

Stochastic Optimization

From Dynamic Programming to Reinforcement Learning

Edoardo Fadda

Dip. di Scienze Matematiche – Politecnico di Torino

e-mail: edoardo.fadda@polito.it

Version: 27 novembre 2024

NOTA: A uso didattico interno. Da non postare o ridistribuire.

Markov Decision Process

- *Markov chains* are dynamical models whose time evolution follows a random law. They are a special case of stochastic processes
- What makes them special among the other stochastic processes is the particularly simple structure of the dependence between the observations.
- Markov chains are used as models for an enormous amount of phenomena. Queues, telecommunications, biology, population dynamics, epidemiology. They are often oversimplified models, but simple enough to be studied analytically and by simulations
- Despite the underlying mathematics is not too complicated, very often simulations are needed to evaluate simple quantities, like expectations or probabilities

Ex. 1: Weather forecast

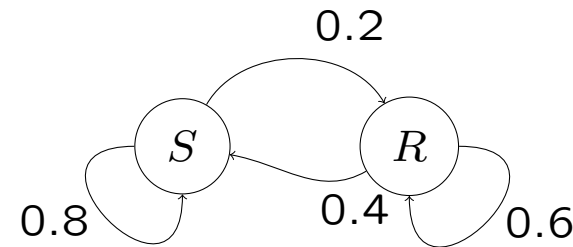
Let us define the following Markov model of weather conditions.

- Let the state be either rainy (state 1) or sunny (state 2).

- Let us make the strong assumption that the weather tomorrow only depends on the weather today (e.g. the weather yesterday, or in the past week does not matter at all). In particular, the probability that a sunny day is followed by another sunny day be 0.8, and the probability that a rainy day is followed by another rainy day be 0.6.

The system evolves according to the following transition matrix

$$P = \begin{pmatrix} 0.8 & 0.2 \\ 0.4 & 0.6 \end{pmatrix}$$



Let $(X_t)_{t=0}^{\infty}$ be a discrete-time stochastic process with a discrete state space $S = \{1, 2, \dots\}$.

$(X_t)_{t=0}^{\infty}$ is a discrete-time Markov Chain if for any $j, i, i_{t-1}, \dots, i_0 \in S$, holds the *Markov property*

$$P(X_{t+1} = j \mid X_t = i, X_{t-1} = i_{t-1}, \dots, X_0 = i_0) = P(X_{t+1} = j \mid X_t = i) = p_{i \rightarrow j}.$$

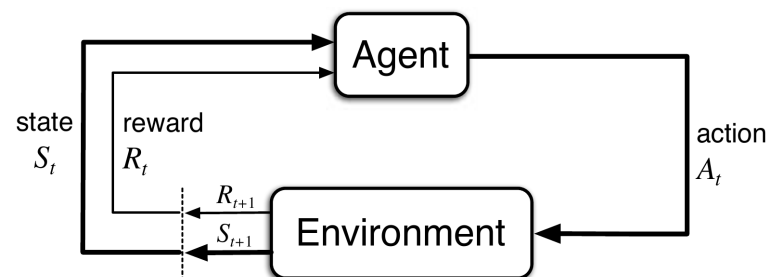
It says that the probabilities associated with a future state only depend upon the current state, and not on the full history of the process.

$p_{i \rightarrow j}$ are called the (one step) *transition probabilities* (also called transition matrix)

Markov decision process

A *Markov decision process* is defined by:

- A set of *actions* (that includes both transition matrices and instantaneous rewards) called \mathcal{A}
- A set of *states* called \mathcal{S}
- A *policy* $\mu : s \in \mathcal{S} \rightarrow a \in \mathcal{A}$
- A *performance metric* to decide which policies are good, and in particular which is the *optimal* one. It depends on the *reward* $R_t(s_t, a_t)$.

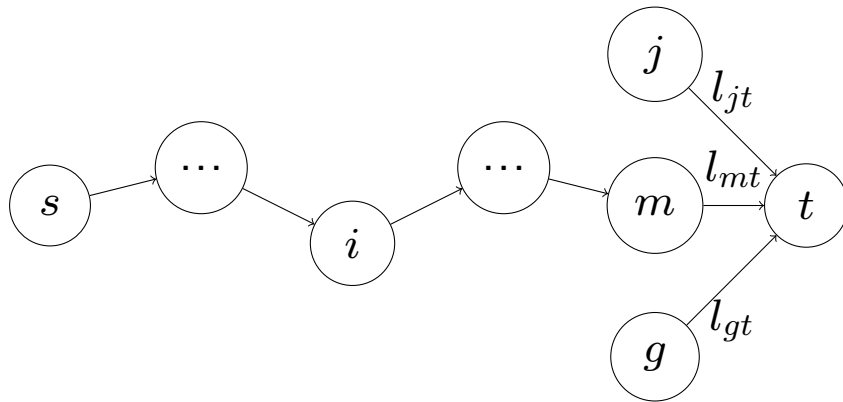


Markov decision processes are *Markov chains* in which we can change the transition probability matrix.

Shortest Path Problem

Given a graph $G = (V, A)$ and a length l_{ij} for each arc, find the shortest path from origin node s to destination node t . Is the shortest path problem a MDP?

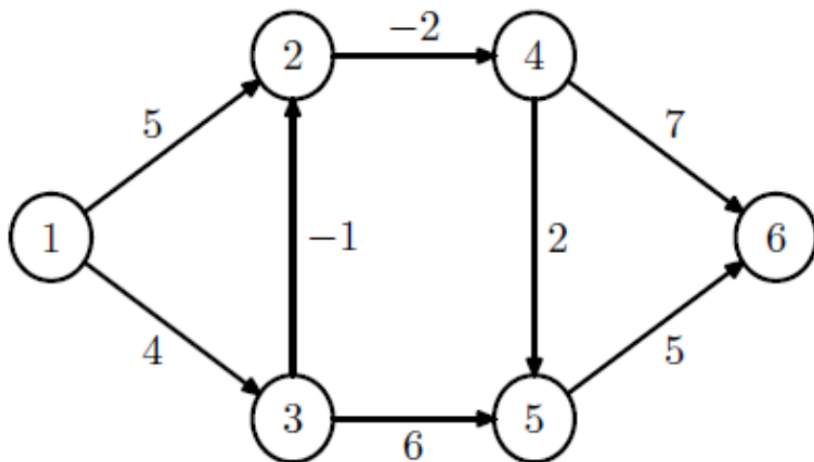
Bellman's optimality principle (1953): "An optimal policy is made of a set of optimal sub-policies"



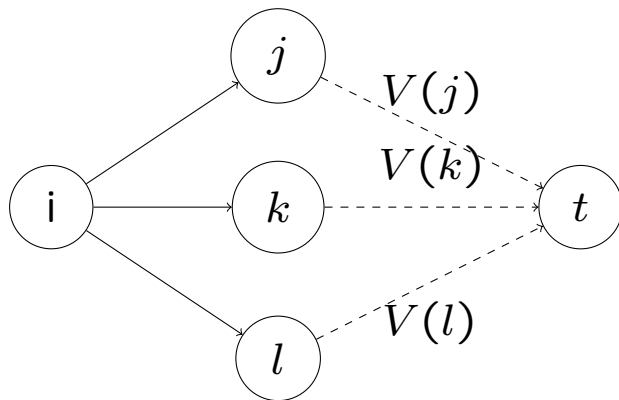
If $P_{s \rightarrow t}$ is optimal, then both $P_{s \rightarrow i}$ and $P_{i \rightarrow t}$ are optimal.

Let $V(i)$ be the shortest path from node i to node t . Then:

- $V(t) = 0$
- $V(i) = \min_{j:(i,j) \in A} l_{ij} + V(j)$



Let us define $V(i) = L(i \xrightarrow{*} t) = L(i \xrightarrow{*} j) + L(j \xrightarrow{*} t)$.
We call $V(i)$ *value function*.



$S(i)$ set of successors of node i

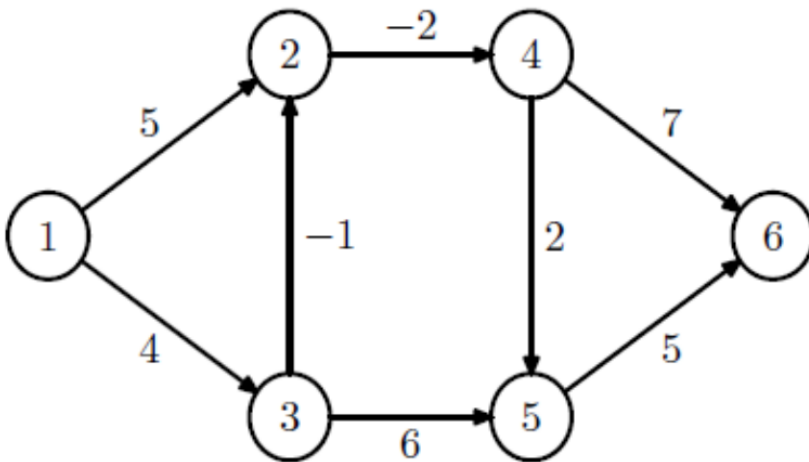
$$V(i) = \min_{j \in S(i)} l_{ij} + V(j), \quad V_t = 0.$$

Given a policy $\mu : V \rightarrow V$, we have that

$$V_\mu = l_{s, \mu(s)} + l_{\mu(s), \mu(\mu(s))} + \dots$$

Iterative procedure, we start with:

$$V^{(0)} = [+\infty, +\infty, +\infty, +\infty, +\infty, 0]$$



Utility maximization

Let us define a utility function $U(\cdot)$ such that $U(0) = 0$ and $U'(\cdot) > 0$

- We have an initial capital x_0
- x_1, x_2, \dots , the levels of capitals at $t = 1, 2, \dots$
- At time t , we can spend an amount $a_t \in [0, x_t]$
- the rest of the money is invested:

$$x_{t+1} = \lambda(x_t - a_t), \quad \lambda \geq 0.$$

Over a period of length T , the total utility is:

$$U(a_0) + U(a_1) + \dots + U(a_{T-1})$$

For any t , we define:

$$V_{T-t}(x) = \text{maximum total utility from } t \text{ on.}$$

Let us consider $U(a) = a$.

For $t = T - 1$, we have:

$$V_1(x) = \max_{a \in [0, x]} a = x$$

For $t = T - 2$, we have:

$$V_2(x) = \max_{a \in [0, x]} a + V_1(\lambda(x - a)) = \max_{a \in [0, x]} a + \lambda(x - a) = \max_{a \in [0, x]} \lambda x + (1 - \lambda)a = \lambda x \quad [a = 0]$$

For $t = T - 3$, we have:

$$V_3(x) = \max_{a \in [0, x]} a + V_2(\lambda(x - a)) = \max_{a \in [0, x]} a + \lambda^2(x - a) = \max_{a \in [0, x]} \lambda^2 x + (1 - \lambda^2)a = \lambda^2 x$$

...

$$V_T(x) = \lambda^{T-1}x$$

The optimal strategy is to keep all the money until the last time since $\lambda > 1$. Therefore the optimal for all the $t \neq T$ is 0.

Stochastic DP for finite time horizons

The natural solution process goes backward in time:

$$V_T(s_T) = R_T(s_T) \quad \forall s_T$$

We can find the value function at each time instant by a recursive unfolding:

$$V_{T-1}(s_{T-1}) = \underset{a_{T-1}}{\text{opt}} R_{T-1}(s_{T-1}, x_{T-1}) + \gamma \mathbb{E}[V_T(S_T) | s_{T-1}, x_{T-1}]$$

...

$$V_0(s_0) = \underset{a_0}{\text{opt}} R_0(s_0, a_0) + \gamma \mathbb{E}[V_1(S_1) | s_0, x_0]$$

From now on, we will consider:

- infinite time horizon problems.
- finite state space

Consider the sequence of stochastic rewards: R_0, R_1, \dots . At time t , we define:

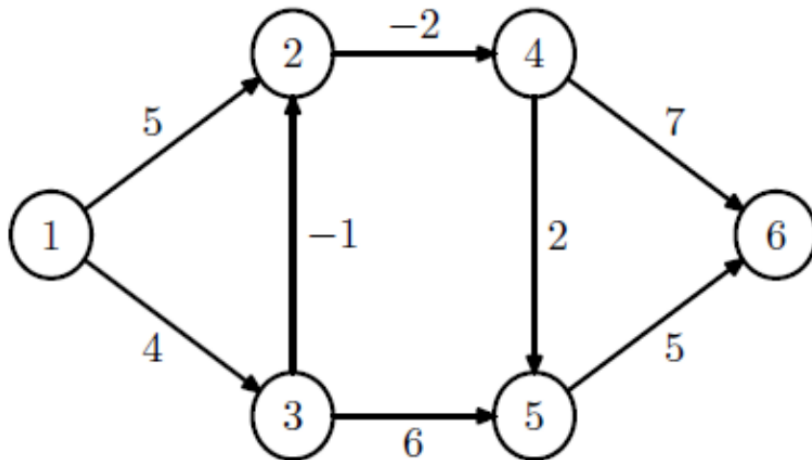
$$\begin{aligned} G_t &= R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots = R_{t+1} + \gamma(R_{t+2} + \gamma R_{t+3} + \dots) = \\ &= R_{t+1} + \gamma G_{t+1} \end{aligned}$$

Given a policy $\mu : \mathcal{S} \rightarrow \mathcal{A}$, we define the state-value function:

$$\begin{aligned}
 V_{\mu(s)} &\doteq \mathbb{E}[G_t | S_t = s] = \mathbb{E}[R_{t+1} + \gamma G_{t+1} | S_t = s] = \\
 &= \mathbb{E}[R_{t+1} | S_t = s] + \gamma \mathbb{E}[G_{t+1} | S_t = s] =^* \\
 &= \mathbb{E}[R_{t+1} | S_t = s] + \gamma \mathbb{E}[\mathbb{E}[G_{t+1} | S_{t+1} = s'] | S_t = s] \\
 &= \mathbb{E}[R_{t+1} | S_t = s] + \mathbb{E}[V_{\mu}(s') | S_t = s] \\
 &= \mathbb{E}[R_{t+1} + \gamma V_{\mu}(s') | S_t = s]
 \end{aligned}$$

$$^* \mathbb{E}[X] = \mathbb{E}_Y[\mathbb{E}_X[X|Y]] \implies \mathbb{E}_{\mu}[G_{t+1} | S_t = s] = \mathbb{E}[\mathbb{E}[G_{t+1} | S_{t+1} = s'] | S_t = s]$$

Compute the value of the greedy policy:



$$\begin{aligned}
 V_{\text{greedy}}(1) &= 4 + V_{\text{greedy}}(3) \\
 V_{\text{greedy}}(2) &= -2 + V_{\text{greedy}}(4) \\
 &\dots
 \end{aligned}$$

We are interested in

$$V^*(s) = \underset{\mu}{\text{opt}} V_{\mu}(s)$$

In the infinite horizon setting the goal is:

$$\begin{aligned} \underset{a_t}{\text{opt}} \mathbb{E} \left[\sum_{t=0}^{+\infty} \gamma^t R_t(s_t, a_t) | s_0 \right] &= \underset{a_t}{\text{opt}} \mathbb{E} [R_0 + \gamma G_0 | s_0] = \underset{a_0}{\text{opt}} \mathbb{E} [R_0 | s_0] + \gamma \underset{a_1, \dots}{\mathbb{E}} [\text{opt } G_0 | s_0] = \\ \underset{a_0}{\text{opt}} \mathbb{E} [R_0 | s_0] + \gamma \underset{a_1, \dots}{\mathbb{E}} [\text{opt } V(s') | s_0] &= \underset{a_0}{\text{opt}} \mathbb{E} [R_0 | s_0] + \gamma \mathbb{E} [V^*(s') | s_0] \end{aligned}$$

By assuming a *stationary model*, the optimal policy is:

$$\underset{a}{\text{opt}} \mathbb{E} [R(s, a) + \gamma V^*(s')]$$

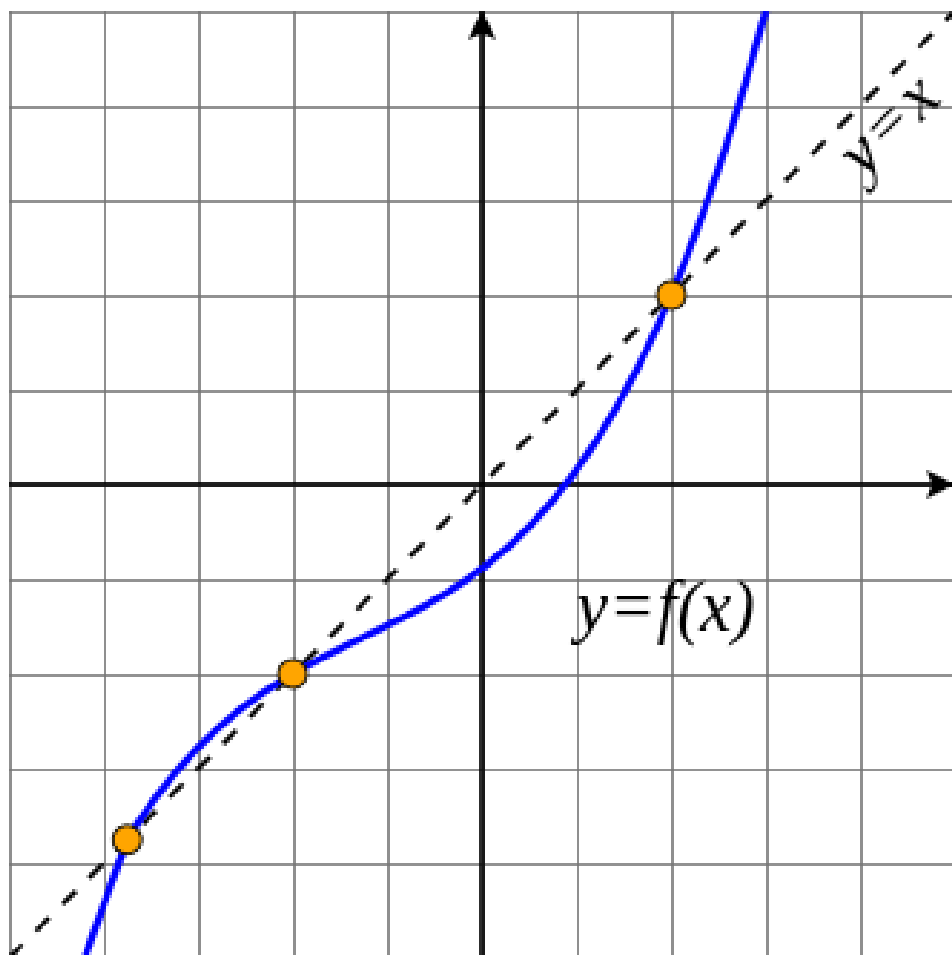
If we set $f(s, a) \doteq \mathbb{E} [R(s, a)]$, we can write:

$$V^*(s) = \underset{a}{\text{opt}} f(s, a) + \gamma \mathbb{E} [V^*(s')] = \underset{a}{\text{opt}} f(s, a) + \gamma \sum_{j \in \mathcal{S}} \pi_{s,a,j} V^*(j)$$

where, since the state space is finite $\mathcal{S} = \{1, \dots, n\}$ and the value function $V : \mathcal{S} \rightarrow \mathbb{R}$ is a vector in \mathbb{R}^n .

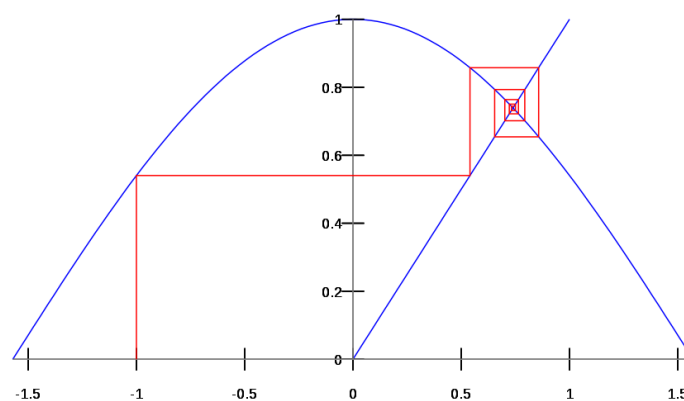
Here, the value function is a *fixed point* of an operator.

A fixed point x of a function F is a point such that: $F(x) = x$

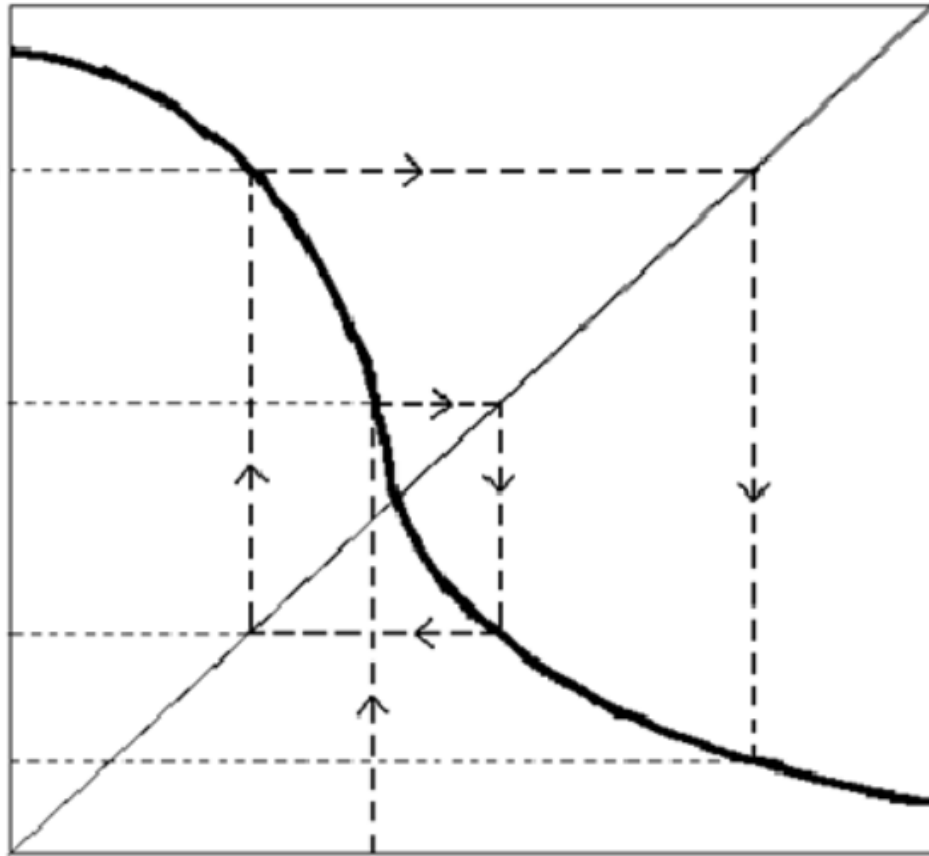


We can think of finding (under some condition) the *fixed-point iteration* as:

$$x_{n+1} = f(x_n), \quad n = 0, 1, 2, \dots$$



An unlucky case:



It turns out that the operators used in DP satisfy the condition for the fixed point iteration to converge.

Our goal is to find V to solve:

$$V(s) = \operatorname{opt}_{a \in \mathcal{A}(s)} f(s, a) + \gamma \sum_{s' \in \mathcal{S}} \pi_{s, a, s'} V(s')$$

We call that V , V^* . We define the two operators:

$$[TV](s) = \operatorname{opt}_{a \in \mathcal{A}(s)} f(s, a) + \gamma \sum_{s' \in \mathcal{S}} \pi_{s, a, s'} V(s'), \quad s \in \mathcal{S}$$

and, given a policy μ :

$$[T_\mu V](s) = f(s, \mu(s)) + \gamma \sum_{s' \in \mathcal{S}} \pi_{s, \mu(s), s'} V(s'), \quad s \in \mathcal{S}$$

- V^* is such that $TV = V$.
- V_μ is such that $\mu: T_\mu V_\mu = V_\mu$.

Value Iteration

Algorithm 1 Value Iteration

1: Start with an initial value function $V^{(0)}$.

2: $k=0$, $\text{stop} = \text{True}$

3: **while** stop **do**

4:

$$V^{(k+1)}(s) = \underset{a \in \mathcal{A}(s)}{\text{opt}} f(s, a) + \gamma \sum_{s' \in \mathcal{S}} \pi_{s,a,s'} V^{(k)}(s') \quad \forall s \in \mathcal{S}$$

5: **if** $\|V^{(k+1)} - V^{(k)}\|_{\infty} < \epsilon$ **then**

6: $\text{stop} = \text{False}$

7: **end if**

8: **end while**

The optimal policy is: $\mu^*(s) = \arg \underset{a \in \mathcal{A}(s)}{\text{opt}} f(s, a) + \gamma \sum_{s' \in \mathcal{S}} \pi_{s,a,s'} V^{(K)}(s') \quad \forall s \in \mathcal{S}$

- It is possible to speed up
- Each iteration requires to solve an optimization problem

- The algorithm will converge to the optimal policy.

Policy Iteration

T_μ does not involve any optimization:

$$[T_\mu V](s) = f(s, \mu(s)) + \gamma \sum_{s' \in \mathcal{S}} \pi_{s, \mu(s), s'} V(s'), s \in \mathcal{S}$$

Thus, to find V_μ we can write it in vector as:

$$V_\mu = f_\mu + \gamma \Pi_\mu V_\mu \rightarrow (I - \gamma \Pi_\mu) V_\mu = f_\mu$$

Given μ , V_μ we can improve it by:

$$\mu'(s) = \arg \operatorname{opt}_{a \in \mathcal{A}(s)} f(s, a) + \gamma \sum_{s' \in \mathcal{S}} \pi_{s, a, s'} V_\mu(s'), s \in \mathcal{S}$$

μ' is better than μ and if they are equal...

Algorithm 2 Policy Iteration

- 1: Start with an initial policy $\mu^{(0)}$.
- 2: $k=0$, $\text{stop} = \text{True}$
- 3: **while** stop **do**
- 4: evaluate policy $\mu^{(k)}$ by solving: $(I - \gamma \Pi_\mu)V_\mu = f_\mu$
- 5: rollout

$$\mu^{(k+1)}(s) = \arg \operatorname{opt}_{a \in \mathcal{A}(s)} f(s, a) + \gamma \sum_{s' \in \mathcal{S}} \pi_{s,a,s'} V_\mu(s'), \quad s \in \mathcal{S}$$

- 6: **if** $\mu^{(k)} = \mu^{(k+1)}$ **then**
 - 7: $\text{stop} = \text{False}$
 - 8: **end if**
 - 9: **end while**
-

- Each iteration of policy iteration is more expensive
- The algorithm will converge to the optimal policy.

Generalized Policy Iteration

Policy iteration can be described by the sequence

$$\mu_0 \xrightarrow{E} V_{\mu_0} \xrightarrow{I} \mu_1 \xrightarrow{E} V_{\mu_1} \xrightarrow{I} \dots$$

If $(I - \gamma \Pi_\mu)V_\mu = f_\mu$ cannot be solved, we can evaluate a stationary policy μ by fixed point iteration of operator T_μ :

$$V_\mu^{(k+1)}(s) = f(s, \mu(s)) + \gamma \sum_{s' \in \mathcal{S}} \pi_{s, \mu(s), s'} V_\mu^{(k)}(s'), s \in \mathcal{S}$$

Convergence may be slow. If we stop before (at iteration K), we can use rollout with an estimation of V_μ :

$$\mu'(s) = \arg \operatorname{opt}_{a \in \mathcal{A}(s)} f(s, a) + \gamma \sum_{s' \in \mathcal{S}} \pi_{s, a, s'} \hat{V}_\mu(s'), s \in \mathcal{S}$$

This generalized both policy and value iteration: with $K = 0$ we have value iteration, for K that reaches convergence we have policy iteration in between we have *generalized policy iteration*.

Ex:

	-1	+1

	-1	+1
0		

$\mu^{(0)}$: random movement

$$V_{\mu^{(0)}}^{(1)}(0,0) = \gamma \left[\frac{1}{2} V_{\mu^{(0)}}^{(0)}(1,0) + \frac{1}{2} V_{\mu^{(0)}}^{(0)}(0,1) \right]$$

...

$$V_{\mu^{(0)}}^{(1)}(1,2) = \gamma \left[\frac{1}{3} V_{\mu^{(0)}}^{(0)}(1,1) + \frac{1}{3} V_{\mu^{(0)}}^{(0)}(2,2) + \frac{1}{3} V_{\mu^{(0)}}^{(0)}(0,2) \right]$$

Policy improvement...

Q-factors

In GPI we have to solve

$$\arg \operatorname{opt}_{a \in \mathcal{A}(s)} f(s, a) + \gamma \sum_{s' \in \mathcal{S}} \pi_{s,a,s'} V(s')$$

What if we do not know $\pi_{s,a,s'}$?

Can we avoid computing the opt of an expected value?

Q-factor $Q(s, a) : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ aims at measuring the value of taking action a when in state s . Recall, the value of being in s and follow policy μ :

$$V_\mu(s) = f(s, \mu(s)) + \gamma \sum_{s' \in \mathcal{S}} \pi_{s,\mu(s),s'} V_\mu(s'), s \in \mathcal{S}$$

The Q-factors for policy μ are:

$$Q_\mu(s, a) = f(s, a) + \gamma \sum_{s' \in \mathcal{S}} \pi_{s,a,s'} V_\mu(s'), s \in \mathcal{S}, a \in \mathcal{A}$$

The optimal Q-factors are:

$$Q^*(s, a) = f(s, a) + \gamma \sum_{s' \in \mathcal{S}} \pi_{s,a,s'} V^*(s'), \quad s \in \mathcal{S}, a \in \mathcal{A}$$

Observing that:

$$V^*(s) = \operatorname{opt}_{a \in \mathcal{A}(s)} Q^*(s, a), \quad s \in \mathcal{S}$$

Hence:

$$Q^*(s, a) = f(s, a) + \gamma \sum_{s' \in \mathcal{S}} \pi_{s, a, s'} \operatorname{opt}_{a \in \mathcal{A}(s')} Q^*(s', a), \quad s \in \mathcal{S}, x \in \mathcal{A}$$

- **Good News:** we swapped opt and expected value \rightarrow estimate with unknown $\pi_{s, a, s'}$.
- **Bad News:** Instead of $|\mathcal{S}|$ values we have to evaluate $|\mathcal{S}| \times |\mathcal{A}|$ values but we may devise a compact approximation architecture (linear regression, ... deep neural networks).

Post-decision State Variables

- $s_t \longrightarrow a_t \xrightarrow{\xi_{t+1}} s_{t+1}$
- It may be convenient to introduce an intermediate state, observed after the decision a_t is made, but before the risk factor is realized.
- $s_t \longrightarrow a_t \longrightarrow s_t^a \xrightarrow{\xi_{t+1}} s_{t+1}$

EX: Imagine to manage an inventory. The state evolution is:

$$I_{t+1} = I_t + a_t - d_{t+1}.$$

It can be useful to consider:

$$I_t^a = I_t + a_t; \quad I_{t+1} = I_t^a - d_{t+1}.$$

We introduce the value of the post-decision state as:

$$V^a(s^a) = \mathbb{E}[V(s)|s^a]$$

Thus, we can rewrite













$$V(s) = \underset{a \in \mathcal{A}(s)}{\text{opt}} f(s, a) + \gamma \mathbb{E}[V(s')] = \underset{a \in \mathcal{A}(s)}{\text{opt}} f(s, a) + \gamma V^a(s^a)$$

Thus:

$$V^a(s^a) = \mathbb{E}[V(s)|s^a] = \mathbb{E}[\underset{a \in \mathcal{A}(s)}{\text{opt}} f(s, a) + \gamma V^a(s^a)]$$

Again, we swap \mathbb{E} and opt .

What is the difference between post-decision and Q-factors? Q-factors are post-decision variables but ...

-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
	-100 	-100 	-100 	-100 	-100 	-100 	-100 	-100 	-100 	-100 	+10 

- $Q(s, a)$: position on the grid, a : feasible moves
- Post-decision states would simply be grid positions.
- Since it does not matter how you land on a given tile, all that matters is the remaining number of steps to the goal, post-decision states are ok.

- we only need a single post-decision value to predict the remaining number of steps
- Having four distinct $Q(s, a)$ values pointing to a single tile is actually a quite inefficient solution.