Approximate Dynamic Programming

A. LAZARIC (*SequeL Team @INRIA-Lille*)
Ecole Centrale - Option DAD

Value Iteration: the Idea

1. Let V_0 be *any* vector in R^N

Value Iteration: the Idea

1. Let V_0 be *any* vector in R^N
2. At each iteration $k = 1, 2, \ldots, K$

Value Iteration: the Idea

1. Let V_0 be *any* vector in R^N
2. At each iteration $k = 1, 2, \ldots, K$
   - Compute $V_{k+1} = \mathcal{T} V_k$

Value Iteration: the Idea

1. Let V_0 be *any* vector in R^N
2. At each iteration $k = 1, 2, \ldots, K$
   - Compute $V_{k+1} = \mathcal{T} V_k$
3. Return the *greedy* policy

$$\pi_K(x) \in \arg\max_{a \in A} \Big[r(x,a) + \gamma \sum_y p(y|x,a) V_K(y) \Big].$$

Value Iteration: the Guarantees

- From the *fixed point* property of $\mathcal{T}$:

$$\lim_{k \to \infty} V_k = V^*$$

Value Iteration: the Guarantees

- From the *fixed point* property of $\mathcal{T}$:

$$\lim_{k \to \infty} V_k = V^*$$

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

$$||V_{k+1} - V^*||_\infty \leq \gamma^{k+1} ||V_0 - V^*||_\infty \to 0$$

Value Iteration: the Guarantees

- From the *fixed point* property of $\mathcal{T}$:

$$\lim_{k\to\infty} V_k = V^*$$

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

$$||V_{k+1} - V^*||_\infty \leq \gamma^{k+1} ||V_0 - V^*||_\infty \to 0$$

Problem: what if $V_{k+1} \neq \mathcal{T} V_k$??

Policy Iteration: the Idea

1. Let π_0 be *any* stationary policy

Policy Iteration: the Idea

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

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 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

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

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 Guarantees

The policy iteration algorithm generates a sequences of policies with *non-decreasing* performance

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

and it converges to π^* in a *finite* number of iterations.

Policy Iteration: the Guarantees

The policy iteration algorithm generates a sequences of policies with *non-decreasing* performance

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

and it converges to π^* in a *finite* number of iterations.

Problem: what if $V_k \neq V^{\pi_k}$??

Sources of Error

- **Approximation error**. If X is *large* or *continuous*, value functions V cannot be *represented* correctly
  $\Rightarrow$ use an *approximation space* $\mathcal{F}$

Sources of Error

- **Approximation error**. If X is *large* or *continuous*, value functions V cannot be *represented* correctly
  $\Rightarrow$ use an *approximation space* $\mathcal{F}$
- **Estimation error**. If the reward r and dynamics p are *unknown*, the Bellman operators $\mathcal{T}$ and $\mathcal{T}^\pi$ cannot be *computed* exactly
  $\Rightarrow$ *estimate* the Bellman operators from *samples*

Outline

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

Question: how do we compute V?

From Approximation Error to Performance Loss

Question: how do we compute 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 V?

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

Objective: given an *approximation space* $\mathcal{F}$, compute an approximation V which is as close as possible to the *best approximation* of V^* in $\mathcal{F}$, i.e.

$$V \approx \arg\inf_{f \in \mathcal{F}} ||V^* - f||$$

Outline

Performance Loss

Approximate Value Iteration

Approximate Policy Iteration

Approximate Value Iteration: the Idea

Let $\mathcal{A}$ be an *approximation operator*.

Approximate Value Iteration: the Idea

Let $\mathcal{A}$ be an *approximation operator*.

1. Let V_0 be *any* vector in R^N
2. At each iteration $k = 1, 2, \dots, K$
   - Compute $V_{k+1} = \mathcal{A}\mathcal{T} V_k$
3. Return the *greedy* policy

$$\pi_K(x) \in \arg\max_{a \in A} \Big[r(x, a) + \gamma \sum_y p(y|x, a) V_K(y) \Big].$$

Approximate Value Iteration: the Idea

Let $\mathcal{A} = \Pi_\infty$ be a projection operator in L_∞-norm, which corresponds to

$$V_{k+1} = \Pi_\infty \mathcal{T} V_k = \arg \inf_{V \in \mathcal{F}} \|\mathcal{T} V_k - V\|_\infty$$

Approximate Value Iteration: convergence

Proposition

The projection Π_∞ is a *non-expansion* and the joint operator $\Pi_\infty \mathcal{T}$ is a *contraction*.
Then there exists a unique fixed point $\tilde{V} = \Pi_\infty \mathcal{T} \tilde{V}$ which guarantees the *convergence* of AVI.

Approximate Value Iteration: performance loss

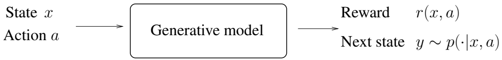
Proposition (Bertsekas & Tsitsiklis, 1996)

Let V^K be the function returned by AVI after K iterations and π_K its corresponding greedy policy. Then

$$\|V^* - V^{\pi_K}\|_\infty \leq \frac{2\gamma}{(1-\gamma)^2} \underbrace{\max_{0\leq k<K} \|\mathcal{T} V_k - \mathcal{A}\mathcal{T} V_k\|_\infty}_{\textit{worst approx. error}} + \frac{2\gamma^{K+1}}{1-\gamma} \underbrace{\|V^* - V_0\|_\infty}_{\textit{initial error}}.$$

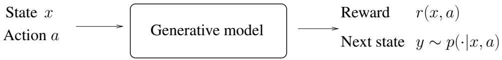
Fitted Q-iteration with linear approximation

Assumption: access to a generative model.



Fitted Q-iteration with linear approximation

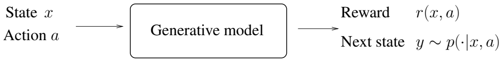
Assumption: access to a generative model.



Idea: work with Q-functions and linear spaces.

Fitted Q-iteration with linear approximation

Assumption: access to a generative model.



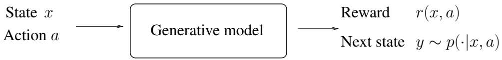
Idea: work with Q-functions and linear spaces.

- Q^* is the unique fixed point of $\mathcal{T}$ defined over $X \times A$ as:

$$\mathcal{T}Q(x, a) = \sum_y p(y|x, a)[r(x, a, y) + \gamma \max_b Q(y, b)].$$

Fitted Q-iteration with linear approximation

Assumption: access to a generative model.



Idea: work with Q-functions and linear spaces.

- Q^* is the unique fixed point of $\mathcal{T}$ defined over $X \times A$ as:

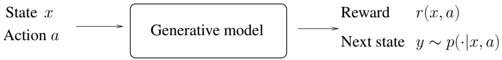
$$\mathcal{T}Q(x,a) = \sum_y p(y|x,a)[r(x,a,y) + \gamma \max_b Q(y,b)].$$

- $\mathcal{F}$ is a space defined by d features $\phi_1, \ldots, \phi_d : X \times A \to \mathbb{R}$ as:

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

Fitted Q-iteration with linear approximation

Assumption: access to a generative model.



Idea: work with Q-functions and linear spaces.

- Q^* is the unique fixed point of $\mathcal{T}$ defined over $X \times A$ as:

$$\mathcal{T}Q(x,a) = \sum_y p(y|x,a)[r(x,a,y) + \gamma \max_b Q(y,b)].$$

- $\mathcal{F}$ is a space defined by d features $\phi_1, \ldots, \phi_d : X \times A \to \mathbb{R}$ as:

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

$\Rightarrow$ At each iteration compute $Q_{k+1} = \Pi_\infty \mathcal{T} Q_k$

Fitted Q-iteration with linear approximation

$\Rightarrow$ At each iteration compute $Q_{k+1} = \Pi_\infty \mathcal{T} Q_k$

Problems:

- the Π_∞ operator cannot be computed *efficiently*
- the Bellman operator $\mathcal{T}$ is often *unknown*

Fitted Q-iteration with linear approximation

Problem: the Π_∞ operator cannot be computed *efficiently*.

Fitted Q-iteration with linear approximation

Problem: the Π_∞ operator cannot be computed *efficiently*.

Let μ a distribution over X. We use a projection in $L_{2,\mu}$-norm onto the space $\mathcal{F}$:

$$Q_{k+1} = \arg\min_{Q \in \mathcal{F}} ||Q - \mathcal{T} Q_k||^2_\mu.$$

Fitted Q-iteration with linear approximation

Problem: the Bellman operator $\mathcal{T}$ is often *unknown*.

Fitted Q-iteration with linear approximation

Problem: the Bellman operator $\mathcal{T}$ is often *unknown*.

1. Sample n state actions (X_i, A_i) with $X_i \sim \mu$ and A_i random,

Fitted Q-iteration with linear approximation

Problem: the Bellman operator $\mathcal{T}$ is often *unknown*.

1. Sample n state actions (X_i, A_i) with $X_i \sim \mu$ and A_i random,
2. Simulate $Y_i \sim p(\cdot|X_i, A_i)$ and $R_i = r(X_i, A_i, Y_i)$ with the generative model,

Fitted Q-iteration with linear approximation

Problem: the Bellman operator $\mathcal{T}$ is often *unknown*.

1. Sample n state actions (X_i, A_i) with $X_i \sim \mu$ and A_i random,
2. Simulate $Y_i \sim p(\cdot|X_i, A_i)$ and $R_i = r(X_i, A_i, Y_i)$ with the generative model,
3. Estimate $\mathcal{T} Q_k(X_i, A_i)$ with

$$Z_i = R_i + \gamma \max_{a \in A} Q_k(Y_i, a)$$

Fitted Q-iteration with linear approximation

Problem: the Bellman operator $\mathcal{T}$ is often *unknown*.

1. Sample n state actions (X_i, A_i) with $X_i \sim \mu$ and A_i random,
2. Simulate $Y_i \sim p(\cdot|X_i, A_i)$ and $R_i = r(X_i, A_i, Y_i)$ with the generative model,
3. Estimate $\mathcal{T}Q_k(X_i, A_i)$ with

$$Z_i = R_i + \gamma \max_{a \in A} Q_k(Y_i, a)$$

(unbiased $\mathbb{E}[Z_i|X_i, A_i] = \mathcal{T}Q_k(X_i, A_i)$),

Fitted Q-iteration with linear approximation

At each iteration k compute Q_{k+1} as

$$Q_{k+1} = \arg\min_{Q_\alpha \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \big[Q_\alpha(X_i, A_i) - Z_i \big]^2$$

Fitted Q-iteration with linear approximation

At each iteration k compute Q_{k+1} as

$$Q_{k+1} = \arg\min_{Q_\alpha \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \big[Q_\alpha(X_i, A_i) - Z_i \big]^2$$

$\Rightarrow$ Since Q_α is a linear function in α, the problem is a simple *quadratic minimization* problem with *closed form* solution.

Other implementations

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

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

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

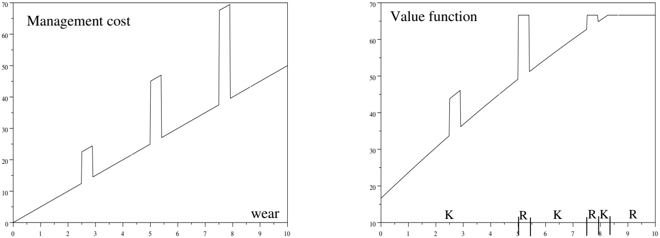
Optimal policy: action that attains the minimum

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

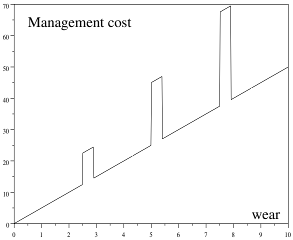


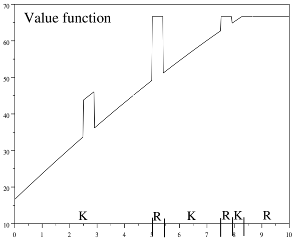
Example: the Optimal Replacement Problem

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

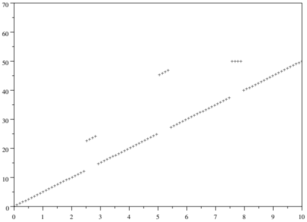




Linear approximation space $\mathcal{F} := \left\{V_n(x) = \sum_{k=1}^{20} \alpha_k \cos(k\pi \frac{x}{x_{\max}})\right\}.$

Example: the Optimal Replacement Problem

Collect N sample on a uniform grid.



Example: the Optimal Replacement Problem

Collect N sample on a uniform grid.

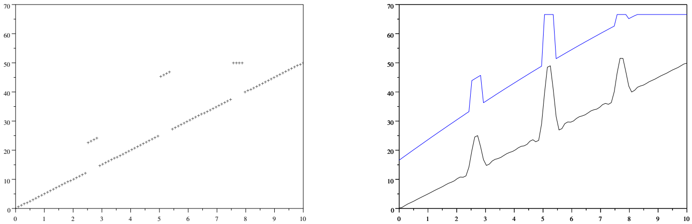


Figure: Left: the *target* values computed as $\{\mathcal{T}V_0(x_n)\}_{1 \leq n \leq N}$. Right: the approximation $V_1 \in \mathcal{F}$ of the target function $\mathcal{T}V_0$.

Example: the Optimal Replacement Problem

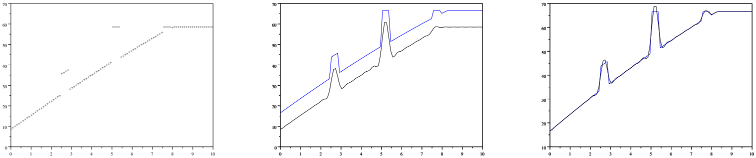


Figure: Left: the *target* values computed as $\{\mathcal{T}V_1(x_n)\}_{1\leq n\leq N}$. Center: the approximation $V_2 \in \mathcal{F}$ of $\mathcal{T}V_1$. Right: the approximation $V_n \in \mathcal{F}$ after n iterations.

Outline

Approximate Policy Iteration: the Idea

Let $\mathcal{A}$ be an *approximation operator*.

- *Policy evaluation:* given the current policy π_k, compute $V_k = \mathcal{A}V^{\pi_k}$
- *Policy improvement:* given the *approximated* value of the current policy, compute the greedy policy w.r.t. V_k as

$$\pi_{k+1}(x) \in \arg\max_{a \in A} \Big[r(x,a) + \gamma \sum_{y \in X} p(y|x,a) V_k(y) \Big].$$

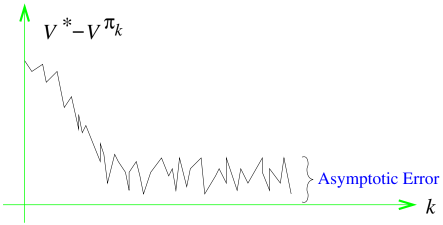
Approximate Policy Iteration: the Idea

Let $\mathcal{A}$ be an *approximation operator*.

- *Policy evaluation:* given the current policy π_k, compute $V_k = \mathcal{A}V^{\pi_k}$
- *Policy improvement:* given the *approximated* value of the current policy, compute the greedy policy w.r.t. V_k as

$$\pi_{k+1}(x) \in \arg\max_{a \in A} \Big[r(x,a) + \gamma \sum_{y \in X} p(y|x,a) V_k(y) \Big].$$

Problem: the algorithm is no longer guaranteed to converge.



Approximate Policy Iteration: performance loss

Proposition

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

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

Outline

Linear TD(λ): the algorithm

Algorithm Definition

Given a *linear space* $\mathcal{F} = \{V_\alpha(x) = \sum_{i=1}^d \alpha_i \phi_i(x), \alpha \in \mathbb{R}^d\}$.
Trace vector $z \in \mathbb{R}^d$ and *parameter* vector $\alpha \in \mathbb{R}^d$ initialized to zero.
Generate a sequence of states $(x_0, x_1, x_2, \dots)$ according to π.
At each step t, the *temporal difference* is

$$d_t = r(x_t, \pi(x_t)) + \gamma V_{\alpha_t}(x_{t+1}) - V_{\alpha_t}(x_t)$$

and the parameters are updated as

$$\begin{aligned} \alpha_{t+1} &= \alpha_t + \eta_t d_t z_t, \\ z_{t+1} &= \lambda \gamma z_t + \phi(x_{t+1}), \end{aligned}$$

where η_t is learning step.

Linear TD(λ): approximation error

Proposition (Tsitsiklis et Van Roy, 1996)

Let the learning rate η_t satisfy

$$\sum_{t\geq 0} \eta_t = \infty, \text{ and } \sum_{t \geq 0} \eta_t^2 < \infty.$$

We assume that π admits a *stationary distribution* μ_π and that the features $(\phi_i)_{1\leq k \leq K}$ are *linearly independent*. There exists a fixed α^* such that

$$\lim_{t\to\infty} \alpha_t = \alpha^*.$$

Furthermore we obtain

$$\underbrace{\|V_{\alpha^*} - V^\pi\|_{2,\mu^\pi}}_{\text{approximation error}} \leq \frac{1-\lambda\gamma}{1-\gamma} \underbrace{\inf_{\alpha} \|V_\alpha - V^\pi\|_{2,\mu^\pi}}_{\text{smallest approximation error}}.$$

Linear TD(λ): approximation error

Remark: for $\lambda = 1$, we recover Monte-Carlo (or TD(1)) and the bound is the smallest!

Linear TD(λ): approximation error

Remark: for $\lambda = 1$, we recover Monte-Carlo (or TD(1)) and the bound is the smallest!

Problem: the bound does not consider the variance (i.e., samples needed for α_t to converge to α^*).

Linear TD(λ): implementation

- **Pros**: simple to implement, computational cost *linear* in d.
- **Cons**: very sample *inefficient*, many samples are needed to converge.

Outline

Performance Loss

Approximate Value Iteration

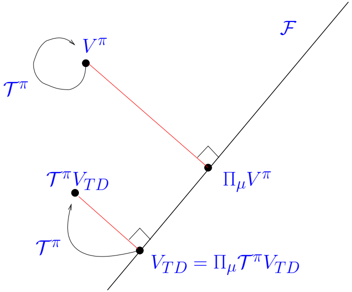
Approximate Policy Iteration

- Linear Temporal-Difference
- Least-Squares Temporal Difference
- Bellman Residual Minimization

Least-squares TD: the algorithm

Recall: $V^\pi = \mathcal{T}^\pi V^\pi$.

Intuition: compute $V = \mathcal{A}\mathcal{T}^\pi V$.



Focus on the $L_{2,\mu}$-weighted norm and projection Π_μ

$$\Pi_\mu g = \arg\min_{f \in \mathcal{F}} \|f - g\|_\mu.$$

Least-squares TD: the algorithm

By construction, the Bellman residual of V_{TD} is orthogonal to $\mathcal{F}$, thus for any $1 \leq i \leq d$

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

Least-squares TD: the algorithm

By construction, the Bellman residual of V_{TD} is orthogonal to $\mathcal{F}$, thus for any $1 \leq i \leq d$

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

and

$$\begin{aligned} \langle r^\pi + \gamma P^\pi V_{TD} - V_{TD}, \phi_i \rangle_\mu &= 0 \\ \langle r^\pi, \phi_i \rangle_\mu + \sum_{j=1}^{d} \langle \gamma P^\pi \phi_j - \phi_j, \phi_i \rangle_\mu \alpha_{TD,j} &= 0, \end{aligned}$$

Least-squares TD: the algorithm

By construction, the Bellman residual of V_{TD} is orthogonal to $\mathcal{F}$, thus for any $1 \leq i \leq d$

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

and

$$\begin{aligned} \langle r^\pi + \gamma P^\pi V_{TD} - V_{TD}, \phi_i \rangle_\mu &= 0 \\ \langle r^\pi, \phi_i \rangle_\mu + \sum_{j=1}^{d} \langle \gamma P^\pi \phi_j - \phi_j, \phi_i \rangle_\mu \alpha_{TD,j} &= 0, \end{aligned}$$

$\Rightarrow \alpha_{TD}$ is the solution of a *linear system* of order d.

Least-squares TD: the algorithm

Algorithm Definition

The LSTD solution α_{TD} can be computed by computing the matrix A and vector b defined as

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

and then solving the system $A\alpha = b$.

Least-squares TD: the approximation error

Problem: in general $\Pi_\mu \mathcal{T}^\pi$ does not admit a fixed point (i.e., matrix A is not invertible).

Least-squares TD: the approximation error

Problem: in general $\Pi_\mu \mathcal{T}^\pi$ does not admit a fixed point (i.e., matrix A is not invertible).

Solution: use the stationary distribution μ_π of policy π, that is

$$\mu_\pi P^\pi = \mu_\pi, \text{ and } \mu_\pi(y) = \sum_x p(y|x,\pi(x))\mu_\pi(x)$$

Least-squares TD: the approximation error

Proposition

The Bellman operator $\mathcal{T}^\pi$ is a *contraction* in the weighted L_{2,μ_π}-norm. Thus the joint operator $\Pi_{\mu_\pi}\mathcal{T}^\pi$ is a contraction and it admits a unique *fixed point V_{TD}*. Then

$$\underbrace{\|V^\pi - V_{TD}\|_{\mu_\pi}}_{\text{approximation error}} \leq \frac{1}{\sqrt{1-\gamma^2}} \underbrace{\inf_{V\in\mathcal{F}} \|V^\pi - V\|_{\mu_\pi}}_{\text{smallest approximation error}} .$$

Least-squares TD: the implementation

- Generate $(X_0, X_1, \dots)$ from *direct execution* of π and observes $R_t = r(X_t, \pi(X_t))$

Least-squares TD: the implementation

- Generate $(X_0, X_1, \ldots)$ from *direct execution* of π and observes $R_t = r(X_t, \pi(X_t))$
- Compute estimates

$$\hat{A}_{ij} = \frac{1}{n}\sum_{t=1}^{n} \phi_i(X_t)[\phi_j(X_t) - \gamma\phi_j(X_{t+1})],$$

$$\hat{b}_i = \frac{1}{n}\sum_{t=1}^{n} \phi_i(X_t)R_t.$$

Least-squares TD: the implementation

- Generate $(X_0, X_1, \dots)$ from *direct execution* of π and observes $R_t = r(X_t, \pi(X_t))$
- Compute estimates

$$\hat{A}_{ij} = \frac{1}{n}\sum_{t=1}^{n} \phi_i(X_t)[\phi_j(X_t) - \gamma\phi_j(X_{t+1})],$$

$$\hat{b}_i = \frac{1}{n}\sum_{t=1}^{n} \phi_i(X_t)R_t.$$

- Solve $\hat{A}\alpha = \hat{b}$

Least-squares TD: the implementation

- Generate $(X_0, X_1, \dots)$ from *direct execution* of π and observes $R_t = r(X_t, \pi(X_t))$
- Compute estimates

$$\begin{aligned}\hat{A}_{ij} &= \frac{1}{n}\sum_{t=1}^{n}\phi_i(X_t)[\phi_j(X_t) - \gamma\phi_j(X_{t+1})], \\ \hat{b}_i &= \frac{1}{n}\sum_{t=1}^{n}\phi_i(X_t)R_t.\end{aligned}$$

- Solve $\hat{A}\alpha = \hat{b}$

Remark:

- No need for a generative model.
- If the chain is ergodic, $\hat{A} \to A$ et $\hat{b} \to b$ when $n \to \infty$.

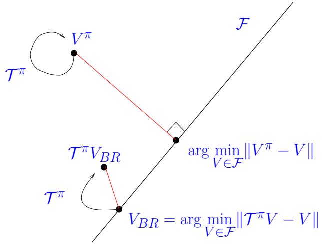
Outline

Performance Loss

Approximate Value Iteration

Approximate Policy Iteration
- Linear Temporal-Difference
- Least-Squares Temporal Difference
- Bellman Residual Minimization

Bellman Residual Minimization (BRM): the idea



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

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

Bellman Residual Minimization (BRM): the idea

The mapping $\alpha \rightarrow \mathcal{T}^\pi V_\alpha - V_\alpha$ is affine
The function $\alpha \rightarrow \|\mathcal{T}^\pi V_\alpha - V_\alpha\|_\mu^2$ 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 $\{\psi_i = \phi_i - \gamma P^\pi \phi_i\}_{i=1\ldots d}$, 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

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

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

BRM: the implementation

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

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

$$\widehat{b}_i = \frac{1}{n}\sum_{t=1}^{n}\big[\phi_i(X_t) - \gamma\frac{\phi_i(Y_t) + \phi_i(Y'_t)}{2}\big] R_t.$$

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.

Bibliography I

Reinforcement Learning

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