tr>
<td>Kull</td>
<td>23%</td>
<td>77%</td>
</tr>
<tr>
<td>Road Runner</td>
<td>19%</td>
<td>81%</td>
</tr>
<tr>
<td>Karan_Master</td>
<td>16%</td>
<td>84%</td>
</tr>
<tr>
<td>Space Invaders</td>
<td>16%</td>
<td>84%</td>
</tr>
<tr>
<td>Beam Rider</td>
<td>11%</td>
<td>89%</td>
</tr>
<tr>
<td>Enduro</td>
<td>9%</td>
<td>91%</td>
</tr>
<tr>
<td>Q-Bert</td>
<td>9%</td>
<td>91%</td>
</tr>
<tr>
<td>H.E.R.O.</td>
<td>9%</td>
<td>91%</td>
</tr>
<tr>
<td>Asterix</td>
<td>8%</td>
<td>92%</td>
</tr>
<tr>
<td>Battle Zone</td>
<td>7%</td>
<td>93%</td>
</tr>
<tr>
<td>Wizard of Wor</td>
<td>7%</td>
<td>93%</td>
</tr>
<tr>
<td>Chopper Command</td>
<td>5%</td>
<td>95%</td>
</tr>
<tr>
<td>Bank Heist</td>
<td>4%</td>
<td>96%</td>
</tr>
<tr>
<td>River Raid</td>
<td>4%</td>
<td>96%</td>
</tr>
<tr>
<td>Zaxxon</td>
<td>4%</td>
<td>96%</td>
</tr>
<tr>
<td>Aliens</td>
<td>2%</td>
<td>98%</td>
</tr>
<tr>
<td>Venture</td>
<td>2%</td>
<td>98%</td>
</tr>
<tr>
<td>Seaquest</td>
<td>2%</td>
<td>98%</td>
</tr>
<tr>
<td>Double Dunk</td>
<td>0%</td>
<td>100%</td>
</tr>
<tr>
<td>Bowling</td>
<td>0%</td>
<td>100%</td>
</tr>
<tr>
<td>Ms. Pac-Man</td>
<td>0%</td>
<td>100%</td>
</tr>
<tr>
<td>Antes</td>
<td>0%</td>
<td>100%</td>
</tr>
<tr>
<td>Frogger</td>
<td>0%</td>
<td>100%</td>
</tr>
<tr>
<td>Computer's Revenge</td>
<td>0%</td>
<td>100%</td>
</tr>
<tr>
<td>Mr. Do</td>
<td>0%</td>
<td>100%</td>
</tr>
</tbody>
</table>

Don: Best linear learner
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

⇒ would it still work in, eg, financial applications?
Reinforcement Learning

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