Markov Decision Processes and Dynamic Programming

Alessandro LAZARIC (*Facebook AI Research* / on leave *Inria Lille*)

ENS Cachan - Master 2 MVA

FAIR / Inria

How to *model* an RL problem

The Markov Decision Process

How to *model* an RL problem

The Markov Decision Process

Tools

Model

Value Functions

How to *model* an RL problem

The Markov Decision Process

Tools

Model

Value Functions

Probability Theory

Definition (Conditional probability)

Given two events A and B with $\mathbb{P}(B) > 0$, the ***conditional probability*** *of A given B is*

$$\mathbb{P}(A|B) = \frac{\mathbb{P}(A \cap B)}{\mathbb{P}(B)}.$$

Probability Theory

Definition (Conditional probability)

Given two events A and B with $\mathbb{P}(B) > 0$, the ***conditional probability*** *of A given B is*

$$\mathbb{P}(A|B) = \frac{\mathbb{P}(A \cap B)}{\mathbb{P}(B)}.$$

Similarly, if X and Y are non-degenerate and jointly continuous random variables with density $f_{X,Y}(x,y)$ then if B has positive measure then the ***conditional probability*** *is*

$$\mathbb{P}(X \in A | Y \in B) = \frac{\int_{y \in B} \int_{x \in A} f_{X,Y}(x,y) dx dy}{\int_{y \in B} \int_{x} f_{X,Y}(x,y) dx dy}.$$

Probability Theory

Definition (Law of total expectation)

Given a function f and two random variables X, Y we have that

$$\mathbb{E}_{X,Y}\big[f(X,Y)\big] = \mathbb{E}_X\Big[\mathbb{E}_Y\big[f(x,Y)|X=x\big]\Big].$$

Norms and Contractions

Definition

*Given a vector space $\mathcal{V} \subseteq \mathbb{R}^d$ a function $f : \mathcal{V} \to \mathbb{R}_0^+$ is a **norm** if an only if*

- *If $f(v) = 0$ for some $v \in \mathcal{V}$, then $v = 0$.*
- *For any $\lambda \in \mathbb{R}, v \in \mathcal{V}$, $f(\lambda v) = |\lambda| f(v)$.*
- *Triangle inequality: For any $v, u \in \mathcal{V}$, $f(v + u) \leq f(v) + f(u)$.*

Norms and Contractions

- L_p-norm

$$||v||_p = \left(\sum_{i=1}^{d} |v_i|^p \right)^{1/p}.$$

Norms and Contractions

- L_p-norm

$$||v||_p = \left(\sum_{i=1}^{d} |v_i|^p \right)^{1/p}.$$

- L_∞-norm

$$||v||_\infty = \max_{1 \leq i \leq d} |v_i|.$$

Norms and Contractions

- L_p-norm

$$||v||_p = \left(\sum_{i=1}^{d} |v_i|^{p} \right)^{1/p}.$$

- L_∞-norm

$$||v||_\infty = \max_{1 \leq i \leq d} |v_i|.$$

- $L_{\mu,p}$-norm

$$||v||_{\mu,p} = \left(\sum_{i=1}^{d} \frac{|v_i|^p}{\mu_i} \right)^{1/p}.$$

Norms and Contractions

- L_p-norm

$$||v||_p = \left(\sum_{i=1}^{d} |v_i|^p \right)^{1/p}.$$

- L_∞-norm

$$||v||_\infty = \max_{1 \leq i \leq d} |v_i|.$$

- $L_{\mu,p}$-norm

$$||v||_{\mu,p} = \left(\sum_{i=1}^{d} \frac{|v_i|^p}{\mu_i} \right)^{1/p}.$$

- $L_{\mu,\infty}$-norm

$$||v||_{\mu,\infty} = \max_{1 \leq i \leq d} \frac{|v_i|}{\mu_i}.$$

Norms and Contractions

- L_p-norm

$$||v||_p = \left(\sum_{i=1}^{d} |v_i|^p \right)^{1/p}.$$

- L_∞-norm

$$||v||_\infty = \max_{1 \leq i \leq d} |v_i|.$$

- $L_{\mu,p}$-norm

$$||v||_{\mu,p} = \left(\sum_{i=1}^{d} \frac{|v_i|^p}{\mu_i} \right)^{1/p}.$$

- $L_{\mu,\infty}$-norm

$$||v||_{\mu,\infty} = \max_{1 \leq i \leq d} \frac{|v_i|}{\mu_i}.$$

- $L_{2,P}$-matrix norm (P is a positive definite matrix)

$$||v||_P^2 = v^\top P v.$$

Norms and Contractions

Definition

A sequence of vectors $v_n \in \mathcal{V}$ (with $n \in \mathbb{N}$) is said to converge in norm $||\cdot||$ to $v \in \mathcal{V}$ if

$$\lim_{n\to\infty} ||v_n - v|| = 0.$$

Norms and Contractions

Definition

A sequence of vectors $v_n \in \mathcal{V}$ (with $n \in \mathbb{N}$) is said to converge in norm $|| \cdot ||$ to $v \in \mathcal{V}$ if

$$\lim_{n \to \infty} ||v_n - v|| = 0.$$

Definition

A sequence of vectors $v_n \in \mathcal{V}$ (with $n \in \mathbb{N}$) is a Cauchy sequence if

$$\lim_{n \to \infty} \sup_{m \geq n} ||v_n - v_m|| = 0.$$

Norms and Contractions

Definition

A sequence of vectors $v_n \in \mathcal{V}$ (with $n \in \mathbb{N}$) is said to converge in norm $||\cdot||$ to $v \in \mathcal{V}$ if

$$\lim_{n\to\infty} ||v_n - v|| = 0.$$

Definition

A sequence of vectors $v_n \in \mathcal{V}$ (with $n \in \mathbb{N}$) is a Cauchy sequence if

$$\lim_{n\to\infty} \sup_{m\geq n} ||v_n - v_m|| = 0.$$

Definition

A vector space $\mathcal{V}$ equipped with a norm $||\cdot||$ is complete if every Cauchy sequence in $\mathcal{V}$ is convergent in the norm of the space.

Norms and Contractions

Definition

An operator $\mathcal{T} : \mathcal{V} \to \mathcal{V}$ *is L-Lipschitz if for any* $v, u \in \mathcal{V}$

$$||\mathcal{T}v - \mathcal{T}u|| \leq L||u - v||.$$

If $L \leq 1$ *then* $\mathcal{T}$ *is a non-expansion, while if* $L < 1$ *then* $\mathcal{T}$ *is a L-contraction.*
If $\mathcal{T}$ *is Lipschitz then it is also continuous, that is*

$$\text{if } v_n \rightarrow_{||\cdot||} v \quad \text{then} \quad \mathcal{T}v_n \rightarrow_{||\cdot||} \mathcal{T}v.$$

Norms and Contractions

Definition

An operator $\mathcal{T} : \mathcal{V} \rightarrow \mathcal{V}$ is L-Lipschitz if for any $v, u \in \mathcal{V}$

$$||\mathcal{T}v - \mathcal{T}u|| \leq L||u - v||.$$

If $L \leq 1$ then $\mathcal{T}$ is a non-expansion, while if $L < 1$ then $\mathcal{T}$ is a L-contraction.
If $\mathcal{T}$ is Lipschitz then it is also continuous, that is

$$\text{if } v_n \rightarrow_{||\cdot||} v \quad \text{then} \quad \mathcal{T}v_n \rightarrow_{||\cdot||} \mathcal{T}v.$$

Definition

A vector $v \in \mathcal{V}$ is a fixed point of the operator $\mathcal{T} : \mathcal{V} \rightarrow \mathcal{V}$ if $\mathcal{T}v = v$.

Norms and Contractions

Proposition (Banach Fixed Point Theorem)

Let $\mathcal{V}$ be a *complete* vector space equipped with the norm $||\cdot||$ and $\mathcal{T}:\mathcal{V}\to\mathcal{V}$ be a *γ-contraction* mapping. Then

1. $\mathcal{T}$ admits a *unique fixed point* v.
2. For any $v_0\in\mathcal{V}$, if $v_{n+1}=\mathcal{T}v_n$ then $v_n \to_{||\cdot||} v$ with a *geometric convergence rate*:

$$||v_n - v|| \leq \gamma^n ||v_0 - v||.$$

Linear Algebra

Given a square matrix $A \in \mathbb{R}^{N \times N}$:

- *Eigenvalues of a matrix (1).* $v \in \mathbb{R}^N$ and $\lambda \in \mathbb{R}$ are *eigenvector* and *eigenvalue* of A if

$$Av = \lambda v.$$

Linear Algebra

Given a square matrix $A \in \mathbb{R}^{N \times N}$:

- *Eigenvalues of a matrix (1).* $v \in \mathbb{R}^N$ and $\lambda \in \mathbb{R}$ are *eigenvector* and *eigenvalue* of A if

$$Av = \lambda v.$$

- *Eigenvalues of a matrix (2).* If A has eigenvalues $\{\lambda_i\}_{i=1}^N$, then $B = (I - \alpha A)$ has eigenvalues $\{\mu_i\}$

$$\mu_i = 1 - \alpha \lambda_i.$$

Linear Algebra

Given a square matrix $A \in \mathbb{R}^{N \times N}$:

- *Eigenvalues of a matrix (1).* $v \in \mathbb{R}^N$ and $\lambda \in \mathbb{R}$ are *eigenvector* and *eigenvalue* of A if

$$Av = \lambda v.$$

- *Eigenvalues of a matrix (2).* If A has eigenvalues $\{\lambda_i\}_{i=1}^N$, then $B = (I - \alpha A)$ has eigenvalues $\{\mu_i\}$

$$\mu_i = 1 - \alpha \lambda_i.$$

- *Matrix inversion.* A can be *inverted* if and only if $\forall i$, $\lambda_i \neq 0$.

Linear Algebra

- *Stochastic matrix.* A square matrix $P \in \mathbb{R}^{N \times N}$ is a stochastic matrix if
  1. all non-zero entries, $\forall i, j$, $[P]_{i,j} \geq 0$
  2. all the rows sum to one, $\forall i$, $\sum_{j=1}^{N} [P]_{i,j} = 1$.

  All the eigenvalues of a stochastic matrix are bounded by 1, i.e., $\forall i$, $\lambda_i \leq 1$.

How to *model* an RL problem

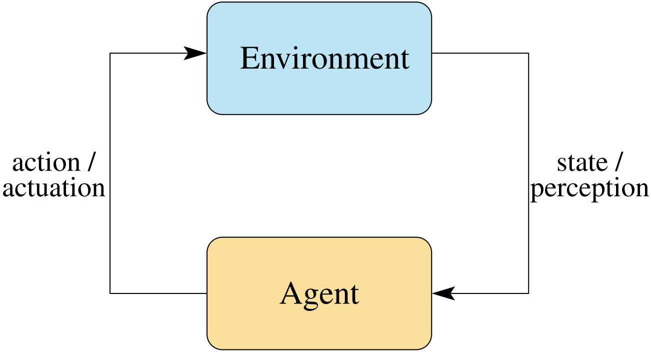
The Markov Decision Process

Tools

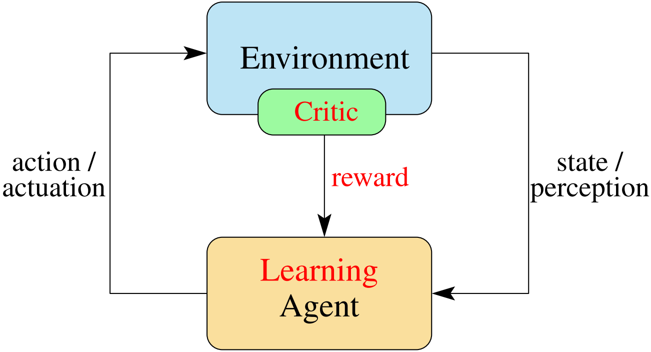
Model

Value Functions

The Reinforcement Learning Model



The Reinforcement Learning Model



The Reinforcement Learning Model

The environment

- *Controllability*: fully (e.g., chess) or partially (e.g., portfolio optimization)
- *Uncertainty*: deterministic (e.g., chess) or stochastic (e.g., backgammon)
- *Reactive*: adversarial (e.g., chess) or fixed (e.g., tetris)
- *Observability*: full (e.g., chess) or partial (e.g., robotics)
- *Availability*: known (e.g., chess) or unknown (e.g., robotics)

The Reinforcement Learning Model

The environment

- *Controllability*: fully (e.g., chess) or partially (e.g., portfolio optimization)
- *Uncertainty*: deterministic (e.g., chess) or stochastic (e.g., backgammon)
- *Reactive*: adversarial (e.g., chess) or fixed (e.g., tetris)
- *Observability*: full (e.g., chess) or partial (e.g., robotics)
- *Availability*: known (e.g., chess) or unknown (e.g., robotics)

The critic

- Sparse (e.g., win or loose) vs informative (e.g., closer or further)
- Preference reward
- Frequent or sporadic
- Known or unknown

The Reinforcement Learning Model

The environment

- *Controllability*: fully (e.g., chess) or partially (e.g., portfolio optimization)
- *Uncertainty*: deterministic (e.g., chess) or stochastic (e.g., backgammon)
- *Reactive*: adversarial (e.g., chess) or fixed (e.g., tetris)
- *Observability*: full (e.g., chess) or partial (e.g., robotics)
- *Availability*: known (e.g., chess) or unknown (e.g., robotics)

The critic

- Sparse (e.g., win or loose) vs informative (e.g., closer or further)
- Preference reward
- Frequent or sporadic
- Known or unknown

The agent

- Open loop control
- Close loop control (i.e., *adaptive*)
- Non-stationary close loop control (i.e., *learning*)

Markov Chains

Definition (Markov chain)

Let the state space X be a bounded compact subset of the Euclidean space, the discrete-time dynamic system $(x_t)_{t\in\mathbb{N}} \in X$ is a Markov chain if it satisfies the Markov property

$$\mathbb{P}(x_{t+1} = x \,|\, x_t, x_{t-1}, \ldots, x_0) = \mathbb{P}(x_{t+1} = x \,|\, x_t),$$

Given an initial state $x_0 \in X$, a Markov chain is defined by the transition probability p

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

Markov Decision Process

Definition (Markov decision process [1, 4, 3, 5, 2])

A ***Markov decision process*** *is defined as a tuple* $M = (X, A, p, r)$ *where*

Markov Decision Process

Definition (Markov decision process [1, 4, 3, 5, 2])

A ***Markov decision process*** *is defined as a tuple* $M = (X, A, p, r)$ *where*

- X *is the state space,*

Markov Decision Process

Definition (Markov decision process [1, 4, 3, 5, 2])

A ***Markov decision process*** *is defined as a tuple* $M = (X, A, p, r)$ *where*

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

Markov Decision Process

Definition (Markov decision process [1, 4, 3, 5, 2])

*A **Markov decision process** is defined as a tuple $M = (X, A, p, r)$ where*

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

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

Markov Decision Process

Definition (Markov decision process [1, 4, 3, 5, 2])

A ***Markov decision process*** *is defined as a tuple* $M = (X, A, p, r)$ *where*

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

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

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

Markov Decision Process: the Assumptions

Time assumption: time is discrete

$$t \rightarrow t + 1$$

Possible relaxations

- Identify the proper time granularity
- Most of MDP literature extends to continuous time

Markov Decision Process: the Assumptions

Markov assumption: the current state x and action a are a sufficient statistics for the next state y

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

Possible relaxations

- Define a new state $h_t = (x_t, x_{t-1}, x_{t-2}, \ldots)$
- Move to partially observable MDP (PO-MDP)
- Move to predictive state representation (PSR) model

Markov Decision Process: the Assumptions

Reward assumption: the reward is uniquely defined by a transition (or part of it)

$$r(x, a, y)$$

Possible relaxations

- Distinguish between global goal and reward function
- Move to inverse reinforcement learning (IRL) to induce the reward function from desired behaviors

Markov Decision Process: the Assumptions

Stationarity assumption: the dynamics and reward do not change over time

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

Possible relaxations

- Identify and remove the non-stationary components (e.g., cyclo-stationary dynamics)
- Identify the time-scale of the changes

Question

Is the MDP formalism powerful enough?

$\Rightarrow$ *Let's try!*

Example: the Retail Store Management Problem

Description. At each month t, a store contains x_t *items* of a specific goods and the demand for that goods is D_t. At the end of each month the manager of the store can *order* a_t more items from his supplier. Furthermore we know that

- The *cost* of maintaining an inventory of x is $h(x)$.
- The *cost* to order a items is $C(a)$.
- The *income* for selling q items is $f(q)$.
- If the demand D is bigger than the available inventory x, customers that cannot be served leave.
- The *value of the remaining inventory* at the end of the year is $g(x)$.
- *Constraint*: the store has a maximum capacity M.

Example: the Retail Store Management Problem

- *State space*: $x \in X = \{0, 1, \ldots, M\}$.

Example: the Retail Store Management Problem

- *State space*: $x \in X = \{0, 1, \dots, M\}$.
- *Action space*: it is not possible to order more items that the capacity of the store, then the action space should depend on the current state. Formally, at state$x$, $a \in A(x) = \{0, 1, \dots, M - x\}$.

Example: the Retail Store Management Problem

- *State space*: $x \in X = \{0, 1, \dots, M\}$.
- *Action space*: it is not possible to order more items that the capacity of the store, then the action space should depend on the current state. Formally, at state$x$, $a \in A(x) = \{0, 1, \dots, M - x\}$.
- *Dynamics*: $x_{t+1} = [x_t + a_t - D_t]^+$.
  Problem: the dynamics should be Markov and stationary!

Example: the Retail Store Management Problem

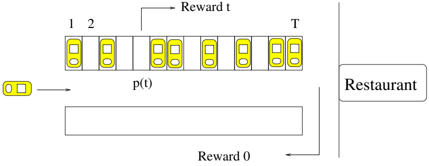
- *State space*: $x \in X = \{0, 1, \dots, M\}$.
- *Action space*: it is not possible to order more items that the capacity of the store, then the action space should depend on the current state. Formally, at state$x$, $a \in A(x) = \{0, 1, \dots, M - x\}$.
- *Dynamics*: $x_{t+1} = [x_t + a_t - D_t]^+$.
  Problem: the dynamics should be Markov and stationary!
- The demand D_t is *stochastic and time-independent*. Formally, $D_t \overset{i.i.d.}{\sim} \mathcal{D}$.

Example: the Retail Store Management Problem

- *State space*: $x \in X = \{0, 1, \dots, M\}$.
- *Action space*: it is not possible to order more items that the capacity of the store, then the action space should depend on the current state. Formally, at state$x$, $a \in A(x) = \{0, 1, \dots, M - x\}$.
- *Dynamics*: $x_{t+1} = [x_t + a_t - D_t]^+$.
  Problem: the dynamics should be Markov and stationary!
- The demand D_t is *stochastic and time-independent*. Formally, $D_t \overset{i.i.d.}{\sim} \mathcal{D}$.
- *Reward*: $r_t = -C(a_t) - h(x_t + a_t) + f([x_t + a_t - x_{t+1}]^+)$.

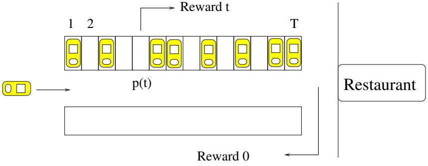
Exercise: the Parking Problem

A driver wants to park his car as close as possible to the restaurant.



Exercise: the Parking Problem

A driver wants to park his car as close as possible to the restaurant.



- The driver cannot see whether a place is available unless he is in front of it.
- There are P places.
- At each place i the driver can either move to the next place or park (if the place is available).
- The closer to the restaurant the parking, the higher the satisfaction.
- If the driver doesn't park anywhere, then he/she leaves the restaurant and has to find another one.

Policy

Definition (Policy)

A decision rule π_t *can be*

- *Deterministic:* $\pi_t : X \rightarrow A$,
- *Stochastic:* $\pi_t : X \rightarrow \Delta(A)$,

Policy

Definition (Policy)

A decision rule π_t *can be*

- *Deterministic:* $\pi_t : X \rightarrow A$,
- *Stochastic:* $\pi_t : X \rightarrow \Delta(A)$,

A policy (strategy, plan) can be

- *Non-stationary:* $\pi = (\pi_0, \pi_1, \pi_2, \dots)$,
- *Stationary (Markovian):* $\pi = (\pi, \pi, \pi, \dots)$.

Policy

Definition (Policy)

A decision rule π_t *can be*

- *Deterministic:* $\pi_t : X \rightarrow A$,
- *Stochastic:* $\pi_t : X \rightarrow \Delta(A)$,

A policy (strategy, plan) can be

- *Non-stationary:* $\pi = (\pi_0, \pi_1, \pi_2, \dots)$,
- *Stationary (Markovian):* $\pi = (\pi, \pi, \pi, \dots)$.

Remark: MDP M + stationary policy $\pi \Rightarrow$ *Markov chain* of state X and transition probability $p(y|x) = p(y|x, \pi(x))$.

Example: the Retail Store Management Problem

- Stationary policy 1

$$\pi(x) = \begin{cases} M - x & \text{if } x < M/4 \\ 0 & \text{otherwise} \end{cases}$$

- Stationary policy 2

$$\pi(x) = \max\{(M - x)/2 - x; 0\}$$

- Non-stationary policy

$$\pi_t(x) = \begin{cases} M - x & \text{if } t < 6 \\ \lfloor (M - x)/5 \rfloor & \text{otherwise} \end{cases}$$

How to *model* an RL problem

The Markov Decision Process

The Model

Value Functions

Question

How do we evaluate a policy and compare two policies?

$\Rightarrow$ *Value function!*

Optimization over Time Horizon

- *Finite time horizon T*: deadline at time T, the agent focuses on the sum of the rewards up to T.

Optimization over Time Horizon

- *Finite time horizon T*: deadline at time T, the agent focuses on the sum of the rewards up to T.
- *Infinite time horizon with discount*: the problem never terminates but rewards which are *closer* in time receive a *higher* importance.

Optimization over Time Horizon

- *Finite time horizon T*: deadline at time T, the agent focuses on the sum of the rewards up to T.
- *Infinite time horizon with discount*: the problem never terminates but rewards which are *closer* in time receive a *higher* importance.
- *Infinite time horizon with terminal state*: the problem never terminates but the agent will eventually reach a *termination state*.

Optimization over Time Horizon

- *Finite time horizon T*: deadline at time T, the agent focuses on the sum of the rewards up to T.
- *Infinite time horizon with discount*: the problem never terminates but rewards which are *closer* in time receive a *higher* importance.
- *Infinite time horizon with terminal state*: the problem never terminates but the agent will eventually reach a *termination state*.
- *Infinite time horizon with average reward*: the problem never terminates but the agent only focuses on the (expected) *average of the rewards*.

State Value Function

- *Finite time horizon T*: deadline at time T, the agent focuses on the sum of the rewards up to T.

$$V^\pi(t,x) = \mathbb{E}\Big[\sum_{s=t}^{T-1} r(x_s, \pi_s(x_s)) + R(x_T) \big| \, x_t = x; \pi\Big],$$

where R is a value function for the final state.

State Value Function

- *Finite time horizon T*: deadline at time T, the agent focuses on the sum of the rewards up to T.

$$V^{\pi}(t,x) = \mathbb{E}\Big[\sum_{s=t}^{T-1} r(x_s, \pi_s(x_s)) + R(x_T) \big| \, x_t = x; \pi\Big],$$

  where R is a value function for the final state.
- *Used when:* there is an intrinsic deadline to meet.

State Value Function

- *Infinite time horizon with discount*: the problem never terminates but rewards which are *closer* in time receive a *higher* importance.

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

with discount factor $0 \leq \gamma < 1$:

  - *small* $=$ short-term rewards, *big* $=$ long-term rewards
  - for any $\gamma \in [0, 1)$ the series always converge (for bounded rewards)

State Value Function

- *Infinite time horizon with discount*: the problem never terminates but rewards which are *closer* in time receive a *higher* importance.

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

  with discount factor $0 \leq \gamma < 1$:
  - *small* = short-term rewards, *big* = long-term rewards
  - for any $\gamma \in [0, 1)$ the series always converge (for bounded rewards)
- *Used when:* there is uncertainty about the deadline and/or an intrinsic definition of discount.

State Value Function

- *Infinite time horizon with terminal state*: the problem never terminates but the agent will eventually reach a *termination state*.

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

where T is the first (*random*) time when the *termination state* is achieved.

State Value Function

- *Infinite time horizon with terminal state*: the problem never terminates but the agent will eventually reach a *termination state*.

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

  where T is the first (*random*) time when the *termination state* is achieved.

- *Used when:* there is a known goal or a failure condition.

State Value Function

- *Infinite time horizon with average reward*: the problem never terminates but the agent only focuses on the (expected) *average of the rewards*.

$$V^\pi(x) = \lim_{T \to \infty} \mathbb{E}\Big[\frac{1}{T} \sum_{t=0}^{T-1} r(x_t, \pi(x_t)) \,|\, x_0 = x; \pi \Big].$$

State Value Function

- *Infinite time horizon with average reward*: the problem never terminates but the agent only focuses on the (expected) *average of the rewards*.

$$V^\pi(x) = \lim_{T\to\infty} \mathbb{E}\Big[\frac{1}{T}\sum_{t=0}^{T-1} r(x_t, \pi(x_t)) \,|\, x_0 = x; \pi\Big].$$

- *Used when:* the system should be constantly controlled over time.

State Value Function

Technical note: the expectations refer to all possible stochastic trajectories.

State Value Function

Technical note: the expectations refer to all possible stochastic trajectories.
A non-stationary policy π applied from state x_0 returns

$$(x_0, r_0, x_1, r_1, x_2, r_2, \ldots)$$

where $r_t = r(x_t, \pi_t(x_t))$ and $x_t \sim p(\cdot|x_{t-1}, a_t = \pi(x_t))$ are *random* realizations.

The value function (discounted infinite horizon) is

$$V^\pi(x) = \mathbb{E}_{(x_1,x_2,\ldots)}\Big[\sum_{t=0}^{\infty} \gamma^t r(x_t, \pi(x_t)) \,|\, x_0 = x; \pi\Big],$$

Example: the Retail Store Management Problem

Simulation

Optimal Value Function

Definition (Optimal policy and optimal value function)

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

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

in all the states $x \in X$, *where* Π *is some policy set of interest.*

Optimal Value Function

Definition (Optimal policy and optimal value function)

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

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

in all the states $x \in X$, where Π is some policy set of interest.

The corresponding value function is the optimal value function

$$V^* = V^{\pi^*}$$

Optimal Value Function

Remarks

1. $\pi^* \in \arg\max(\cdot)$ and not $\pi^* = \arg\max(\cdot)$ because an MDP may admit ***more than one*** optimal policy
2. π^* achieves the largest possible value function in ***every*** state
3. there always exists an optimal ***deterministic*** policy
4. except for problems with a finite horizon, there always exists an optimal ***stationary*** policy

Summary

1. MDP is a powerful model for interaction between an agent and a stochastic environment
2. The value function defines the objective to optimize

Limitations

1. All the previous value functions define an objective ***in expectation***

2. Other ***utility functions*** may be used

3. Risk measures could be integrated but they may induce "weird" problems and make the solution more difficult

How to solve *exactly* an MDP

Dynamic Programming

How to solve *exactly* an MDP

Dynamic Programming

Bellman Equations

Value Iteration

Policy Iteration

Notice

From now on we mostly work on the *discounted infinite horizon* setting.

Most results smoothly extend to other settings.

The Optimization Problem

$$\max_{\pi} \quad V^{\pi}(x_0) =$$

$$\max_{\pi} \quad \mathbb{E}\Big[r(x_0, \pi(x_0)) + \gamma r(x_1, \pi(x_1)) + \gamma^2 r(x_2, \pi(x_2)) + \dots\Big]$$

$$\Downarrow$$

very challenging (we should try as many as $|A|^{|S|}$ policies!)

The Optimization Problem

$$\max_{\pi} \; V^{\pi}(x_0) =$$

$$\max_{\pi} \; \mathbb{E}\big[r(x_0, \pi(x_0)) + \gamma r(x_1, \pi(x_1)) + \gamma^2 r(x_2, \pi(x_2)) + \ldots\big]$$

$$\Downarrow$$

very challenging (we should try as many as $|A|^{|S|}$ policies!)

$$\Downarrow$$

we need to leverage the *structure* of the MDP
to *simplify* the optimization problem

How to solve *exactly* an MDP

Dynamic Programming

Bellman Equations

Value Iteration

Policy Iteration

The Bellman Equation

Proposition

For any stationary policy $\pi = (\pi, \pi, \dots)$, the state value function at a state $x \in X$ satisfies the *Bellman equation*:

$$V^\pi(x) = r(x, \pi(x)) + \gamma \sum_y p(y|x, \pi(x)) V^\pi(y).$$

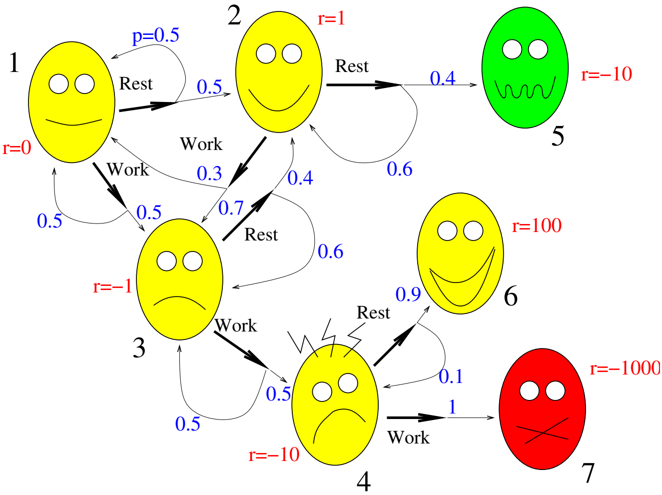
The Bellman Equation

Proof.
For any policy π,

$$
\begin{aligned}
V^\pi(x) &= \mathbb{E}\Big[\sum_{t\geq 0}\gamma^t r(x_t,\pi(x_t))\,|\,x_0 = x;\pi\Big] \\
&= r(x,\pi(x)) + \mathbb{E}\Big[\sum_{t\geq 1}\gamma^t r(x_t,\pi(x_t))\,|\,x_0 = x;\pi\Big] \\
&= r(x,\pi(x)) \\
&\qquad + \gamma\sum_y \mathbb{P}(x_1 = y\,|\,x_0 = x;\pi(x_0))\mathbb{E}\Big[\sum_{t\geq 1}\gamma^{t-1} r(x_t,\pi(x_t))\,|\,x_1 = y;\pi\Big] \\
&= r(x,\pi(x)) + \gamma\sum_y p(y|x,\pi(x))V^\pi(y).
\end{aligned}
$$

∎

Example: the student dilemma

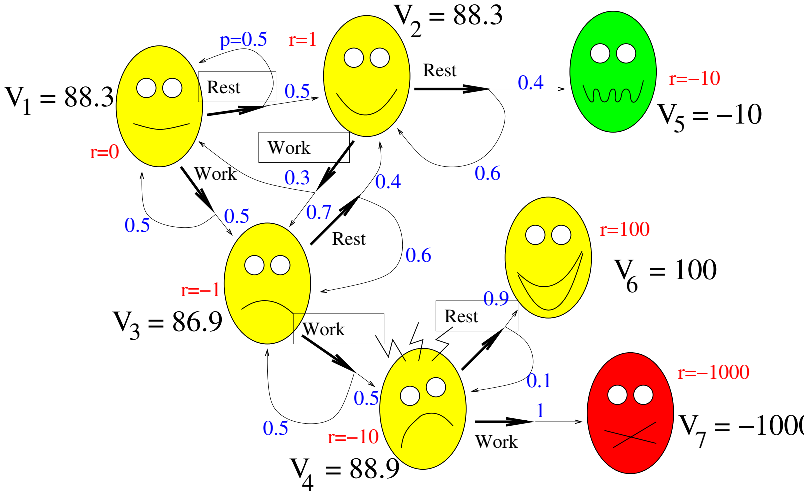


Example: the student dilemma

- *Model*: all the transitions are Markov, states x_5, x_6, x_7 are terminal.
- *Setting*: infinite horizon with terminal states.
- *Objective*: find the policy that maximizes the expected sum of rewards before achieving a terminal state.

Notice: not a discounted infinite horizon setting! But the Bellman equations hold unchanged.

Example: the student dilemma



Example: the student dilemma

Computing V_4:

$$V_6 = 100$$
$$V_4 = -10 + (0.9V_6 + 0.1V_4)$$

$$\Rightarrow V_4 = \frac{-10 + 0.9V_6}{0.9} = 88.8$$

Example: the student dilemma

Computing V_3: *no need* to consider all possible trajectories

$$
\begin{aligned}
V_4 &= 88.8 \\
V_3 &= -1 + (0.5 V_4 + 0.5 V_3)
\end{aligned}
$$

$$
\Rightarrow V_3 = \frac{-1 + 0.5 V_4}{0.5} = 86.8
$$

Example: the student dilemma

Computing V_3: *no need* to consider all possible trajectories

$$V_4 = 88.8$$
$$V_3 = -1 + (0.5V_4 + 0.5V_3)$$

$$\Rightarrow V_3 = \frac{-1 + 0.5V_4}{0.5} = 86.8$$

and so on for the rest...

The Optimal Bellman Equation

Bellman's Principle of Optimality [1]:

> *"An optimal policy has the property that, whatever the initial state and the initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision."*

The Optimal Bellman Equation

Proposition

The optimal value function V^* (i.e., $V^* = \max_\pi V^\pi$) is the solution to the *optimal Bellman equation*:

$$V^*(x) = \max_{a \in A} \big[r(x,a) + \gamma \sum_y p(y|x,a) V^*(y) \big].$$

and the optimal policy is

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

The Optimal Bellman Equation

Proof.
For any policy $\pi = (a, \pi')$ (possibly non-stationary),

$$
\begin{aligned}
V^*(x) &\overset{(a)}{=} \max_{\pi} \mathbb{E}\Big[\sum_{t\geq 0} \gamma^t r(x_t, \pi(x_t)) \,|\, x_0 = x; \pi\Big] \\
&\overset{(b)}{=} \max_{(a,\pi')} \Big[r(x,a) + \gamma \sum_y p(y|x,a) V^{\pi'}(y)\Big] \\
&\overset{(c)}{=} \max_{a} \Big[r(x,a) + \gamma \sum_y p(y|x,a) \max_{\pi'} V^{\pi'}(y)\Big] \\
&\overset{(d)}{=} \max_{a} \Big[r(x,a) + \gamma \sum_y p(y|x,a) V^{*}(y)\Big].
\end{aligned}
$$

■

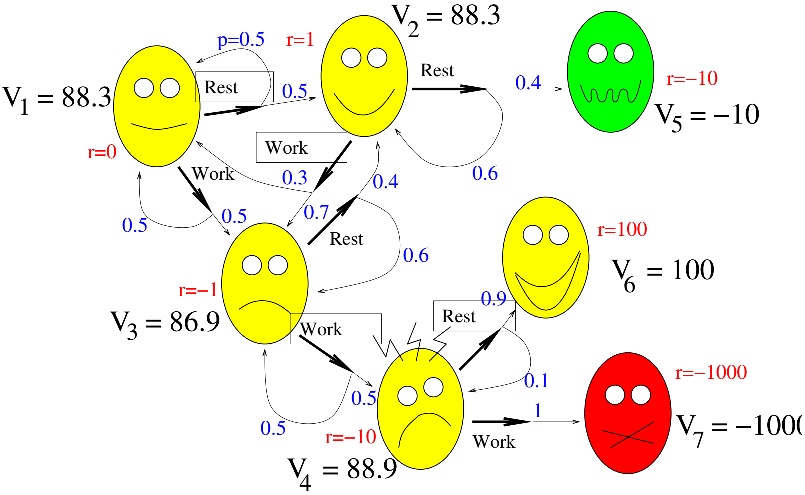
System of Equations

The Bellman equation

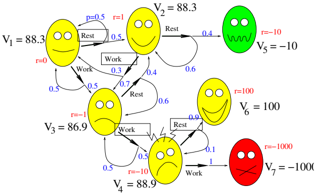
$$V^\pi(x) = r(x, \pi(x)) + \gamma \sum_y p(y|x, \pi(x)) V^\pi(y).$$

is a ***linear*** system of equations with N unknowns and N linear constraints.

Example: the student dilemma



Example: the student dilemma



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

System of equations

$$\begin{cases} V_1 &= 0 + 0.5V_1 + 0.5V_2 \\ V_2 &= 1 + 0.3V_1 + 0.7V_3 \\ V_3 &= -1 + 0.5V_4 + 0.5V_3 \\ V_4 &= -10 + 0.9V_6 + 0.1V_4 \\ V_5 &= -10 \\ V_6 &= 100 \\ V_7 &= -1000 \end{cases} \Rightarrow$$

$$(V, R \in \mathbb{R}^7,\ P \in \mathbb{R}^{7\times 7})$$

$$V = R + PV$$

$$\Downarrow$$

$$V = (I - P)^{-1} R$$

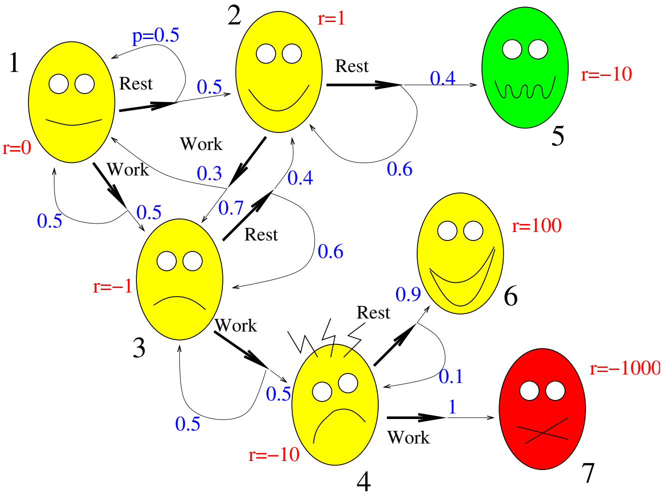
System of Equations

The optimal Bellman equation

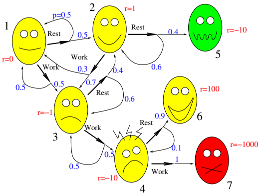
$$V^*(x) = \max_{a \in A} \Big[r(x,a) + \gamma \sum_y p(y|x,a) V^*(y) \Big].$$

is a (highly) ***non-linear*** system of equations with N unknowns and N non-linear constraints (i.e., the max operator).

Example: the student dilemma



Example: the student dilemma



$$V^*(x) = \max_{a \in A}\Big[r(x,a) + \gamma \sum_y p(y|x,a) V^*(y)\Big]$$

System of equations

$$\begin{cases} V_1 & = \max\big\{0 + 0.5V_1 + 0.5V_2;\ \ 0 + 0.5V_1 + 0.5V_3\big\} \\ V_2 & = \max\big\{1 + 0.4V_5 + 0.6V_2;\ \ 1 + 0.3V_1 + 0.7V_3\big\} \\ V_3 & = \max\big\{-1 + 0.4V_2 + 0.6V_3;\ \ -1 + 0.5V_4 + 0.5V_3\big\} \\ V_4 & = \max\big\{-10 + 0.9V_6 + 0.1V_4;\ \ -10 + V_7\big\} \\ V_5 & = -10 \\ V_6 & = 100 \\ V_7 & = -1000 \end{cases}$$

$\Rightarrow$ too complicated, we need to find an alternative solution.

The Bellman Operators

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

Definition

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

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

and the optimal Bellman operator (or dynamic programming operator) is

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

The Bellman Operators

Proposition

Properties of the Bellman operators

1. *Monotonicity*: for any $W_1, W_2 \in \mathbb{R}^N$, if $W_1 \leq W_2$ component-wise, then

$$\begin{aligned} \mathcal{T}^\pi W_1 &\leq \mathcal{T}^\pi W_2, \\ \mathcal{T} W_1 &\leq \mathcal{T} W_2. \end{aligned}$$

The Bellman Operators

Proposition

Properties of the Bellman operators

1. *Monotonicity*: for any $W_1, W_2 \in \mathbb{R}^N$, if $W_1 \leq W_2$ component-wise, then

$$\begin{aligned} \mathcal{T}^\pi W_1 &\leq \mathcal{T}^\pi W_2, \\ \mathcal{T} W_1 &\leq \mathcal{T} W_2. \end{aligned}$$

2. *Offset*: for any scalar $c \in \mathbb{R}$,

$$\begin{aligned} \mathcal{T}^\pi(W + cI_N) &= \mathcal{T}^\pi W + \gamma c I_N, \\ \mathcal{T}(W + cI_N) &= \mathcal{T} W + \gamma c I_N, \end{aligned}$$

The Bellman Operators

Proposition

3. *Contraction in L_∞-norm*: for any $W_1, W_2 \in \mathbb{R}^N$

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

The Bellman Operators

Proposition

3. *Contraction in L_∞-norm*: for any $W_1, W_2 \in \mathbb{R}^N$

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

4. *Fixed point*: For any policy π

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

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

The Bellman Operators

Proposition

3. *Contraction in L_∞-norm*: for any $W_1, W_2 \in \mathbb{R}^N$

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

4. *Fixed point*: For any policy π

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

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

Furthermore for any $W \in \mathbb{R}^N$ and any stationary policy π

$$
\begin{aligned}
\lim_{k\to\infty} (\mathcal{T}^\pi)^k W &= V^\pi, \\
\lim_{k\to\infty} (\mathcal{T})^k W &= V^*.
\end{aligned}
$$

The Bellman Equation

Proof.
The contraction property (3) holds since for any $x \in X$ we have

$$
\begin{aligned}
&|\mathcal{T} W_1(x) - \mathcal{T} W_2(x)| \\
&= \Big| \max_a \Big[r(x,a) + \gamma \sum_y p(y|x,a) W_1(y) \Big] - \max_{a'} \Big[r(x,a') + \gamma \sum_y p(y|x,a') W_2(y) \Big] \Big| \\
&\overset{(a)}{\leq} \max_a \Big| \Big[r(x,a) + \gamma \sum_y p(y|x,a) W_1(y) \Big] - \Big[r(x,a) + \gamma \sum_y p(y|x,a) W_2(y) \Big] \Big| \\
&= \gamma \max_a \sum_y p(y|x,a) |W_1(y) - W_2(y)| \\
&\leq \gamma ||W_1 - W_2||_\infty \max_a \sum_y p(y|x,a) = \gamma ||W_1 - W_2||_\infty,
\end{aligned}
$$

where in (a) we used $\max_a f(a) - \max_{a'} g(a') \leq \max_a (f(a) - g(a))$. ■

Exercise: Fixed Point

Revise the Banach fixed point theorem and prove the fixed point property of the Bellman operator.

How to solve *exactly* an MDP

Dynamic Programming

Bellman Equations

Value Iteration

Policy Iteration

Question

How do we compute the value functions / solve an MDP?

$\Rightarrow$ *Value/Policy Iteration algorithms!*

System of Equations

The Bellman equation

$$V^\pi(x) = r(x, \pi(x)) + \gamma \sum_y p(y|x, \pi(x)) V^\pi(y).$$

is a *linear* system of equations with N unknowns and N linear constraints.

System of Equations

The Bellman equation

$$V^\pi(x) = r(x, \pi(x)) + \gamma \sum_y p(y|x, \pi(x)) V^\pi(y).$$

is a *linear* system of equations with N unknowns and N linear constraints.
The optimal Bellman equation

$$V^*(x) = \max_{a \in A} \big[r(x, a) + \gamma \sum_y p(y|x, a) V^*(y) \big].$$

is a (highly) ***non-linear*** system of equations with N unknowns and N non-linear constraints (i.e., the max operator).

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 $\mathcal{T}$

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

Value Iteration: the Guarantees

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

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

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

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

- *Convergence rate*. Let $\epsilon > 0$ and $||r||_\infty \leq r_{\max}$, then after *at most*

$$K = \frac{\log(r_{\max}/\epsilon)}{\log(1/\gamma)}$$

iterations $||V_K - V^*||_\infty \leq \epsilon$.

Value Iteration: the Complexity

Time complexity

- Each iteration and the computation of the greedy policy take $O(N^2|A|)$ operations.

$$V_{k+1}(x) = \mathcal{T} V_k(x) = \max_{a \in A}\Big[r(x,a) + \gamma \sum_y p(y|x,a) V_k(y)\Big]$$

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

- Total time complexity $O(KN^2|A|)$

Space complexity

- Storing the MDP: dynamics $O(N^2|A|)$ and reward $O(N|A|)$.
- Storing the value function and the optimal policy $O(N)$.

State-Action Value Function

Definition

In discounted infinite horizon problems, for any policy π, the state-action value function (or Q-function) $Q^\pi : X \times A \mapsto \mathbb{R}$ is

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

and the corresponding optimal Q-function is

$$Q^*(x, a) = \max_\pi Q^\pi(x, a).$$

State-Action Value Function

The relationships between the V-function and the Q-function are:

$$
\begin{aligned}
Q^\pi(x,a) &= r(x,a) + \gamma \sum_{y \in X} p(y|x,a) V^\pi(y) \\
V^\pi(x) &= Q^\pi(x, \pi(x)) \\
Q^*(x,a) &= r(x,a) + \gamma \sum_{y \in X} p(y|x,a) V^*(y) \\
V^*(x) &= Q^*(x, \pi^*(x)) = \max_{a \in A} Q^*(x,a).
\end{aligned}
$$

Value Iteration: Extensions and Implementations

Q-iteration.

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

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

Comparison

- Increased space and time complexity to $O(N|A|)$ and $O(N^2|A|^2)$
- Computing the greedy policy is cheaper $O(N|A|)$

Value Iteration: Extensions and Implementations

Asynchronous VI.

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

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

Comparison

- Reduced time complexity to $O(N|A|)$
- Increased number of iterations to at most $O(KN)$ but much smaller in practice if states are properly *prioritized*
- Convergence guarantees

How to solve *exactly* an MDP

Dynamic Programming

Bellman Equations

Value Iteration

Policy Iteration

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^{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^{π_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 $k = 1, 2, \dots, K$
   - *Policy evaluation* given π_k, compute V^{π_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 $k = 1, 2, \ldots, K$
   - *Policy evaluation* given π_k, compute V^{π_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

Remark: usually K is the smallest k such that $V^{\pi_k} = V^{\pi_{k+1}}$.

Policy Iteration: the Guarantees

Proposition

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

Proof.

From the definition of the Bellman operators and the greedy policy π_{k+1}

$$V^{\pi_k} = \mathcal{T}^{\pi_k} V^{\pi_k} \leq \mathcal{T} V^{\pi_k} = \mathcal{T}^{\pi_{k+1}} V^{\pi_k}, \tag{1}$$

Policy Iteration: the Guarantees

Proof.

From the definition of the Bellman operators and the greedy policy π_{k+1}

$$V^{\pi_k} = \mathcal{T}^{\pi_k} V^{\pi_k} \leq \mathcal{T} V^{\pi_k} = \mathcal{T}^{\pi_{k+1}} V^{\pi_k}, \tag{1}$$

and from the monotonicity property of $\mathcal{T}^{\pi_{k+1}}$, it follows that

$$\begin{aligned} V^{\pi_k} &\leq \mathcal{T}^{\pi_{k+1}} V^{\pi_k}, \\ \mathcal{T}^{\pi_{k+1}} V^{\pi_k} &\leq (\mathcal{T}^{\pi_{k+1}})^2 V^{\pi_k}, \\ &\dots \\ (\mathcal{T}^{\pi_{k+1}})^{n-1} V^{\pi_k} &\leq (\mathcal{T}^{\pi_{k+1}})^n V^{\pi_k}, \\ &\dots \end{aligned}$$

Policy Iteration: the Guarantees

Proof.

From the definition of the Bellman operators and the greedy policy π_{k+1}

$$V^{\pi_k} = \mathcal{T}^{\pi_k} V^{\pi_k} \leq \mathcal{T} V^{\pi_k} = \mathcal{T}^{\pi_{k+1}} V^{\pi_k}, \tag{1}$$

and from the monotonicity property of $\mathcal{T}^{\pi_{k+1}}$, it follows that

$$\begin{aligned} V^{\pi_k} &\leq \mathcal{T}^{\pi_{k+1}} V^{\pi_k}, \\ \mathcal{T}^{\pi_{k+1}} V^{\pi_k} &\leq (\mathcal{T}^{\pi_{k+1}})^2 V^{\pi_k}, \\ &\dots \\ (\mathcal{T}^{\pi_{k+1}})^{n-1} V^{\pi_k} &\leq (\mathcal{T}^{\pi_{k+1}})^n V^{\pi_k}, \\ &\dots \end{aligned}$$

Joining all the inequalities in the chain we obtain

$$V^{\pi_k} \leq \lim_{n\to\infty} (\mathcal{T}^{\pi_{k+1}})^n V^{\pi_k} = V^{\pi_{k+1}}.$$

Policy Iteration: the Guarantees

Proof.

From the definition of the Bellman operators and the greedy policy π_{k+1}

$$V^{\pi_k} = \mathcal{T}^{\pi_k} V^{\pi_k} \leq \mathcal{T} V^{\pi_k} = \mathcal{T}^{\pi_{k+1}} V^{\pi_k}, \tag{1}$$

and from the monotonicity property of $\mathcal{T}^{\pi_{k+1}}$, it follows that

$$\begin{aligned} V^{\pi_k} &\leq \mathcal{T}^{\pi_{k+1}} V^{\pi_k}, \\ \mathcal{T}^{\pi_{k+1}} V^{\pi_k} &\leq (\mathcal{T}^{\pi_{k+1}})^2 V^{\pi_k}, \\ &\dots \\ (\mathcal{T}^{\pi_{k+1}})^{n-1} V^{\pi_k} &\leq (\mathcal{T}^{\pi_{k+1}})^n V^{\pi_k}, \\ &\dots \end{aligned}$$

Joining all the inequalities in the chain we obtain

$$V^{\pi_k} \leq \lim_{n \to \infty} (\mathcal{T}^{\pi_{k+1}})^n V^{\pi_k} = V^{\pi_{k+1}}.$$

Then $(V^{\pi_k})_k$ is a non-decreasing sequence.

Policy Iteration: the Guarantees

Proof (cont'd).
Since a finite MDP admits a finite number of policies, then the termination condition is eventually met for a specific k.
Thus eq. 1 holds with an equality and we obtain

$$V^{\pi_k} = \mathcal{T} V^{\pi_k}$$

and $V^{\pi_k} = V^*$ which implies that π_k is an optimal policy. ■

Policy Iteration

Notation. For any policy π the reward *vector* is $r^\pi(x) = r(x, \pi(x))$ and the transition *matrix* is $[P^\pi]_{x,y} = p(y|x, \pi(x))$

Policy Iteration: the Policy Evaluation Step

- *Direct computation.* For any policy π compute

$$V^{\pi} = (I - \gamma P^{\pi})^{-1} r^{\pi}.$$

Complexity: $O(N^3)$ (improvable to $O(N^{2.807})$).

Policy Iteration: the Policy Evaluation Step

- *Direct computation.* For any policy π compute

$$V^{\pi} = (I - \gamma P^{\pi})^{-1} r^{\pi}.$$

Complexity: $O(N^3)$ (improvable to $O(N^{2.807})$).

- *Iterative policy evaluation.* For any policy π

$$\lim_{n \to \infty} \mathcal{T}^{\pi} V_0 = V^{\pi}.$$

Complexity: An ϵ-approximation of V^{π} requires $O(N^2 \frac{\log 1/\epsilon}{\log 1/\gamma})$ steps.

Policy Iteration: the Policy Evaluation Step

- *Direct computation.* For any policy π compute

$$V^\pi = (I - \gamma P^\pi)^{-1} r^\pi.$$

Complexity: $O(N^3)$ (improvable to $O(N^{2.807})$).

- *Iterative policy evaluation.* For any policy π

$$\lim_{n\to\infty} \mathcal{T}^\pi V_0 = V^\pi.$$

Complexity: An ϵ-approximation of V^π requires $O(N^2 \frac{\log 1/\epsilon}{\log 1/\gamma})$ steps.

- *Monte-Carlo simulation.* In each state x, simulate n trajectories $((x_t^i)_{t\geq 0,})_{1\leq i\leq n}$ following policy π and compute

$$\hat{V}^\pi(x) \simeq \frac{1}{n}\sum_{i=1}^{n}\sum_{t\geq 0}\gamma^t r(x_t^i, \pi(x_t^i)).$$

Complexity: In each state, the approximation error is $O(1/\sqrt{n})$.

Policy Iteration: the Policy Improvement Step

- If the policy is evaluated with V, then the policy improvement has complexity $O(N|A|)$ (computation of an expectation).

Policy Iteration: the Policy Improvement Step

- If the policy is evaluated with V, then the policy improvement has complexity $O(N|A|)$ (computation of an expectation).
- If the policy is evaluated with Q, then the policy improvement has complexity $O(|A|)$ corresponding to

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

Policy Iteration: Number of Iterations

- At most $O\left(\frac{N|A|}{1-\gamma}\log\left(\frac{1}{1-\gamma}\right)\right)$

Comparison between Value and Policy Iteration

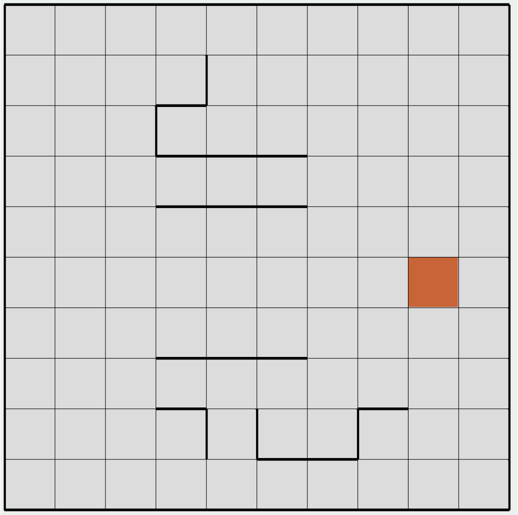
Value Iteration

- *Pros:* each iteration is very *computationally efficient*.
- *Cons:* convergence is only *asymptotic*.

Policy Iteration

- *Pros:* converge in a *finite* number of iterations (often small in practice).
- *Cons:* each iteration requires a full *policy evaluation* and it might be expensive.

The Grid-World Problem



How to solve *exactly* an MDP

Dynamic Programming

Bellman Equations

Value Iteration

Policy Iteration

Other Algorithms

- Modified Policy Iteration
- λ-Policy Iteration
- Linear programming
- Policy search

Summary

- Bellman equations provide a compact formulation of value functions
- DP provide a *general* tool to solve MDPs

Bibliography I

R. E. Bellman.
Dynamic Programming.
Princeton University Press, Princeton, N.J., 1957.

D.P. Bertsekas and J. Tsitsiklis.
Neuro-Dynamic Programming.
Athena Scientific, Belmont, MA, 1996.

W. Fleming and R. Rishel.
Deterministic and stochastic optimal control.
Applications of Mathematics, 1, Springer-Verlag, Berlin New York, 1975.

R. A. Howard.
Dynamic Programming and Markov Processes.
MIT Press, Cambridge, MA, 1960.

M.L. Puterman.
Markov Decision Processes: Discrete Stochastic Dynamic Programming.
John Wiley & Sons, Inc., New York, Etats-Unis, 1994.

Reinforcement Learning

Alessandro Lazaric
alessandro.lazaric@inria.fr

sequel.lille.inria.fr