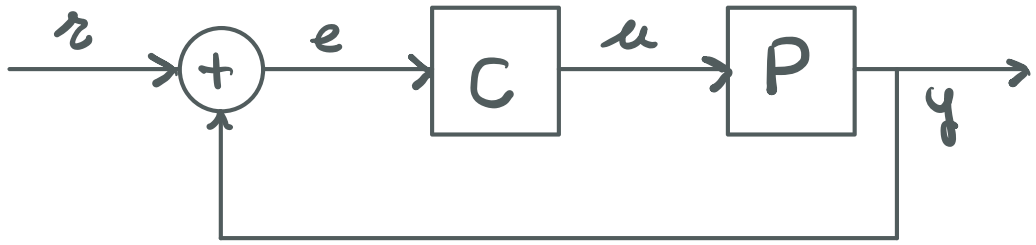
Introduction.

Consider a model provided with a plant and a controller:

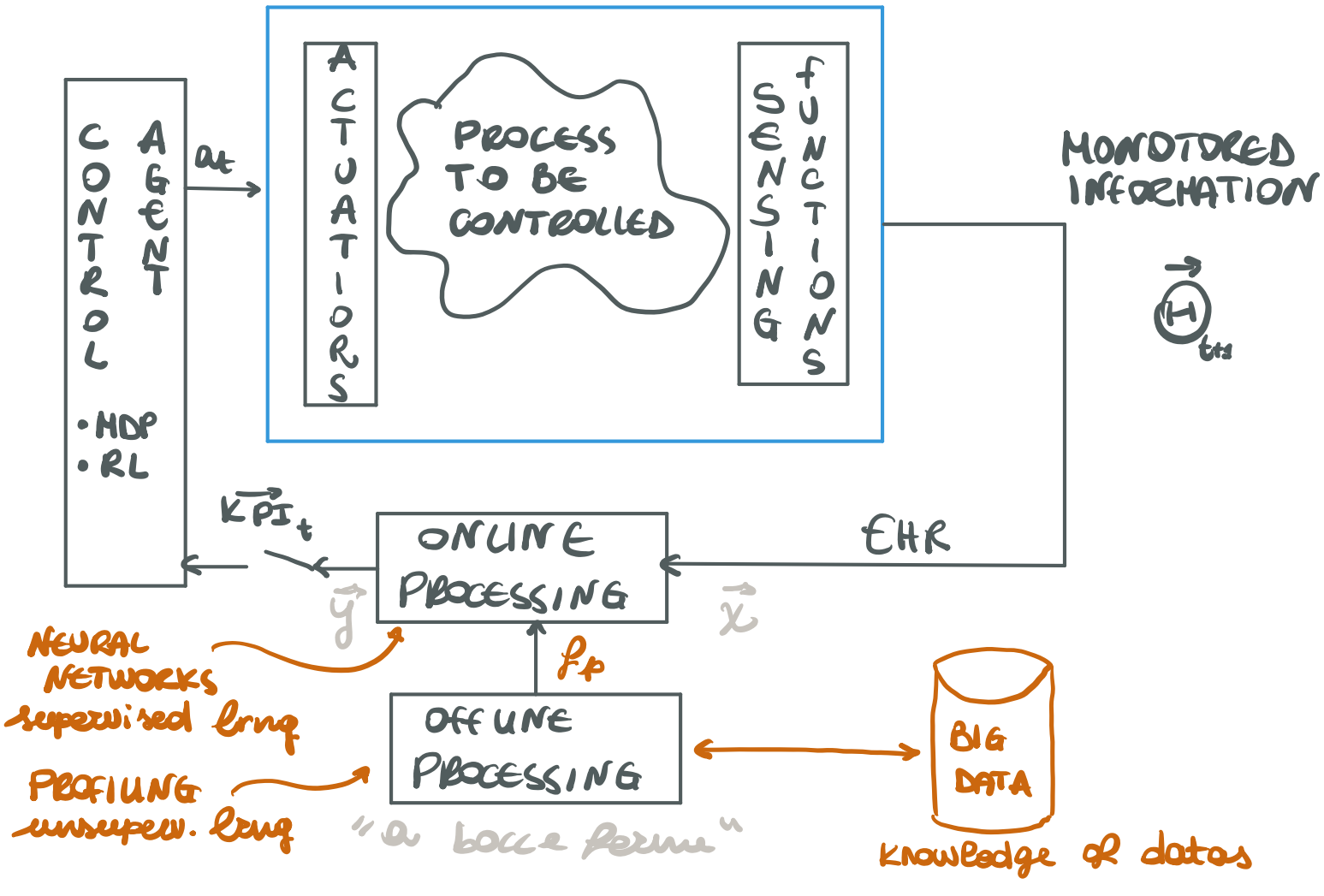


error $e = r - y$

$r = y_d$

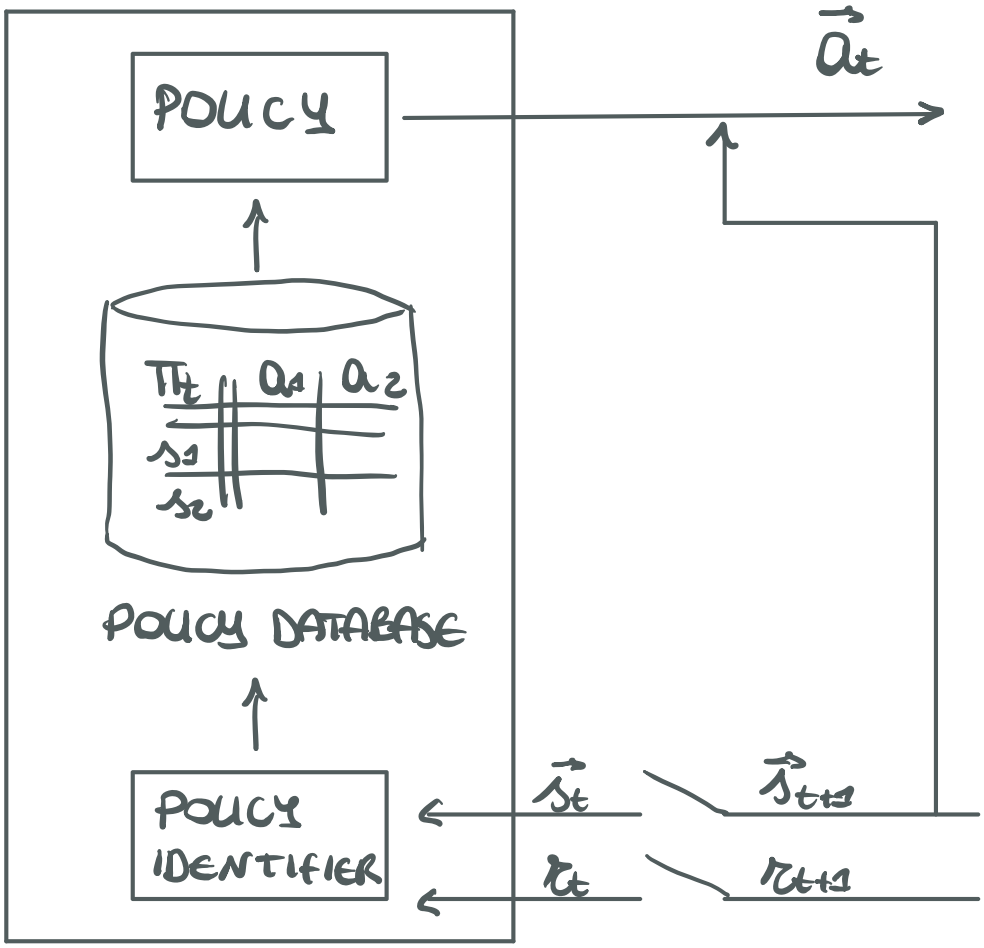
This scheme is good for a beginner in the control field but it is actually aged because, for the porposes of the analysis, the process model needs to be well known. So we will need to take to account model-free techniques.

First of all it is needed to get to know better a new a more specific scheme on which we will work from now on.

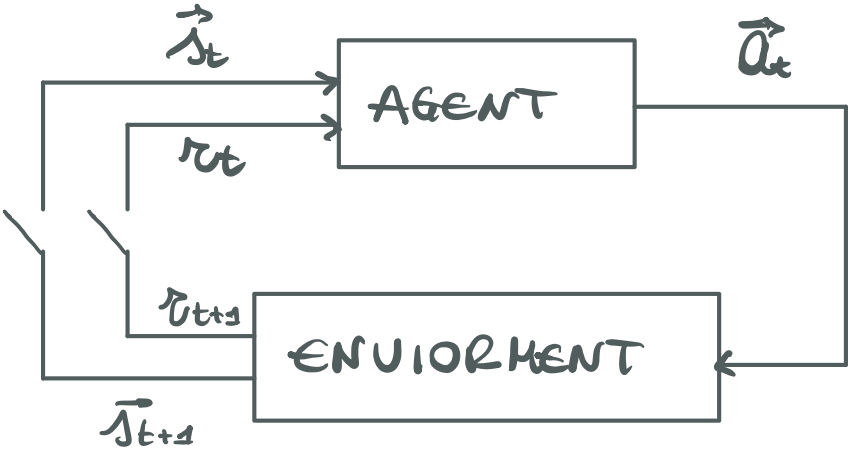


Internal structure of:

AGENT



OVERALL CONTROL SCHEME



Description:

Process to be controlled: *the general process.*

Example: Human body, energy networks…

Actuation functionality: a component of a device or machine that is responsible for moving and controlling a mechanism or system that helps it to achieve physical movements by converting energy, often electrical, air, or hydraulic, into mechanical force.

Sensing functionality: *giving a certain amount of variables to control that most of the time are etherogeneous and complex so it is needed to monitor the actual performance which will be summarised thanks to online processing.*

Online processing: *helps to elaborate the monotored information throught the machine learning technique of supervised learning. Online means that everytime a new sample appears it can be elaborated in real time.*

Offline processing: *giving a certain data base it creates a general function that will assign a sort of "label" to each new sample that will appear during online processing. This will be done through machine learning techniques of unsupervised learning such as profiling.*

Control Agent: *receives orders from the Automation Manager and issues commands that are based on the defined automation policy. If the process model is known then we will use Model Predictive Control (MPC), if the process model is uknown then Reinforcement Learning (RL) will be used and Markov Decision Process (MDP) if it is something in between.*

Define now the system variables:

$\overrightarrow{KPI_t}$ *Key Performance Indicator.*

$$\overrightarrow{KPI_t} = \begin{pmatrix} \vec{s}_t \\ r_t \end{pmatrix}$$

$\vec{s}_t$ state.

r_t reward.

a_t action.

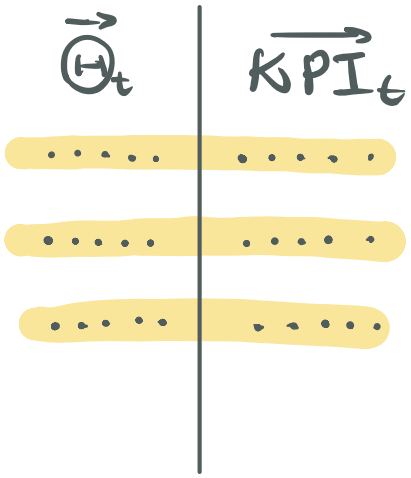
Please note that sometimes it is good to use both MPC and RL approaches because even if a phenomena it is easy to model this one can be too simple for engineering porpuses.

The Processing blocks.

As we already said a **Profiling Function** will be defined in the offline processing block such that:

$$\overrightarrow{KPI_t} = f_p\ \vec{\Theta}_t$$

f_p is the rule which links the inputs and outputs variables that are expressed through the training Neural Networks (NN) table that looks like this:



The trining can be done offline.

Characteristic of the State.

The state $\vec{s}$ has to be pregnant, compact and able to sumarise the past.

Markov state property for good performances.

$$Pr\{s_{t+1} \mid s_t, a_t\} = Pr\{s_{t+1} \mid s_t, s_{t-1}, \dots; a_t, a_{t-1}, \dots\}$$

probability

For state we mean whatever information is available to the agent. Empirically speaking it is proved that RL can provide good performances even though this property is not compleatelly true, i.e. the two sides of the equation are not *exactly* equal.

The Reward.

It is related to the overall control loop: higher rewards better is the performance. Sometimes it is not necessary to get real time rewards but we will use long term returns: "it is better an egg today or a chicket tomorrow?" Well this of course depends on the specifications of the problem but in any case we define a *stocastic variable* called long term return.

$$R_t = \sum_{k=0}^{T} \gamma^k r_{t+k+1}$$

certain weight

rewards

The agent goal is to maximise the cumulative rewards it recieves in the long run which is the expected value of the long term return:

$$E_\pi[R_t] = E_\pi\left[\sum_{k=0}^{T} \gamma^k r_{t+k+1}\right]$$

Markov property to the reward.

$$Pr\{r_{t+1} \mid s_{t+1}, s_t; a_t\} =$$

$$Pr\{r_{t+1} \mid s_{t+1}, s_t, s_{t-1}, \dots; a_t, a_{t-1}, \dots\}$$

The Key Performance Indicator.

It is represented by a switch which doesn't really exists; it is a simbol to represent the campionation time for which state e rewards are given.

$$\Rightarrow \quad s_t \mapsto s_{t+1}$$
$$r_t \mapsto r_{t+1}$$

The Action.

It is physically a non deterministic selected action that the agent will perform on the actuator functionalities. This is selected according to what we call a policy i.e. the probability of doing an action on a given state.

Since the process is a black box, the agent will perform an action on it and then monitor the information given by the state and the reward.

Notice that

$$a_t \mapsto u(t)$$
$$\vec{H}_t \mapsto x(t)$$
$$\vec{KPI}_t \mapsto y(t)$$
$$\pi_t \mapsto GC(s)$$

control scheme

Markov Decision Problem.

Attention to the dimension of the problem: *Curse of Dimensionality.*

This problem is solvable thanks to NN.

MDP: the model of the process is usually known.

Before we used Plant t-function $P(s)$

Now we will use the **Transiction Probability**

$$P^{a}_{ss'} = Pr\{s_{t+1} = s' \mid s_t = s; a_t = a\}$$

Return: expected value of the next reward

$$R^{a}_{ss'} = E\{r_{t+1} \mid s_t = s, s_{t+1} = s'; a_t = a\}$$

Each of these parameters are known.

ex: Recycling Robot.

Actions:

- Search for cans to trash.
- Stay still: people will bring to it cans to trash -steady state-.
- Recharge battery.

a_1 = WAIT

a_2 = SEARCH

a_3 = RECHARGE

s_1 = HIGH

s_2 = LOW

battery state

expected number of cans

reward

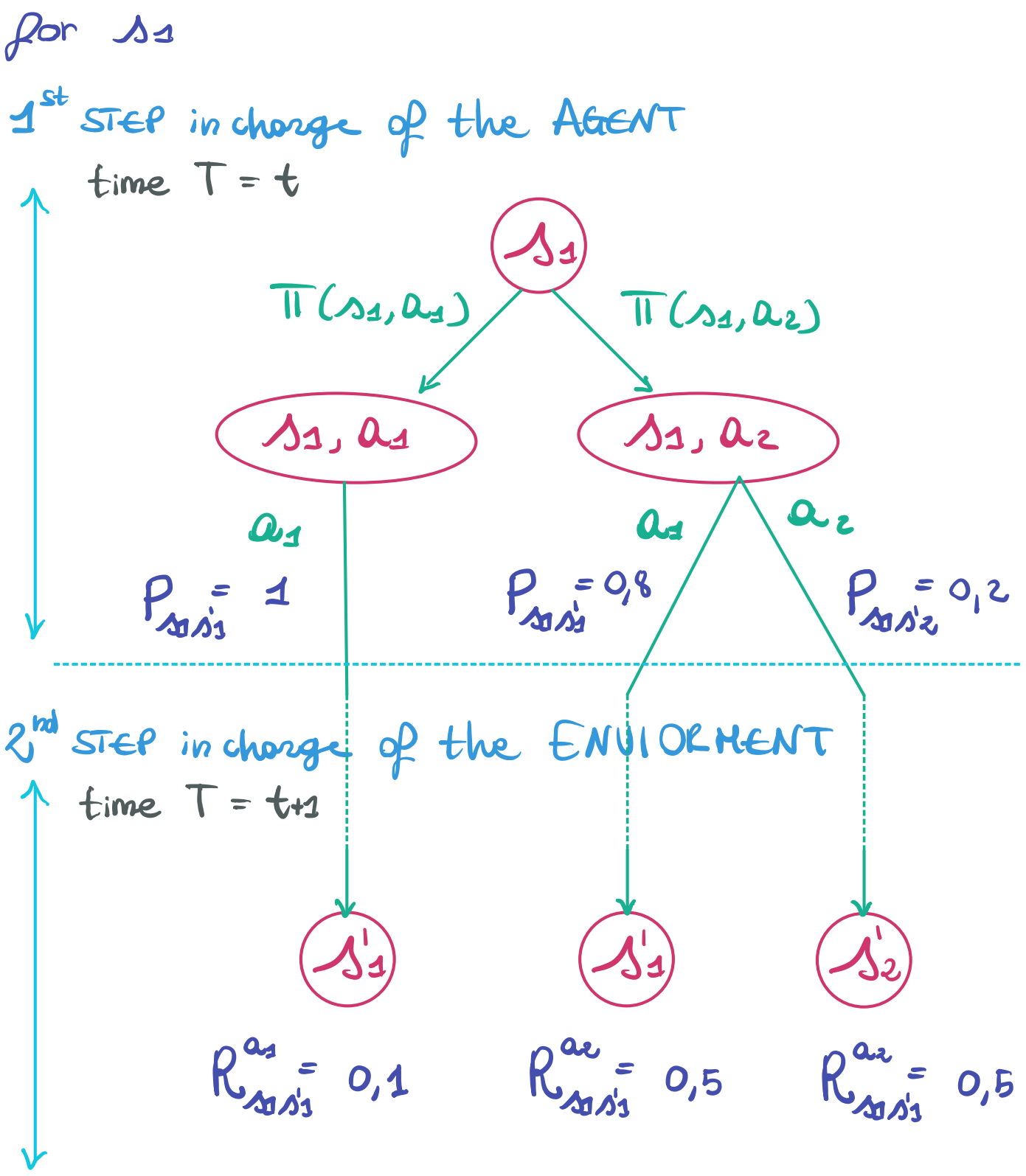
$S = \{s_1, s_2\}$ *set of states.*

$A(s_1) = \{a_1, a_2\}$

$A(s_2) = \{a_1, a_2, a_3\}$

set of actions that it is possible to perform at each state

Back-up diagram.



And we will do the same resoning for the second state.

Bellman Equation.

The aim is to maximise the expected value of the long term return.

$$E[R_t] = E_\pi\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1}\right]$$

with $T \mapsto \infty$

$0 \leq \gamma \leq 1$ *without any lost of generality*

We want to find the *optimal policy* $\pi^*(s,a)$ which maximise $E_\pi[R_t]$

We can write the expected value of long term return also as

$$E_\pi[R_t] = \sum_{s \in S} \boxed{E_\pi[R_t \mid s_t = s]} P_r\{s_t = s\}$$

$$V_\pi(s)$$

state value function for the policy

$$V_\pi(s) = E_\pi[R_t \mid s_t = s]$$

$$= E_\pi\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \,\middle|\, s_t = s\right\}$$

This represents the expected long termi reward when starting in the state and following the policy thereafter.

Bellman Equation.

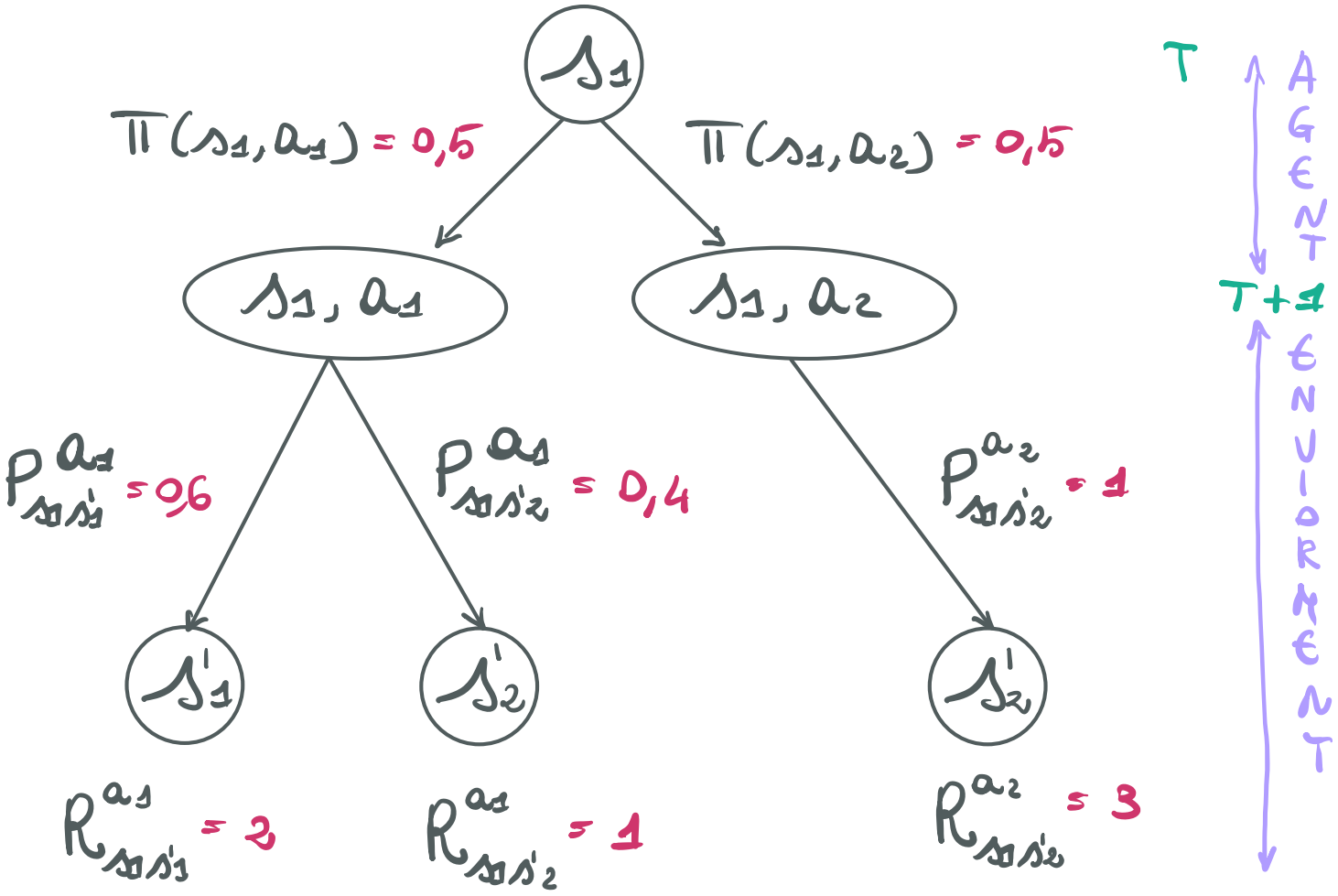
$$V_\pi(s) = \sum_{a \in A(s)} \pi(a|s) \sum P^a_{ss'}\left[R^a_{ss'} + \gamma V^\pi(s')\right]$$

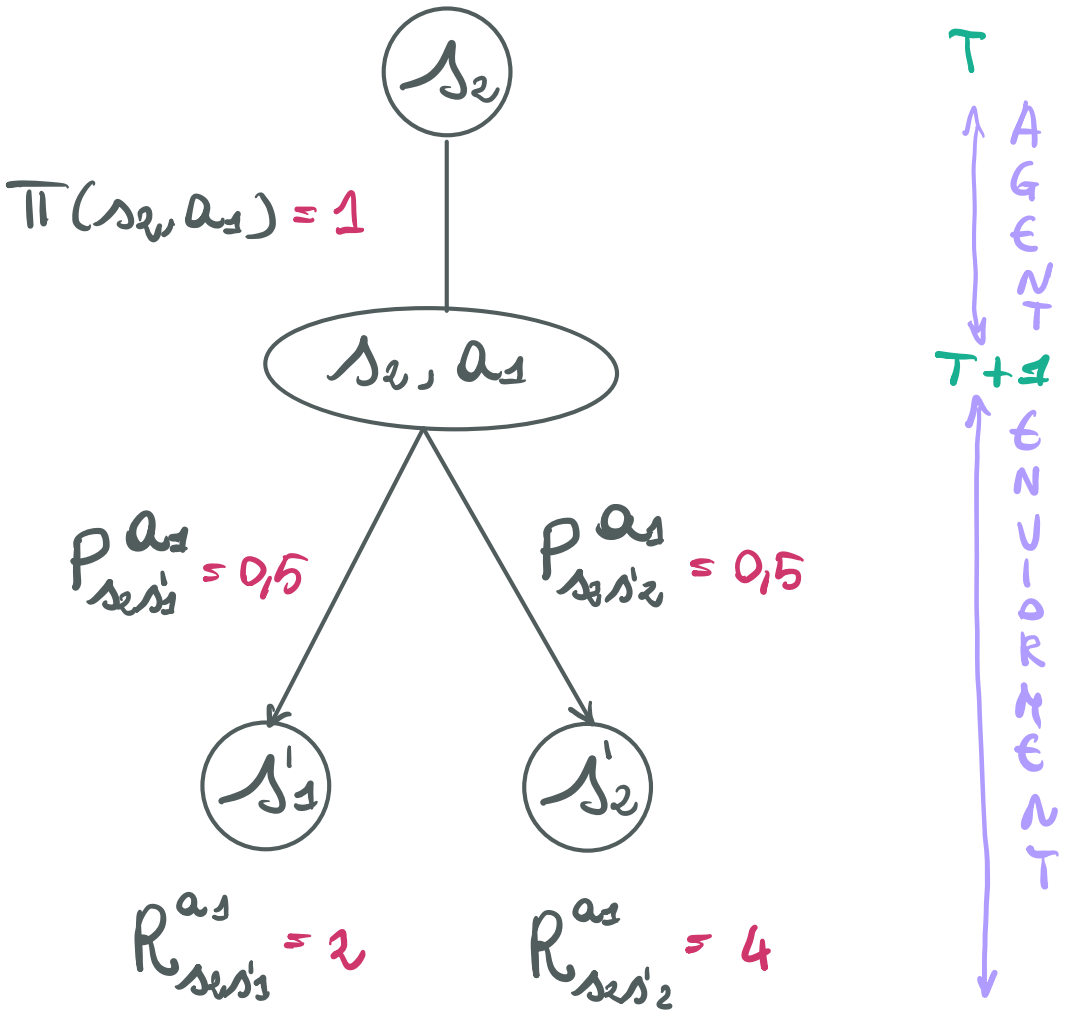
We will prove this equality through an **example**:

$$S = \{s_1, s_2\} \; ; \quad A = \{a_1, a_2\}$$

$$A(s_1) = \{a_1, a_2\}$$

$$A(s_2) = \{a_1\}$$


s_1
$\pi(s_1, a_1) = 0,5$
$\pi(s_1, a_2) = 0,5$
s_1, a_1
s_1, a_2
$P^{a_1}_{s_1 s'_1} = 0,6$
$P^{a_1}_{s_1 s'_2} = 0,4$
$P^{a_2}_{s_1 s'_2} = 1$
s'_1
s'_2
s'_2
$R^{a_1}_{s_1 s'_1} = 2$
$R^{a_1}_{s_1 s'_2} = 1$
$R^{a_2}_{s_1 s'_2} = 3$
T
AGENT
T+1
ENVIORMENT


s_2
$\pi(s_2, a_1) = 1$
s_2, a_1
$P^{a_1}_{s_2 s'_1} = 0,5$
$P^{a_1}_{s_2 s'_2} = 0,5$
s'_1
s'_2
$R^{a_1}_{s_2 s'_1} = 2$
$R^{a_1}_{s_2 s'_2} = 4$
T
AGENT
T+1
ENVIORMENT

$$V_\pi(s_1) = 0{,}5 \cdot 0{,}6\,[2 + \gamma V_\pi(s'_1)] +$$

$t \mapsto t+1$ $\quad$ $t+1 \mapsto a$

$$+ 0{,}5 \cdot 0{,}4\,[1 + \gamma V_\pi(s'_2)] +$$
$$+ 0{,}5 \cdot 1\,[3 + \gamma V_\pi(s'_2)]$$

$$V_\pi(s_2) = 1 \cdot 0{,}5\,[2 + \gamma V_\pi(s'_1)] +$$

$t \mapsto t+1$ $\quad$ $t+1 \mapsto a$

$$+ 1 \cdot 0{,}5\,[4 + \gamma V_\pi(s'_2)]$$

From this we get that the *intuition* of Bellman was that $V_\pi(s_i)$ are computed in a ricoursive way.

If we fix $\gamma = 0{,}9$

and consuider the state $V_\pi(s_1) = V_\pi(s'_1) = x$

$$V_\pi(s_2) = V_\pi(s'_2) = y$$

The equations become a linear state system:

$$x = (0{,}6 + 0{,}27x) + (0{,}2 + 0{,}18y) + (1{,}5 + 0{,}45y)$$
$$y = (1 + 0{,}45x) + 2(0{,}45y)$$

And after trivial computation

$$\begin{cases} x = 26{,}66 \\ y = 27{,}27 \end{cases}$$

But what does this mean? Let us start by computing the expected value of R_t

$$E_\pi[R_t] = \underbrace{26{,}66}_{V_\pi(s_1)} \cdot P_\pi(s_1 = s_1) + \underbrace{27{,}27}_{V_\pi(s_2)} P_\pi(s_2 = s_2)$$

$$V_\pi(s) = \sum_{a \in A(s)} \pi(a|s) \sum_{s'} P^a_{ss'} \left[R^a_{ss'} + \gamma V^\pi(s') \right]$$

$R^a_{ss'}$: reward $t \mapsto t+1$

$V^\pi(s')$: reward $t+1 \mapsto \infty$

So the Bellman equation actually is a system of $N = |S|$ equations in **N** unknowns for a *given* policy.

The value of the starting state is equal to the weighted sum of the values of possible arrival states plus the rewards associated to the transition from the starting states to the possible arrival ones. The weight of each possible arrival state is equal to the probability of arriving at such state from the generating state.

Are all policies optimal?

Consider a policy π' such that is considered *better* than π if

$\exists \pi > \pi'$

if $\forall s \quad V_\pi(s) \geq V_{\pi'}(s)$

We call π^* ***optimal policy*** if $\exists V_{\pi^*}(s) \geq V_\pi(s) \quad \forall s, \forall \pi$

Theorem: there always exist an optimal policy.

Before going forward with the speculation on what is an optima policy and how to find it, we will need to introduce another function which it is strictly linked to the state value of the policy $V_\pi(s)$ and we will use it for some algorithms that will be discussed layer on.

Queu Function.

$Q_\pi(s,a)$: *action-value function for the policy* π

It is the long term return when taking the action "**a**" in the state "**s**" and following the policy π there after.

$$= E_\pi \{ R_t \mid s_t = s ; a_t = a \}$$

$$= E_\pi \left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s ; a_t = a \right\}$$

Compute now the state value function for the policy

$$V_\pi = E_\pi \{ R_t \mid s_t = s \}$$

$$= \sum_{a \in A(s)} E_\pi \{ R_t \mid s_t = s ; a_t = a \} \, P_r \{ a_t = a \mid s_t = s \}$$

$$= \sum_{a \in A(s)} Q_\pi(s,a) \, \pi(s,a)$$

This equation above represents the relationship between

$V_\pi(s)$ and $Q_\pi(s)$

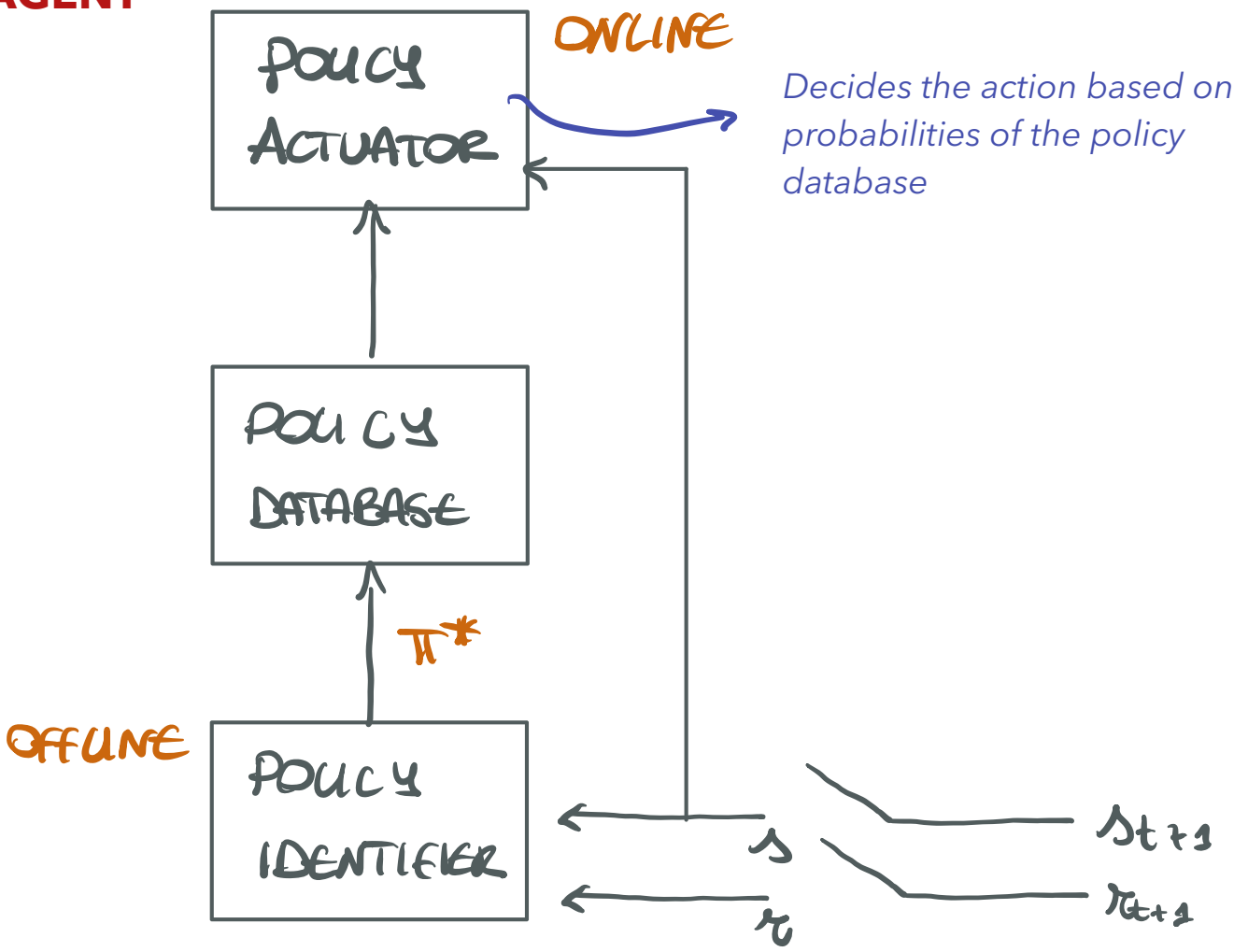
Optimal Bellman Equation.

$$V_{\pi^*}(s) = \max_{a \in A(s)} \sum_{s' \in S} P^a_{ss'} \left[R^a_{ss'} + \gamma V_\pi(s') \right]$$

Which is a system of non linear **N = |S|** equations in **N = |S|** unknowns .

At this point we can comment a little bit better how the agent is structured:

THE AGENT



In **conclusion** we have just explained how from the state value function for the policy we get the optimal policy:

$$\text{from} \quad V_{\pi^*}(s) \longmapsto \pi^*(s;a)$$

recap:

Giving the expectation long term reward $E_\pi[R_\pi] = E_\pi\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1}\right]$ *with* $0 \leq \gamma \leq 1$ *it is possible to find the optimal policy* π^* *through maximisation of every reward up to the future which, by the way, is the key concept of reinforcement learning.*

$$E_\pi[R_\pi] = E_\pi\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1}\right]$$

$$= E_\pi\left[V_\pi(s) P_r\{s_t = s\}\right]$$

initial condition

$$\exists \; V_{\pi^*}(s) \geq V_\pi(s) \quad \forall \pi, s$$

THIS IS WHAT WE need TO FIND, IN ORDER TO FIND π^*

***Please note**, Bellman equation is what we use to find a generic policy.*

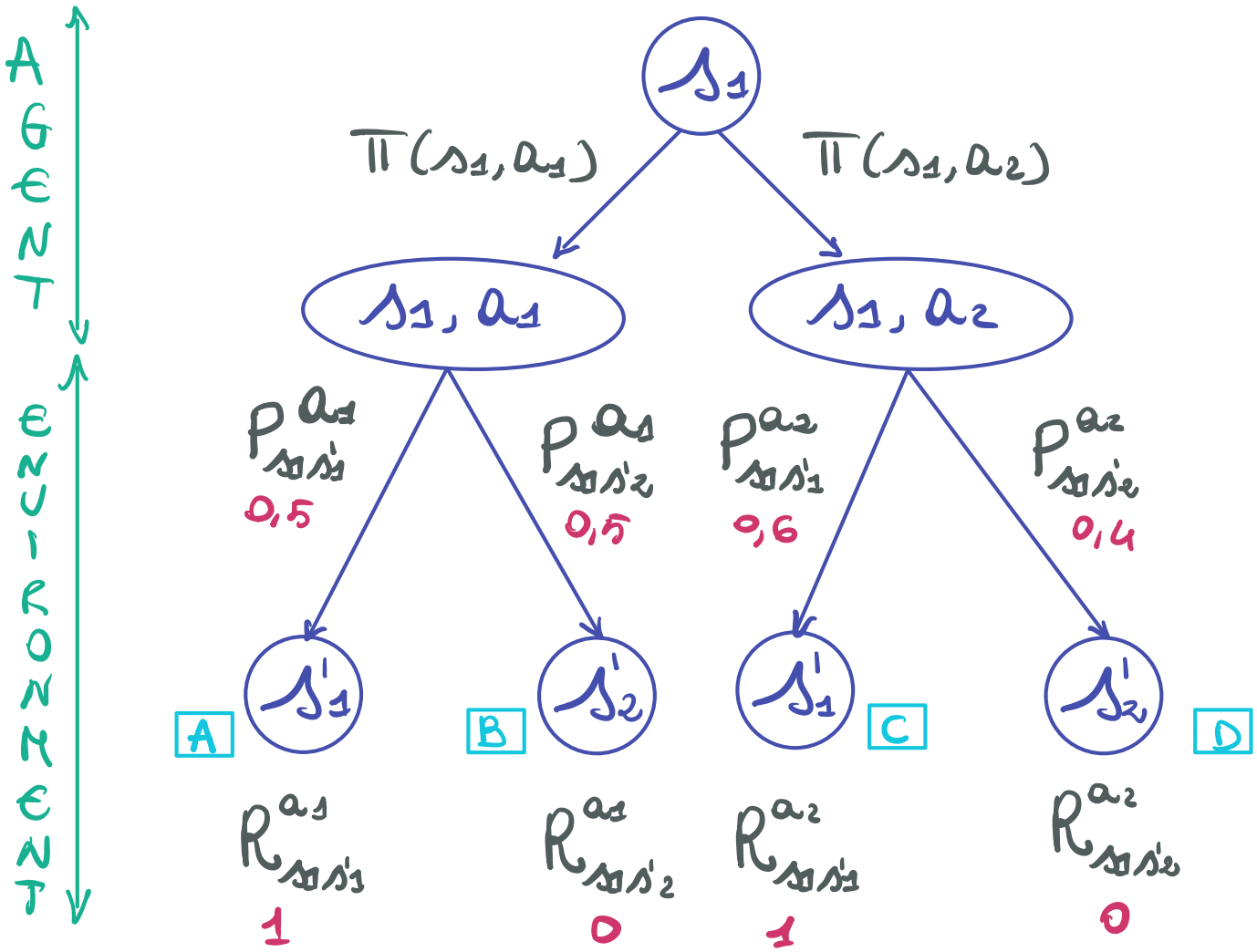
To find an optimal policy we need use the Optimal Bellman Equation:

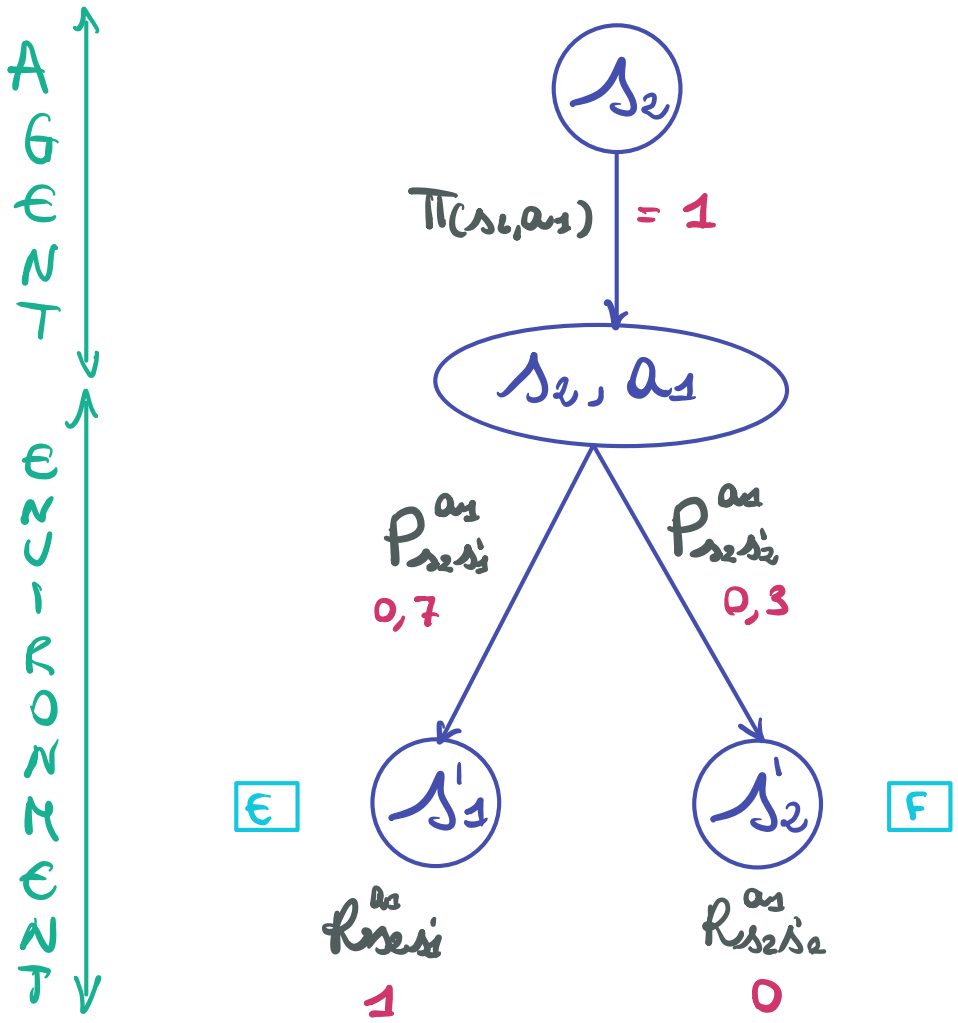
$$V_{\pi^*}(s) = \max_{a \in A(s)} \sum_{s' \in S} P^a_{ss'} \left[R^a_{ss'} + \gamma \, V_\pi(s') \right]$$

Now, to understand bettere what the Bellman equations are saying to us we will proceed with an **example**:

$$S = \{s_1, s_2\} \qquad A = \{a_1, a_2\}$$

$$A(s_1) = \{a_1, a_2\} \qquad A(s_2) = \{a_1\}$$





For this example it is easy to find that the optimal action which gives us the optimal arrival states, for s_1 , is a_2 ; of couse since the state s_2 has just an action the optimal action will be a_1 .

We will get now the same result through algebraic steps.

First we need to assume that $\gamma = 0,9$ and also that we know the optimal policy's arrival states for each path

A $V_{\pi^*}(s'_1) = 10$

B $V_{\pi^*}(s'_2) = 5$

C $V_{\pi^*}(s'_1) = 10$

D $V_{\pi^*}(s'_2) = 5$

Let us now compute the optimal policy state for s_1

$$V_{\pi^*}(s_1) = \max_{a_1, a_2} \left\{ \begin{array}{l} P^{a_1}_{s_1 s_1}\left[R^{a_1}_{s_1 s_1} + \gamma\, V_{\pi^*}(s_1)\right] \\ \quad + P^{a_1}_{s_1 s_2}\left[R^{a_1}_{s_1 s_2} + \gamma\, V_{\pi^*}(s_2)\right] \\ P^{a_2}_{s_1 s_1}\left[R^{a_2}_{s_1 s_1} + \gamma\, V_{\pi^*}(s_1)\right] \\ \quad + P^{a_2}_{s_1 s_2}\left[R^{a_2}_{s_1 s_2} + \gamma\, V_{\pi^*}(s_2)\right] \end{array} \right.$$

$V_{\pi^*}(s_1)$ → a_1: 9; $4{,}5$ — a_2: 9; $4{,}5$

$$= \max_{a_1, a_2} \left\{ \begin{array}{l} 0{,}5[1+9] + 0{,}5[0+4{,}5] \\ 0{,}6[1+9] + 0{,}4[0+4{,}5] \end{array} \right.$$

$$= \max_{a_1, a_2} \left\{ 5 + 2{,}25 \;;\; 6 + 1{,}8 \right\}$$

$$= \max_{a_1, a_2} \left\{ \underset{a_1}{7{,}25} \;;\; 7{,}8 \right\}$$

WINNER a_2

To get a more direct result it is needed to define **the optimal action**:

$$a^* = \underset{a \in A(s)}{\operatorname{argmax}} \sum P^{a}_{ss'}\left[R^{a}_{ss'} + \gamma\, V_{\pi}(s')\right]$$

So in our case we obtain:

$$a^*(s_1) = \underset{a_1, a_2}{\operatorname{argmax}} \left\{ \underset{a_1}{7{,}25} \;;\; \underset{a_2}{7{,}4} \right\} = a_2$$

As we already said for the state s_2 the optimal action is a_1 because it is the only one that it is able to perform.

$$V_{\pi^*}(s_2) = P^{a_1}_{s_2 s_1}\left[R^{a_1}_{s_2 s_1} + \gamma\, V^*_{\pi}(s_1)\right]$$

$$+ P^{a_2}_{s_2 s'_2}\left[R^{a_2}_{s_2 s'_2} + \gamma V_\pi^*(s'_2)\right]$$
$$= 0{,}7\left[1 + 0{,}9\, V_\pi^*(s'_1)\right] + 0{,}3\left[0 + 0{,}9\, V_\pi^*(s'_2)\right]$$

Now, according to Bellman reasoning: $V_\pi^*(s) = V_\pi^*(s')$

Follows that we need to solve a system of N = |S| *cardinality of the states set* linear equations in N = |S| unknowns.

$$V_\pi^*(s_1) = V_\pi^*(s'_1) = x_1$$
$$V_\pi^*(s_2) = V_\pi^*(s'_2) = x_2$$

And we solve the system of equations:

$$\begin{cases} x_1 = \max\left\{ 0{,}5\,(1 + 0{,}9\,x_1) + 0{,}5\,(0 + 0{,}9\,x_2)\,;\; 0{,}6\,(1 + 0{,}9\,x_1) + 0{,}4\,(0 + 0{,}9\,x_2) \right\} \\ x_2 = 0{,}7\,(1 + 0{,}9\,x_1) + 0{,}3\,(0 + 0{,}9\,x_2) \end{cases}$$

$$\begin{cases} x_1 = \max\left\{ 0{,}45\,x_1 + 0{,}45\,x_2 + 0{,}5\,;\; 0{,}54\,x_1 + 0{,}36\,x_2 + 0{,}6 \right\} \\ x_2 = 0{,}63\,x_1 + 0{,}27\,x_2 + 0{,}7 \end{cases}$$

Let us proceed by trial and errors:

Rp: a_1 WINS $a_1 > a_2$

$$\begin{cases} x_1 = 0{,}45\,x_1 + 0{,}45\,x_2 + 0{,}5 \\ x_2 = 0{,}63\,x_1 + 0{,}27\,x_2 + 0{,}7 \end{cases}$$

$$\begin{cases} x_1 = 5{,}76 \\ x_2 = 5{,}93 \end{cases}$$

Rp: a_2 WINS $a_2 > a_1$

$$\begin{cases} x_1 = 0{,}54\,x_1 + 0{,}36\,x_2 + 0{,}6 \\ x_2 = 0{,}63\,x_1 + 0{,}27\,x_2 + 0{,}7 \end{cases}$$

$$\begin{cases} x_1 = 6{,}33 \\ x_2 = 6{,}42 \end{cases}$$

THIS IS THE BEST

$\pi^*(s,a)$	a_1	a_2
s_1	0	1
s_2	1	X

Please note, this method defines deterministic policies.

The optimality of a policy is not true just at time **t** but also at time **t+1** $\longrightarrow \infty$

This means that once we have found an optimal policy, this stays optimal even for the future steps.

Quoting from Sutton and Burton's:

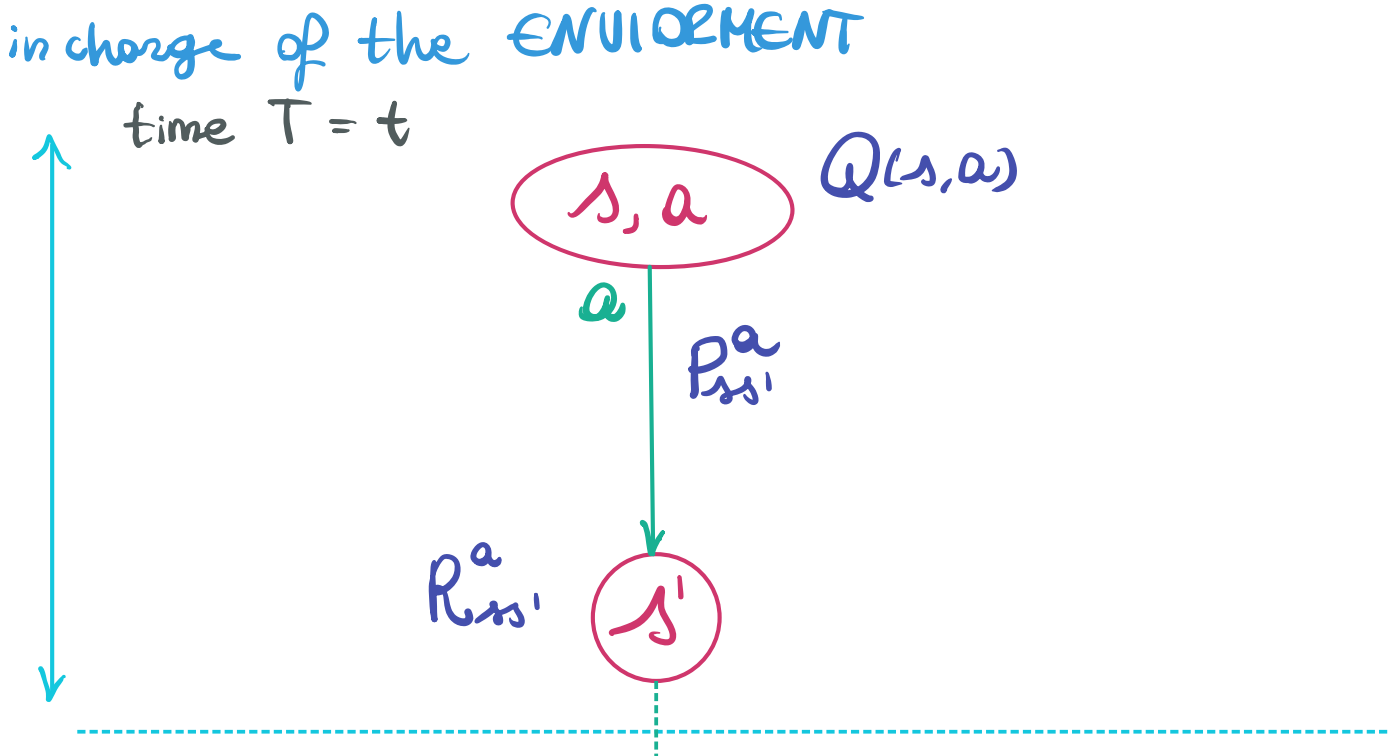
"Once one has v_*, it is relatively easy to determine an optimal policy. For each state s, there will be one or more actions at which the maximum is ob- tained in the Bellman optimality equation. Any policy that assigns nonzero probability only to these actions is an optimal policy. You can think of this as a one-step search. If you have the optimal value function, v_*, then the actions that appear best after a one-step search will be optimal actions. An- other way of saying this is that any policy that is greedy with respect to the optimal evaluation function v_* is an optimal policy. The term greedy is used in computer science to describe any search or decision procedure that selects alternatives

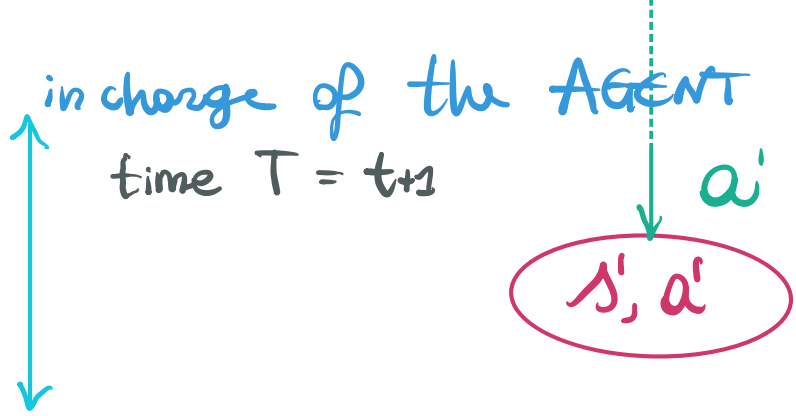
based only on local or immediate considerations, without considering the possibility that such a selection may prevent future access to even better alternatives. Consequently, it describes policies that select actions based only on their short-term consequences. The beauty of v_* is that if one uses it to evaluate the short-term consequences of actions—specifically, the one-step consequences—then a greedy policy is actually optimal in the long- term sense in which we are interested because v_* already takes into account the reward consequences of all possible future behavior. By means of v_*, the optimal expected long-term return is turned into a quantity that is locally and immediately available for each state. Hence, a one-step-ahead search yields the long-term optimal actions."

It is possible to get a formulation of the Optimal Bellman equation using the **Q-function**:

$$Q_{\pi^*}(s,a) = \sum_{s' \in S} P^a_{ss'} \left[R^a_{ss'} + \gamma \max_{a' \in A(s')} Q_{\pi^*}(s',a') \right]$$

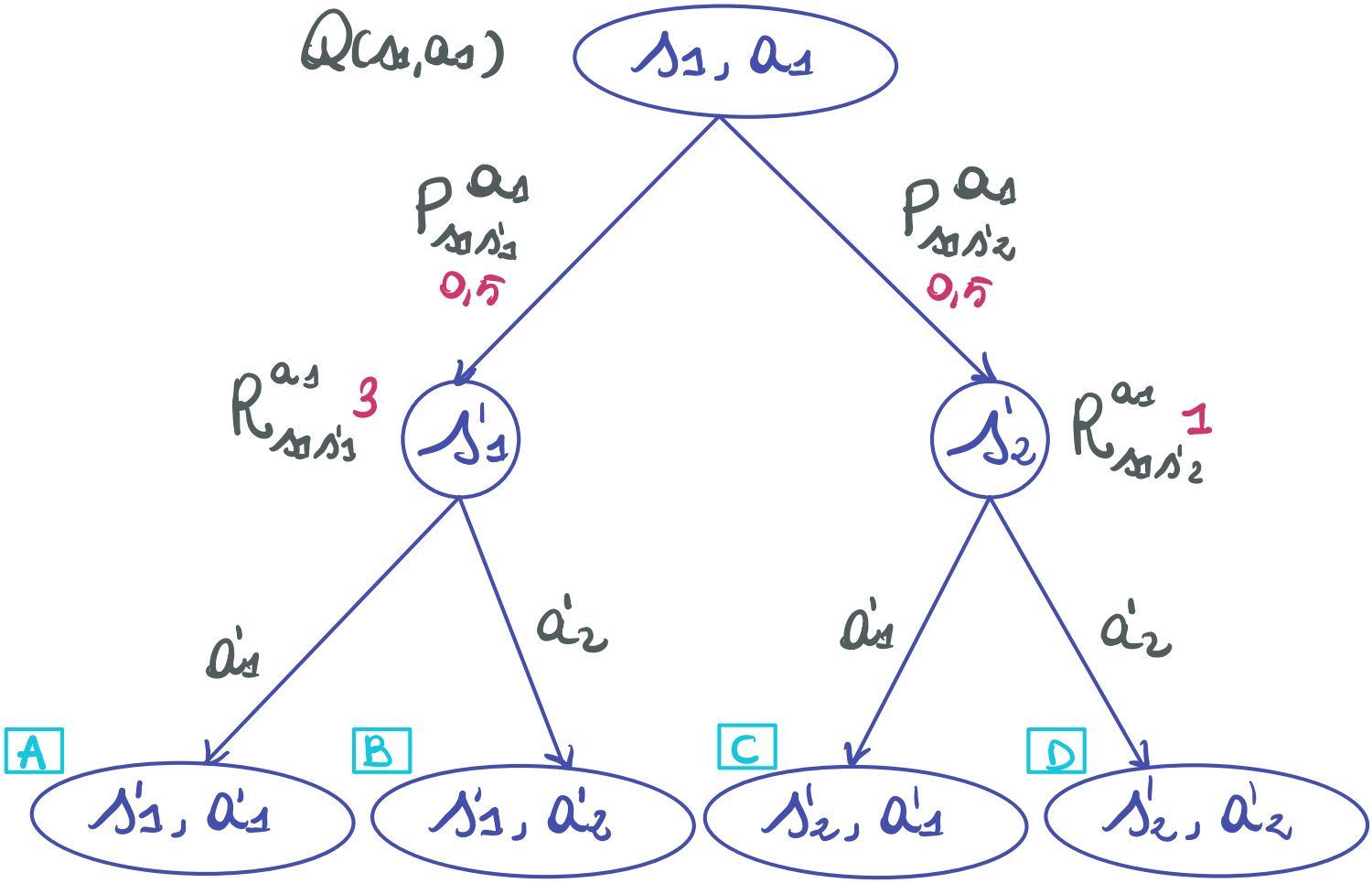
Before discussing the meaning of this equation it is important to notice that if we refer to a Q-function the back-up diagram is different





In this prospective $Q(s,a)$ rapresents a value of the (s,a) couple

Let us make, once again, an **example**:



We now assume, as before, that ve know optimal values of the Q-function of each different path:

A $Q_{\pi^*}(s'_1, a'_1) = 2$

B $Q_{\pi^*}(s'_1, a'_2) = 3$

C $Q_{\pi^*}(s'_2, a'_1) = 4$

D $Q_{\pi^*}(s'_2, a'_2) = 5$

Suppose also that $\gamma = 0,9$

Now we are going to do similar computations as we did before with the optimal policy state function and exactly as before we can intuitively say that the optimal solution is B.

Let us now see why through computations.

A. $[R^{a_1}_{s_1 s'_1} + \gamma Q_{\pi^*}(s'_1, a'_1)] = 3 + 0,9 \cdot 2 = 4,8$

B. $[R^{a_1}_{s_1 s'_1} + \gamma Q_{\pi^*}(s'_1, a'_2)] = 3 + 0,9 \cdot 3 = 5,7$

C. $[R^{a_1}_{s_1 s'_2} + \gamma Q_{\pi^*}(s'_2, a'_1)] \quad 1 + 0,9 \cdot 4 = 4,6$

D. $[R^{a_1}_{s_1 s'_2} + \gamma Q_{\pi^*}(s'_2, a'_2)] \quad 1 + 0,9 \cdot 5 = 5,5$

$\hookrightarrow t+1 \mapsto \infty$

Once again exactly as before it is possible to solve a system of **N = |S| |A|** *cardinality of the states set by the cardinality of the actions set* non-linear equations in **N = |S| |A|** unknowns.

We will get in fact the following table:

$Q_{\pi^*}(s,a)$	a_1	a_2
s_1	24	52
s_2	30	18

Which corresponds to the policy's table

$\pi^*(s,a)$	a_1	a_2
s_1	0	1
s_2	1	X

Optimal Policy Evaluation.

This example will be now useful to hep us to find a much smarter way to compute the optimal policy through 2 methods.

1st Method: Policy Iteration.

$$\pi \longmapsto V_\pi \longmapsto \pi' \longmapsto V_{\pi'} \longmapsto \pi'' \longmapsto V_{\pi''} \longmapsto \cdots \longmapsto \pi^*$$

The Algorithm.

Step 1. Evaluation.

Given a policy we compute the correspondent policy state.

$$\pi \longmapsto V_\pi$$

Step 2. Improvement.

Given the policy state perviously computed we find a better policy.

$$V_\pi \longmapsto \pi' \quad \ni \quad \pi' > \pi$$

Step 3. Evaluation.

Now with the new policy the new policy state will be computed.

$$\pi' \longmapsto V_{\pi'}$$

Last Step. Improvement.

The iteration will go on like this untill the policy at the generic step is

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

and the policy state at the generic step is

$$V_{\pi^*} = V_{\bar{\pi}}$$

Notice that it can be proven that the algorithm always converges to a solution $\exists\, \bar{\pi} \longmapsto \pi^*$

How to perform the Algorithm.

Generic **Policy Evaluation Step** or *prediction problem*.

We can of course use the Bellman equation system but when we need to evaluate billions and billions of states it will be not efficient; so we will use another method.

The Bellman euqation will be defined ad an update rule which will help us to build more efficient iterative set of computations.

For the **generic step k=0,1,...**:

$$V_{k+1}^{\pi}(s) := \sum_{a \in A(s)} \pi(s,a) \sum_{s' \in S} P_{ss'}^{a} \left[R_{ss'}^{a} + \gamma V_{k}^{\pi}(s) \right]$$

assignement operator

Notice that it can be shown that for an arvitrary V_0 when

$k \mapsto \infty$ the sequence $\{V_k\} \mapsto V_\pi$

Let us now apply this iterational method to our **example**:

$k = -1$ initialisation

$V_0(s_1) = V_0(s_2) = 0$

$V_0''(s_1')$ $V_0''(s_2')$

$k = 0$

$$V_1^{\pi}(s_1) := 0{,}5 \{ 0{,}5\,(1 + 0{,}9 \cdot 0) + 0{,}5\,(0 + 0{,}9 \cdot 0) \{ + 0{,}5 \{ 0{,}6\,(1 + 0{,}9 \cdot 0) + 0{,}4\,(0 + 0{,}9 \cdot 0) \{ = 0{,}55$$

$$V_1^{\pi}(s_2) := 1\,(0{,}7 + 0{,}9 \cdot V_?(s_1'))$$

$V_0(s_1')$ $V_1(s_1')$

$= 0$ $= 0{,}55$

What is happening here?
We are faced with a choice, we can use two methods: the ***inplace method*** *which will bring us to use the value of the state at step k or the* ***outplace mathod*** *which will bring us to use the value that we have just computed at step k+1.*
For sure every method has their pro and cons but on an efficiency level they are mostly the same but we prefer to use the outplace method.

$$+ 1\,(0{,}3\,(0 + 0{,}3\,V_0(s_2))) = 1{,}085$$

Visually we can upload a table with the policy state values for each step

	$V(s_1)$	$V(s_2)$
$k=-1$	0	0
$k=0$	0,55	1,085

Notice that to avoid the so called *infinite iteration* we formally define a formal upperbound $\Delta \ni$ when

$$\left| V^{\pi}_{k+1}(s) - V^{\pi}_{k}(s) \right| < \Delta$$

The iterations will stop.

Recap: *To produce each successive approximation, vk+1 from vk, iterative policy evaluation applies the same operation to each state s: it replaces the old value of s with a new value obtained from the old values of the successor states of s, and the expected immediate rewards, along all the one-step transitions possible under the policy being evaluated. We call this kind of operation a* ***full backup****. Each iteration of iterative policy evaluation backs up the value of every state once to produce the new approximate value function vk+1. To write a sequential computer program to implement iterative policy evaluation will be enough to follow the pseudocode below:*

Input π, the policy to be evaluated
Initialize an array $V(s) = 0$, for all $s \in \mathcal{S}^+$
Repeat
  $\Delta \leftarrow 0$
  For each $s \in \mathcal{S}$:
    $v \leftarrow V(s)$
    $V(s) \leftarrow \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)\left[r + \gamma V(s')\right]$
    $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$ (a small positive number)
Output $V \approx v_\pi$

Generic **Policy Improvement Step**.

We already know that a generic policy is better then another when

$$\exists \quad \bar{\pi}' > \bar{\pi} \quad \ni \quad V_{\bar{\pi}'} > V_{\bar{\pi}}$$

How to proceed?

Suppose we have determined the value of the policy state function for an arbitrary deterministic policy. For some state s we would like to know whether or not we should change the policy to deterministically choose an action a that correspond to that policy at that state s. Thanks to the *evaluation step* we already know how good it is to follow the current policy from s; would it be better or worse to change to the new policy?

One way to answer this question is to consider to select a certain action a for the state s and thereafter following the existing policy.

This way of behaving can be evalueted with the Q-function for a generic policy:

$$Q_{\bar{\pi}}(s,a) = \sum_{s' \in S} P^a_{ss'} \left[R^a_{ss'} + \gamma V_{\bar{\pi}}(s') \right]$$

The key criterion is whether this is greater than or less than the policy state function. If it is greater, we need to decide if it is better to select an action a once in s and thereafter we will follow the policy that would be to followed all the time. Then one would expect it to be better still to select an action a every time s is encountered, and that the new policy would in fact be a better one overall. This result is called **Policy Improvement Theorem**:

given a pair of deterministic generic policies $\bar{\pi}, \bar{\pi}'$

$$\forall s \in S \quad Q_{\pi}(s, \pi'(s)) \geq V_{\pi}(s)$$

$$\Rightarrow \bar{\pi}' \geq \bar{\pi}$$

$$\Rightarrow V_{\bar{\pi}'} \geq V_{\bar{\pi}}$$

We have seen how, given a policy and its value function, we can easily evaluate a change in the policy at a single state to a particular action. It is a natural extension to consider changes at all states and to all possible actions, selecting at each state the action that appears best according to the Q-function. In other words, to consider the new *greedy policy* at the general choosen action given by:

PREFERRED ACTION

$$\bar{a}(s) = \underset{a \in A(s)}{\operatorname{argmax}} \; Q_{\bar{\pi}}(s, a)$$

$$= \underset{a \in A(s)}{\operatorname{argmax}} \sum_{s' \in S} P^{a}_{ss'} \left[R^{a}_{ss'} + \gamma V_{\bar{\pi}}(s') \right]$$

$$\exists \forall s \in S$$

$$\Rightarrow \bar{\pi}'(s, \bar{a}(s)) = 1$$

$$\bar{\pi}'(s, a(s)) = 0 \quad \forall \, a(s) \neq \bar{a}(s)$$

The greedy policy takes the action that looks best in the short term—after one step of lookahead—according to the policy state function. By construction, the greedy policy meets the conditions of the policy improvement theorem, so we know that it is as good as, or better than, the original policy.

The process of making a new policy that improves on an original policy, by making it greedy with respect to the value function of the original policy, is

called policy improvement.

In our **example:**

$$\bar{a}(s_1) = \underset{a_1, a_2}{\operatorname{argmax}} \{ Q_\pi(s_1, a_1); Q_\pi(s_1, a_2) \}$$

$$= \underset{a_1, a_2}{\operatorname{argmax}} \{ \dots \} = a_2$$

$$\Rightarrow \pi'(s_1, a_2) = 1$$

It can happen that the new *greedy policy* has the same value just as the pervious one: *what does it mean?* It just means that we hav found our **optimal policy** and the iteration can stop.

Recap: full algorithm pseudocode.

1. Initialization
   $V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

2. Policy Evaluation
   Repeat
   $\Delta \leftarrow 0$
   For each $s \in \mathcal{S}$:
   $v \leftarrow V(s)$
   $V(s) \leftarrow \sum_{s',r} p(s', r|s, \pi(s)) [r + \gamma V(s')]$
   $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
   until $\Delta < \theta$ (a small positive number)

3. Policy Improvement
   policy-stable $\leftarrow$ *true*
   For each $s \in \mathcal{S}$:
   $a \leftarrow \pi(s)$
   $\pi(s) \leftarrow \operatorname{arg\,max}_a \sum_{s',r} p(s', r|s, a) [r + \gamma V(s')]$
   If $a \neq \pi(s)$, then *policy-stable* $\leftarrow$ *false*
   If *policy-stable*, then stop and return V and π; else go to 2

One drawback to policy iteration is that each of its iterations involves policy evaluation, which may itself be a protracted iterative computation requiring multiple sweeps through the state set. If policy evaluation is done iteratively, then convergence exactly to vπ occurs only in the limit. Must we wait for exact convergence, or can we stop short of that? Yes, it may be possible to truncate policy evaluation. Policy evaluation iterations can have no effect on the corresponding greedy policy.

In fact, the policy evaluation step of policy iteration can be truncated in several ways without losing the convergence guarantees of policy iteration. One important special case is when policy evaluation is stopped after just one sweep (one backup of each state). This algorithm is called value iteration.

2nd Method: Value Iteration.

It can be written as a particularly simple backup operation that combines thepolicy improvement and truncated policy evaluation steps:

$$\forall s \in S$$

$$V_{k+1}^{\pi}(s) := \max_{a \in A(s)} \sum_{s' \in S} P_{ss'}^{a} \left[R_{ss'}^{a} + \gamma V_{\pi}(s') \right]$$

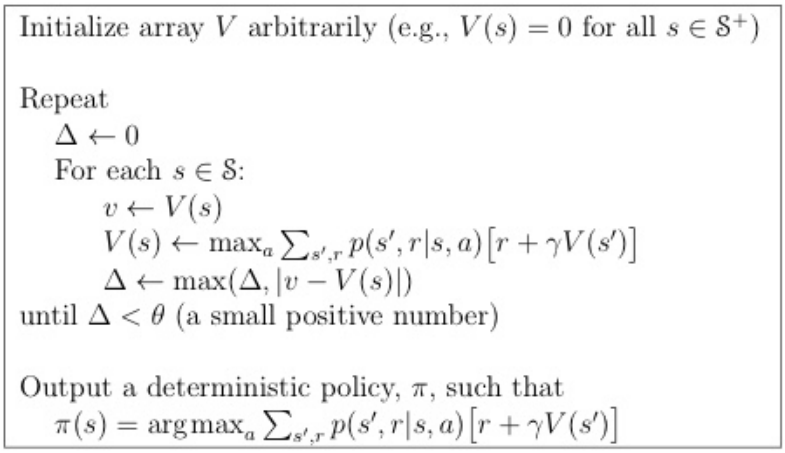
Notice that also in this case in can be shown that for an arbitrary V_0 when

$$k \mapsto \infty \quad \{V_k\} \mapsto V_{\pi^*}$$

It is always guaranteed the existance of an optimal policy state function.

Finally, let us consider how value iteration **terminates**. Like policy evaluation, value iteration formally requires an infinite number of iterations to converge

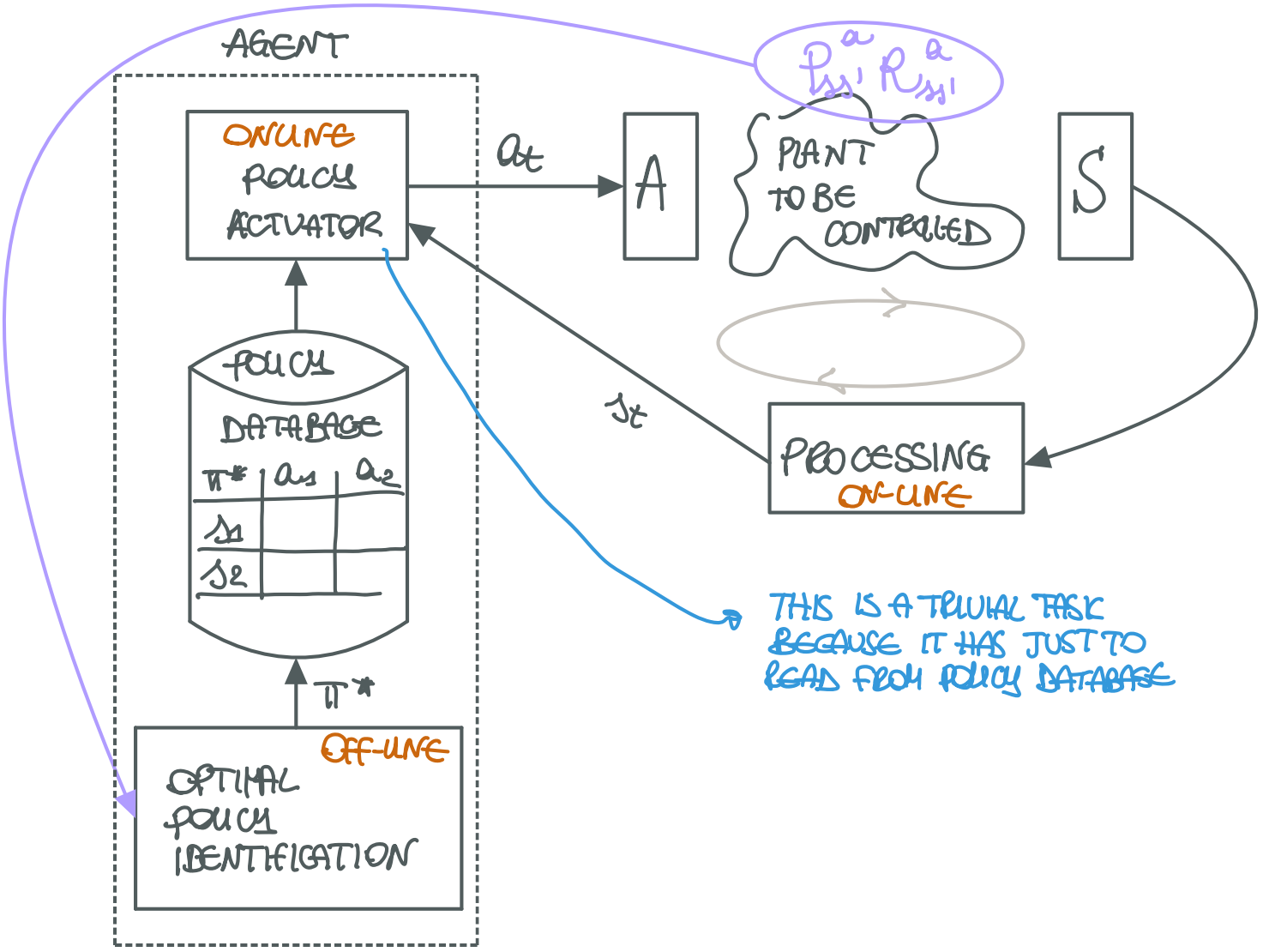
exactly to v∗. In practice, we stop once the value function changes by only a small amount in a sweep.



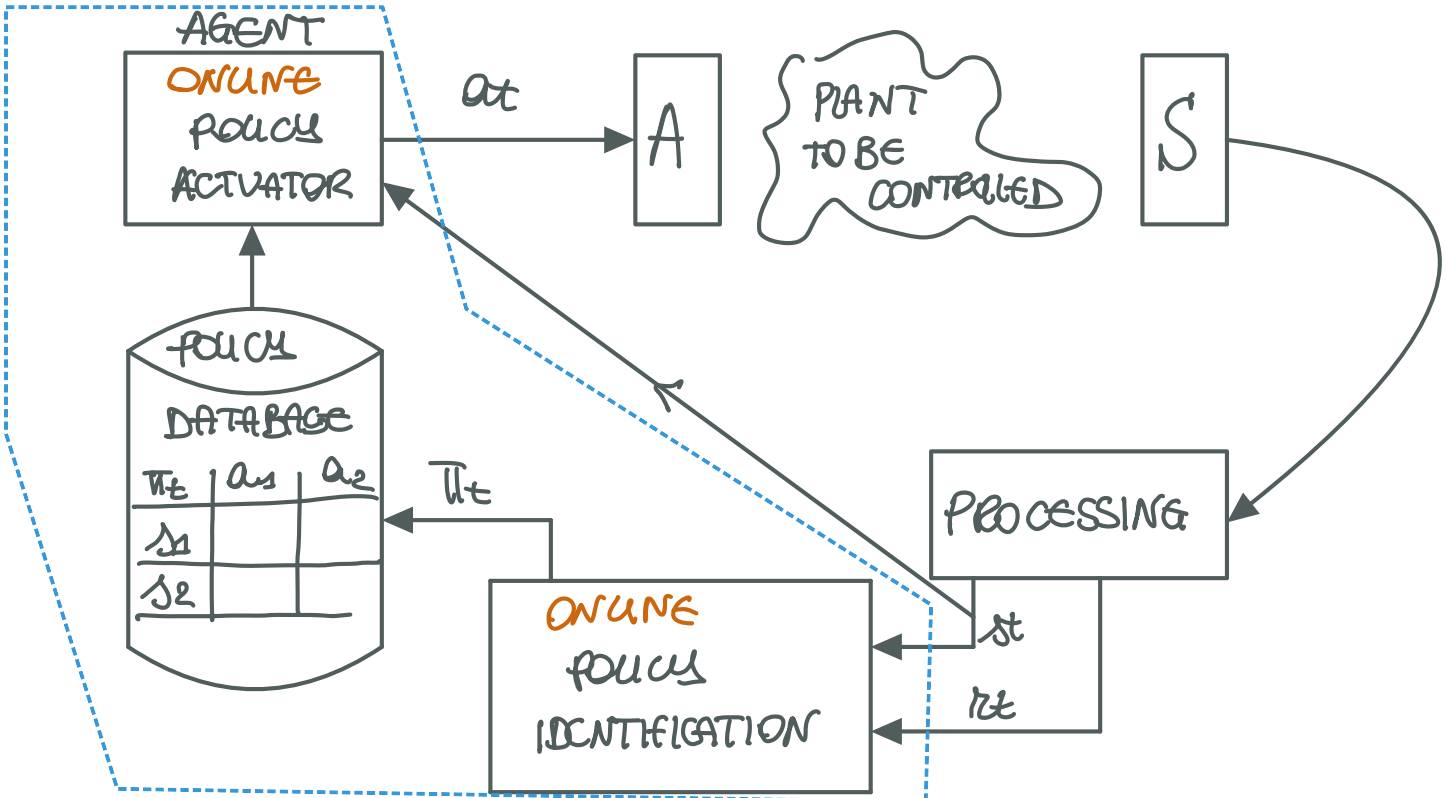
Value iteration effectively combines, in each of its sweeps, one sweep of policy evaluation and one sweep of policy improvement. Faster convergence is often achieved by interposing multiple policy evaluation sweeps between each policy improvement sweep. In general, the entire class of truncated policy iteration algorithms can be thought of as sequences of sweeps, some of which use policy evaluation backups and some of which use value iteration backups. Since the max operation in our update rule is the only difference between these backups, this just means that the max operation is added to some sweeps of policy evaluation. All of these algorithms converge to an optimal policy for discounted finite MDPs.

Please note that both these methods are part of the so called Dynamic Programming.

Given the known control scheme with MDP agent



The focus of the following



Difference between the schemes is that there is no oprimal policy but just a policy at time t **PIt**. The plant is a block box and the policy identification is also online, follows that $\Pi \mapsto \Pi^*$ when $t \mapsto \infty$

because the approximation of **PI** ets better and better.

Remember the Bellman equation tuned on the updated function.

$$V_{k+1}(s) := \sum_{a \in A(s)} \pi(s,a) \sum_{s' \in S'} P_{ss'}^{a} \left[R_{ss'}^{a} + \gamma V_k(s') \right]$$

Consider the average values of Ak,avg and Ak+1,avg.

$$\bar{A}_k = \frac{a_1 + a_2 + a_3 + \dots + a_k}{k} \qquad (k+1) - k$$

$$\bar{A}_{k+1} = \frac{a_1 + a_2 + a_3 + \dots + a_k}{k+1} = \frac{k}{k+1} \frac{(a_1 + a_2 + \dots + a_k)}{k} + \frac{a_{k+1}}{k+1}$$

$$= \frac{k}{k+1} \bar{A}_k + \frac{a_{k+1}}{k+1} = \frac{1}{k+1} \left(k \bar{A}_k + a_{k+1} + \bar{A}_k - \bar{A}_k \right)$$

$$= \frac{1}{k+1} \left[a_{k+1} + (k+1) \bar{A}_k - \bar{A}_k \right] = \bar{A}_k + \frac{1}{k+1} \left(a_{k+1} - \bar{A}_k \right)$$

exercise:

$k = 2$

samples $a_1 = 28$, $a_2 = 30$

$$\bar{A}_k = \frac{28+30}{2} = 29$$

add now $a_3 = 26$

$$\bar{A}_{k+1} = \frac{26+28+30}{3} = 28$$

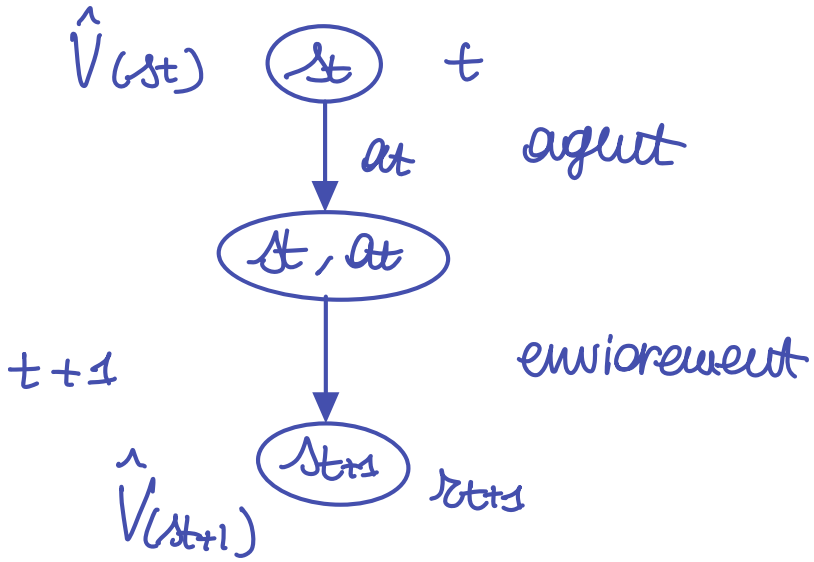
it is the same as

$$\bar{A}_{k+1} = 29 + \frac{1}{3}(26 - 29)$$

RECAP NEW ESTIMATION = OLD ESTIMATION + STEPSIZE (a_{k+1} TARGET − OLD ESTIMATE)

this way it is possible to move average

Stocastic process identifies V(s) at the last state st+1 with rt+1 using the Bellman concept.



$$V(s) = r_{t+1} + \gamma V(s_{t+1})$$

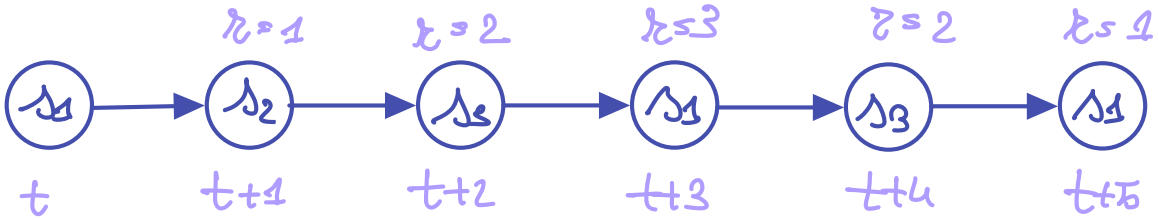
Let translate this concept in the "mean" formula:

$$\text{NEW ESTIMATION} = \text{OLD ESTIMATION} + \text{STEP SIZE}\left(\text{TARGET}^{a_{k+1}} - \text{OLD ESTIMATE}\right)$$

$$V(s_t) := V(s_t) + \alpha_k(s_t)\left(r_{t+1} + \gamma V(s_{t+1}) - V(s_t)\right)$$

We are updating the new estimate at state st which will be more refined cause we will know new information estimate on the state st+1. This way we will et a vector that stores all the V(s) vallues for each state while we are updating only one state.

example:



	$V(s_1)$	$V(s_2)$	$V(s_3)$
t	~~0~~	0	0
t+1	1	~~0~~	0
t+2	1	1	~~0~~
t+3	~~1~~	1	2/3
t+4	1,56	1	~~2/3~~
t+5	1,56	1	0,61

suppose $\alpha_t = \frac{1}{t+1}$

$\gamma = 0,9$

$$V(s_1) := \underbrace{V(s_1)}_{=0} + 1\left[1 + 0,9\,\underbrace{V(s_2)}_{=0} - \underbrace{V(s_1)}_{=0}\right] \qquad \gamma^{(t+1)}$$

$V(s_1) = 1$

$$V(s_2) := \underbrace{V(s_2)}_{=0} + 1/2\left[2 + 0,9\,\underbrace{V(s_3)}_{=0} - \underbrace{V(s_2)}_{=0}\right] \qquad \gamma^{(t+1)}$$

$V(s_2) := 1$

$$V(s_3) := \underbrace{V(s_3)}_{=0} + 1/3\left[3 + 0,9\,\underbrace{V(s_3)}_{=0} - \underbrace{V(s_1)}_{=1}\right] \qquad \gamma^{(t+1)}$$

$V(s_3) := 2/3$

$$V(s_1) := \underbrace{V(s_1)}_{=1} + 1/4\left[2 + 0,9\,\underbrace{V(s_1)}_{1} - \underbrace{V(s_3)}_{2/3 = 0,6}\right] \qquad \gamma^{(t+1)}$$

$V(s_1) := 1,575$

$$V(s_3) := \underbrace{V(s_3)}_{2/3} + 1/5\left[1 + 0,9\,\underbrace{V(s_3)}_{2/3} - \underbrace{V(s_1)}_{1,56}\right] \qquad \gamma^{(t+1)}$$

$V(s_3) := 0,61$

This kind of method introduces TDL - Temporal Differential Learning, SARSA learning and Q-Learning which are all three based on

RANDOM INITIALISATION $V(s)$
RANDOM SELECT $\pi(s,a)$
RANDOM SELECT initial state
$\mapsto$ repeat

Policy data base evolves in time !

π_t	a_1	a_2
s_1		
s_2		

1. Perform the action a_t derived from the current policy π
2. assignement of $V(s_t) := V(s_t) + \alpha_k \left(r_{t+1} + \gamma V(s_{t+1}) - V(s_t) \right)$

CRITIC $\alpha_k = \frac{1}{k+1}$

3. improve policy π deriving a policy π' — ACTOR
4. $s_t := s_{t+1}$, $\pi := \pi'$

Focus on actor improvement.

TD ERROR

$$p(s_t, a_t) = p(s_t, a_t) + \beta_k(s_t) \left[r_{t+1} + \gamma V(s_{t+1}) - V(s_t) \right]$$

NEW SAMPLE

- $p(s_t, a_t)$: future probability of state s_t exactly the same mechanism as before
- $p(s_t, a_t)$: OLD ESTIMATE
- $\beta_k(s_t) > 0$: way the probability moves

it is a PSEUDOPROBABILITY to perform action a_t at state s_t

(PN) if error negative then new estimation < old estimation

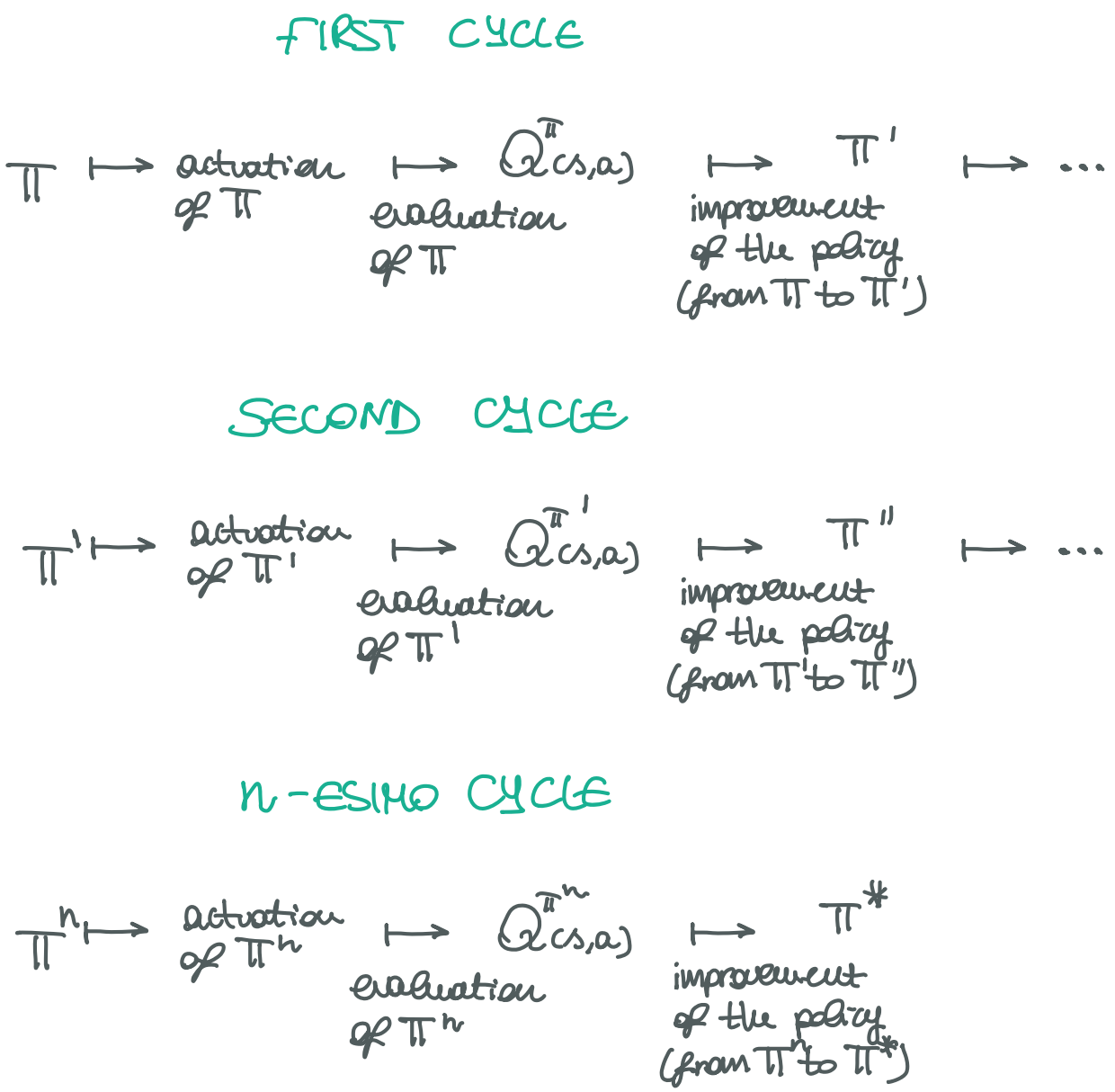
the new est will be worse than the previous and the corresponding action will be less used

$$\pi'(s_t, a_i) := \frac{p(s_t, a_i)}{\sum_{a \in A(s_t)} p(s_t, a_i)}$$

(.....14-21/nov.....)

SARSA - On Policy

Algorithm



More exploitation, less exploration.

Policy Actuated = Policy Evaluated.

Q-Learning - Off Policy

Always starts with estimation of an optimal policy.

FIRST CYCLE

$\tilde{\pi}^* \mapsto$ actuation of a ε-greedy policy derived from $\tilde{\pi}^*$ $\mapsto$ $\tilde{Q}^{\tilde{\pi}^*}(s,a)$ $\mapsto$ evaluation of $\tilde{\pi}^*$ $\mapsto$ $\tilde{\pi}^*$ $\mapsto$ improvement of the optimal policy estimation $\mapsto \dots$

SECOND CYCLE

$\tilde{\tilde{\pi}}^* \mapsto$ actuation of a ε-greedy policy derived from $\tilde{\pi}^*$ $\mapsto$ $\tilde{Q}^{\tilde{\pi}^*}(s,a)$ $\mapsto$ evaluation of $\tilde{\pi}^*$ $\mapsto$ $\tilde{\tilde{\pi}}^*$ $\mapsto$ improvement of the optimal policy estimation $\mapsto \dots$

n-ESIMO CYCLE

$\tilde{\tilde{\pi}}^* \mapsto$ actuation of a ε-greedy policy derived from $\tilde{\tilde{\pi}}^*$ $\mapsto$ $\tilde{Q}^{\tilde{\pi}^*}(s,a)$ $\mapsto$ evaluation of $\tilde{\pi}^{n*}$ $\mapsto$ π^* $\mapsto$ improvement of the optimal policy estimation

This method is 100% exploitation and 0% exploration.

Policy Attuated != Optimal Policy.

The Methods:

$$\text{NEW ESTIMATION} = \text{OLD ESTIMATION} + \text{STEPSIZE}\left(\underset{\text{TARGET}}{a_{k+1}} - \underset{\text{ESTIMATE}}{\text{OLD}}\right)$$

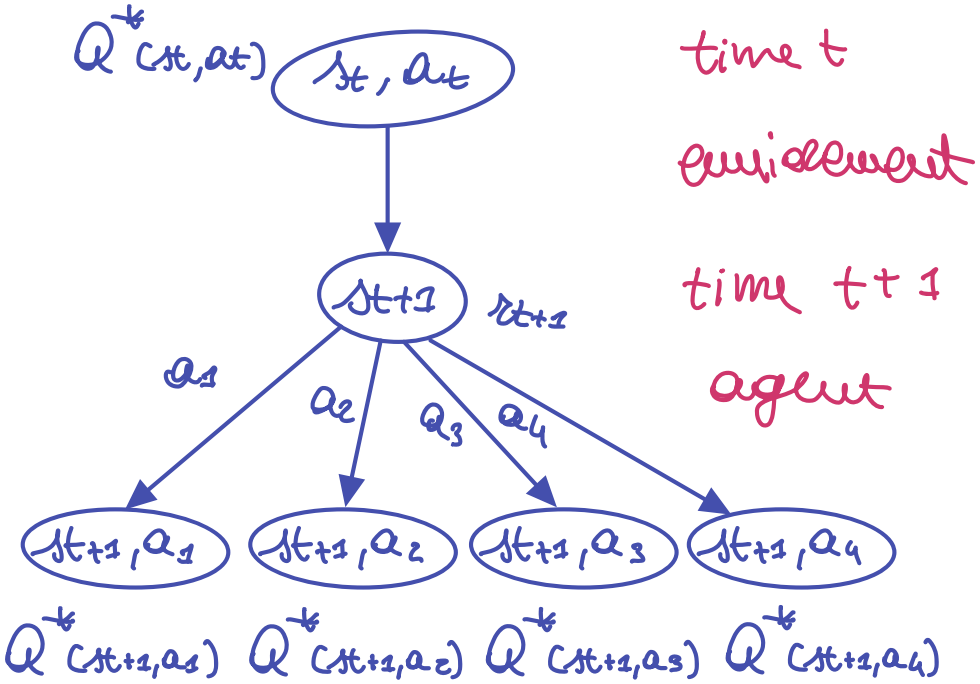
$$\text{TD}: V(s_t) := V(s_t) + \alpha_k(s_t)\left(r_{t+1} + \gamma V(s_{t+1}) - V(s_t)\right)$$

$$\text{SARSA}: Q^{\pi}(s,a) := \text{controller libero}$$

$$\text{Q-L}: Q^{\pi^*}(s,a) := Q^{\pi^*}(s,a) + \underset{= 1/1+k(s,a)}{\alpha_k(s,a)}\left(\text{NEW SAMPLE} - Q^{\pi^*}(s,a)\right)$$

The new concept relies on how to get a new sample.

Consider the following *back up digram*.



Since it is being evaluating an optimal policy PI it is needed to compare all the actions at the state t+1 and choose the maximum.

NEW SAMPLE $r_{t+1} + \gamma \max_{a \in A(s_{t+1})} Q^*(t+1, a_i)$

In the end:

$$Q^*(s_t, a_t) := Q^*(s_t, a_t) + \frac{1}{1+k(s,a)} \left[r_{t+1} + \gamma \max_{a \in A(s_{t+1})} Q^*(s_{t+1}, a) - Q^*(s_t, a_t) \right]$$

$$\hat{a}_t = \arg\max_{a \in A(s_t)} Q^*(s_t, a)$$

example:

if

Q^*	a_1	a_2
s_1	5	6
s_2	7	4

if

Q^*	a_1	a_2
s_1	5	4
s_2	7	4

$$\hat{a}_t = \arg\max_{a \in A(s_t)} Q^*(s_t, a)$$

ε-greedy policy

$$\begin{cases} \pi'(s_t, \hat{a}_t) = 1 - \varepsilon + \dfrac{\varepsilon}{|A(s_t)|} \\ \pi'(s_t, a) = \dfrac{\varepsilon}{|A(s_t)|} \quad \forall a \neq \hat{a}_t \end{cases}$$

↳ remember $\sum_{a \in A(s_t)} \pi(s_t, a) = 1$

↳ maintain a non-null probability of performing every action

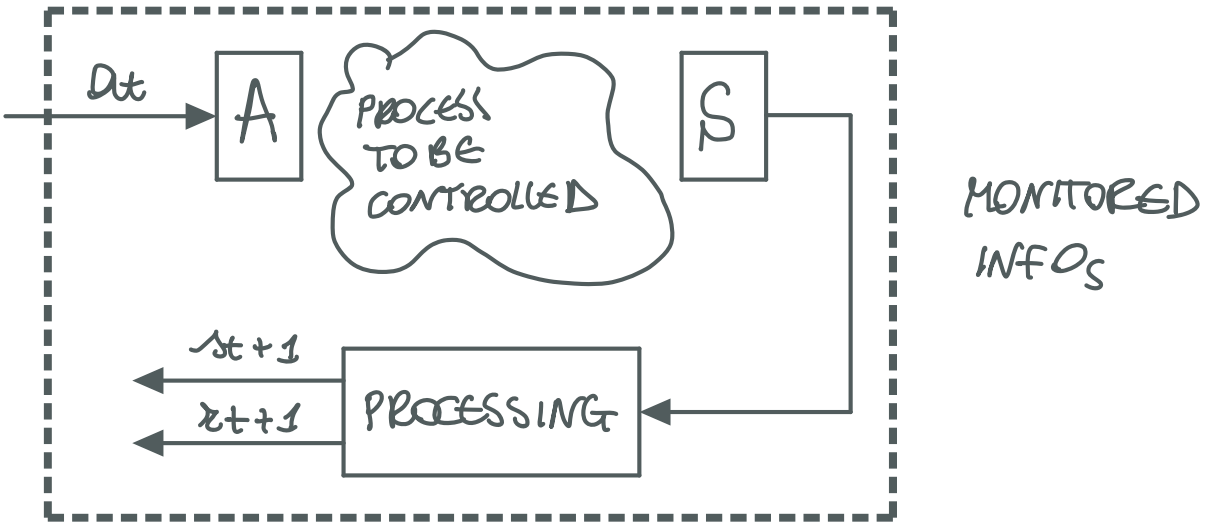
(PN) if care the algoritm converge to π^* independently on which one is being used

It has being said that the first PI policy number is choosen randomly; well this is not entirely true. It is always better to start from reasonable values based on experience.

Since it has no exploration

$$R_t = E\left[\underbrace{r_{t+1}}_{[t, t+1]} + \gamma \underbrace{\sum_{k=1}^{\infty} \gamma^{k-1} r_{t+k+1}}_{[t+1, \infty]} \right]$$

Enviorement scheme becames



Meaning that the state must be observable.

The Q-Learning Algorithm.

1.The policy actuator selcts the action at

2. The action at is actualised on the process and the corresponding st+1 and r+1 are observed.

3. The Q*(s,a) evaluator computes Q*(st,at) according to #1.

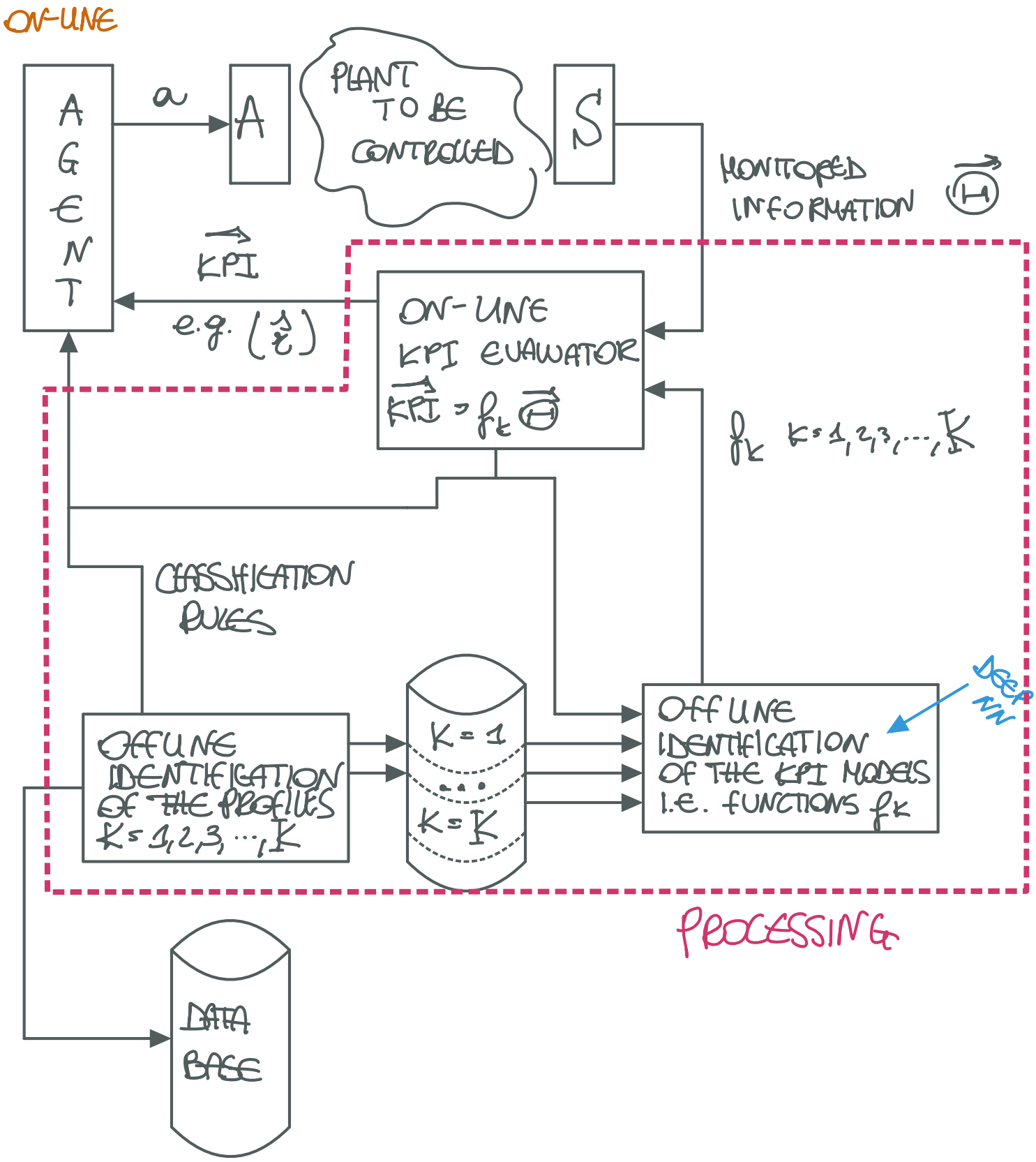
4. The policy improverer updates the policy according to #2.

5. st := st+1 && at := at+1.

when the switch closes

Deep Reinforcement Learning.

call TETA the output of S sensors so all the parameters that it is possible to monitor.



In this case, in the processing block, the only online block is the KPI evaluation.

TETA is often a function of the type y = f(x), to identify f(x) is the problem.

Pay attention pedex k represents the current profile.

Off-line identification in k-means which identifies classification rules.

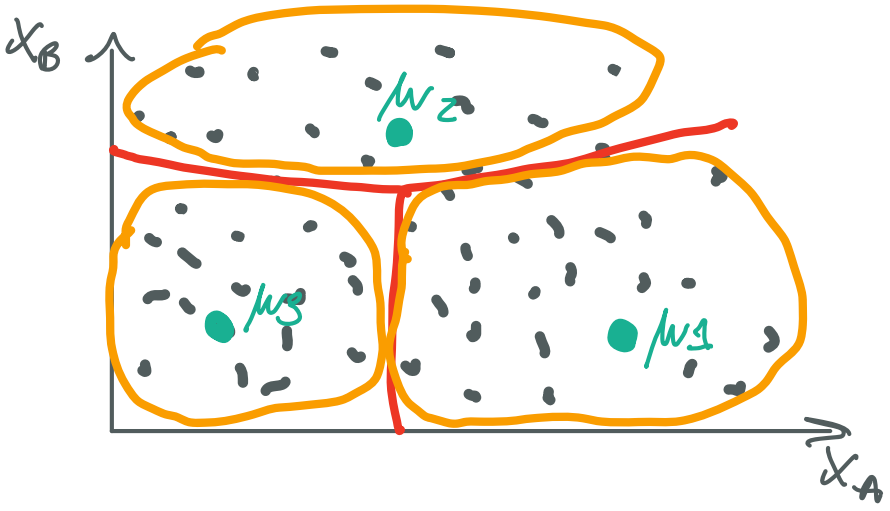
K-Means Clustering.

It is needed to dispose of a dataset

$$D = \{ \vec{x}_1, \vec{x}_2, \dots, \vec{x}_n, \dots, \vec{x}_N \}$$

↳ generic data point

$$\vec{x}_n = \begin{pmatrix} \vec{x}_{An} \\ \vec{x}_{Bn} \end{pmatrix}$$



It is needed to cluster these data points in K profiles belonging to the ones that are near the one and other.

$$r_{nx} = \begin{cases} 1 & \text{if } \vec{x}_n \text{ is assign to cluster K} \\ 0 & \text{otherwise} \end{cases}$$

$\vec{\mu}_K$ centroid or prototype of the cluster K

↳ "baricentric of the cluster

There is an algorithm that can associate each centroid with cluster k.

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{n,k} \| \vec{x}_n - \vec{\mu}_k \|^2$$

GOAL: find $r_{n,k}$ & $\vec{\mu}_k$ so that J is minimised

why do we want $\min_{r_{n,k}, \vec{\mu}_k} J$?

Algorithm.

Random initialisation of $\vec{\mu}_k$

$\vec{\mu}_k$

PHASE 1 — Minimize J with respect to $r_{n,k}$, keeping the $\vec{\mu}_k$ fixed

$\vec{\mu}_k$

PHASE 2 — Minimize J with respect to the $\vec{\mu}_k$ keeping r_{nk} fixed $\longrightarrow \frac{\partial J}{\partial \vec{\mu}_k} = 0$

solve $\frac{\partial J}{\partial \mu_{Ak}} = 0 \quad \frac{\partial J}{\partial \mu_{Bk}} = 0$

if $f=2$, $N=4$

$$J = \sum_{k=1}^{K} \left\{ r_{1k}\left[(x_{A1}-\mu_{Ak})^2 + (x_{B1}-\mu_{Bk})^2\right] + \right.$$

$$+ r_{2k}\left[\quad \cdots\cdots\cdots\cdots \quad \right] +$$

$$+ r_{3k}\left[\quad \cdots\cdots\cdots\cdots \quad \right] +$$

$$\left. + r_{4k}\left[\quad \cdots\cdots\cdots\cdots \quad \right] \right.$$

$$\frac{\partial J}{\partial \mu_{Ak}} = 2\left[r_{1k}(x_{A1}-\mu_{Ak}) + r_{2k}(x_{A2}-\mu_{Ak}) + \right.$$

$$\left. + r_{3k}(x_{A3}-\mu_{Ak}) + r_{4k}(x_{A4}-\mu_{Ak}) \right] = 0$$

$$\mu_{Ak} = \frac{r_{1k}x_{A1} + r_{2k}x_{A2} + r_{3k}x_{A3} + r_{4k}x_{A4}}{r_{1k} + r_{2k} + r_{3k} + r_{4k}}$$

And the same for $\dfrac{\partial J}{\partial \mu_{Bk}} = 0 \;\mapsto\; \mu_{Bk}$