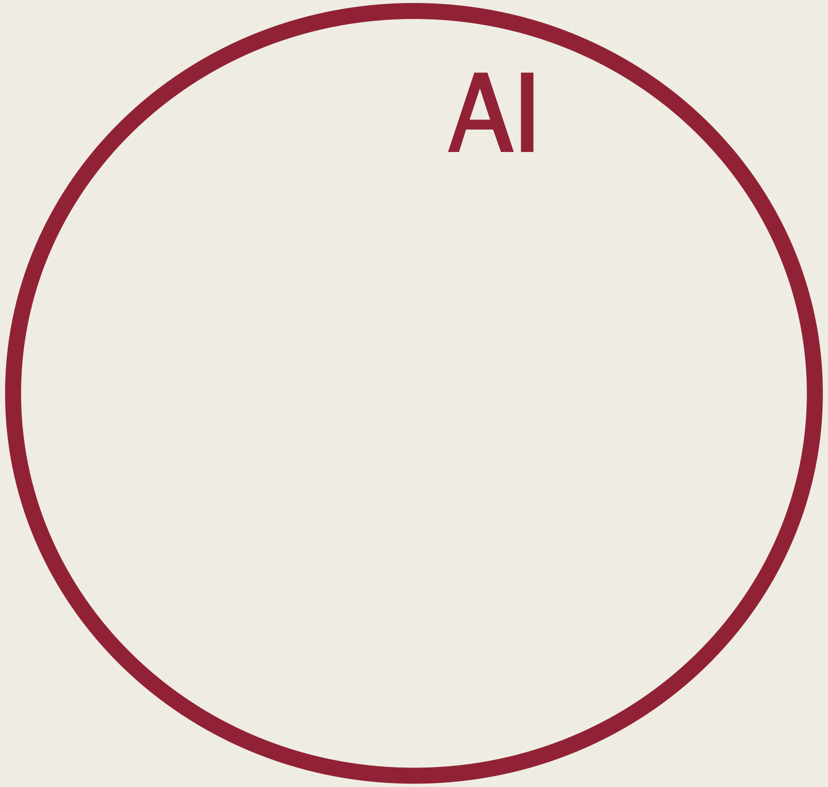
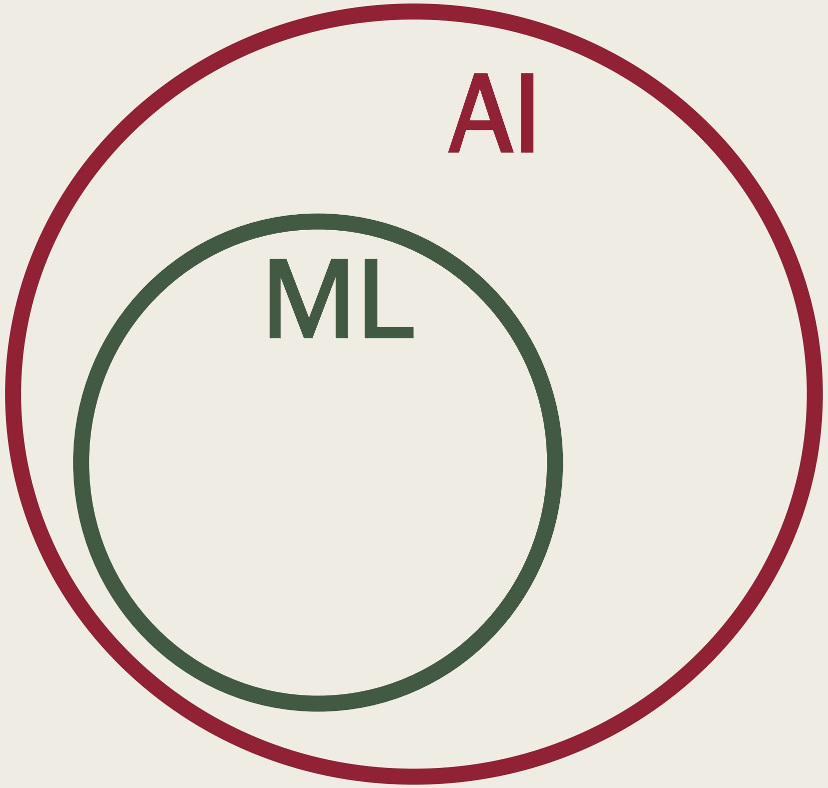
REINFORCEMENT LEARNING

torresjm – jan 2022


AI


AI
ML

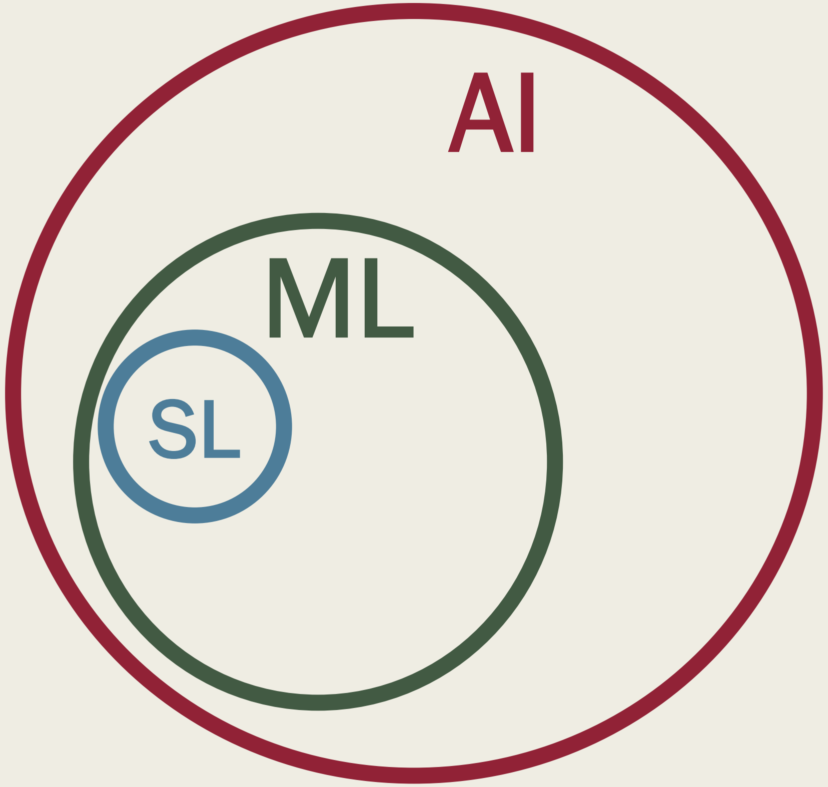
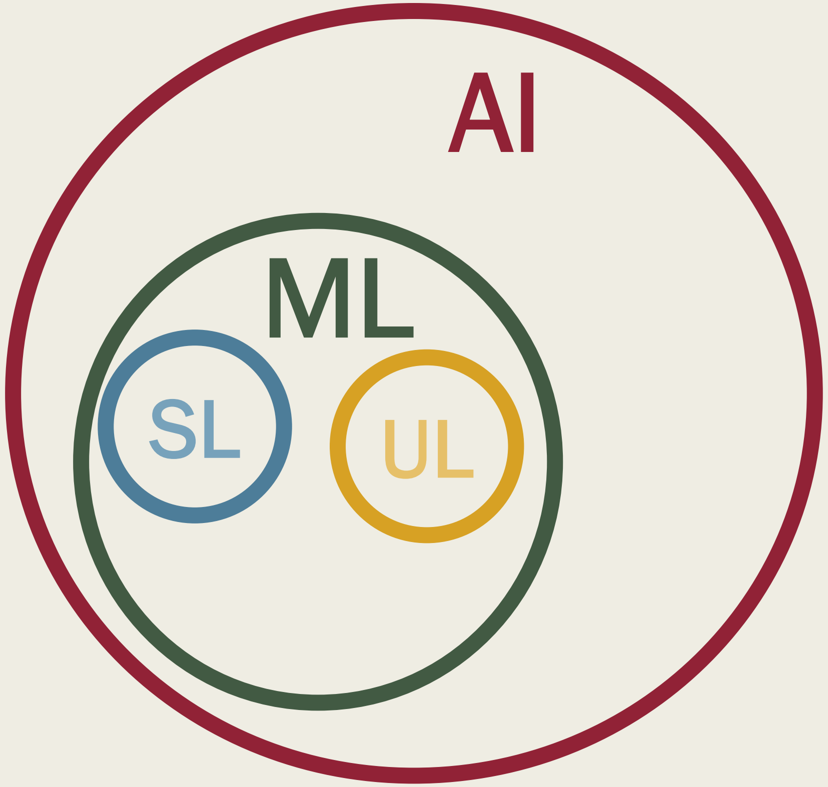


Figure 1.2: *Examples of handwritten digits from U.S. postal envelopes.*

- Data, labels

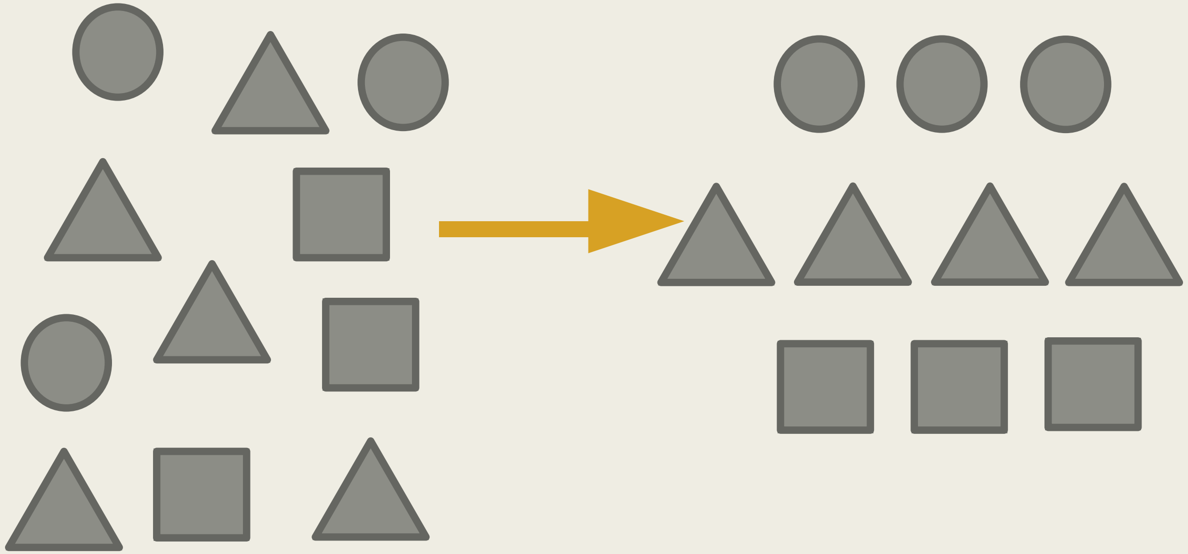
- Train
- Test

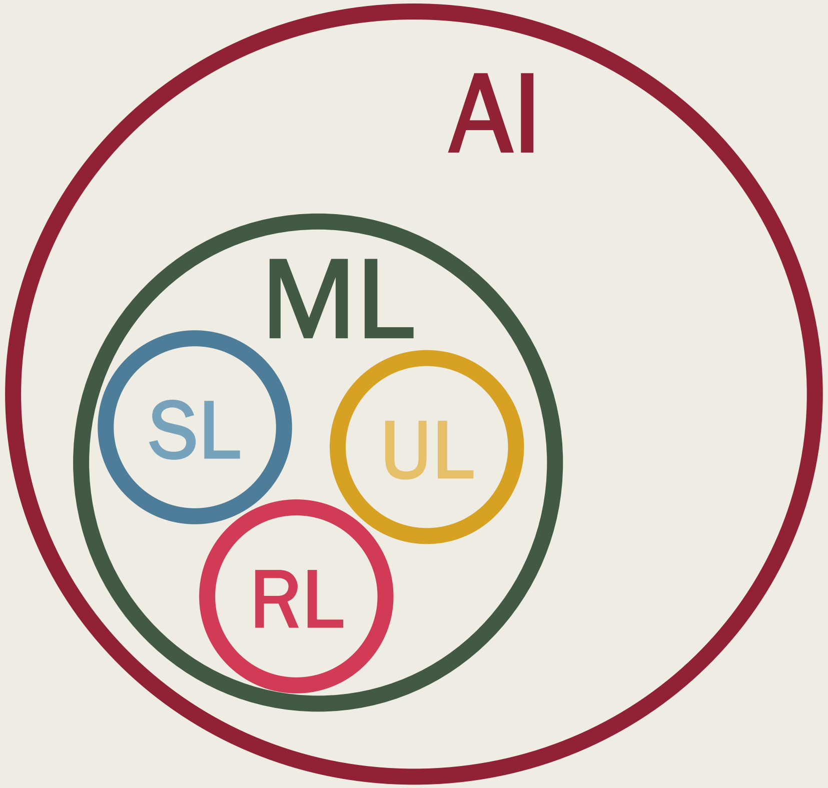
- Predict



Data (and no label)

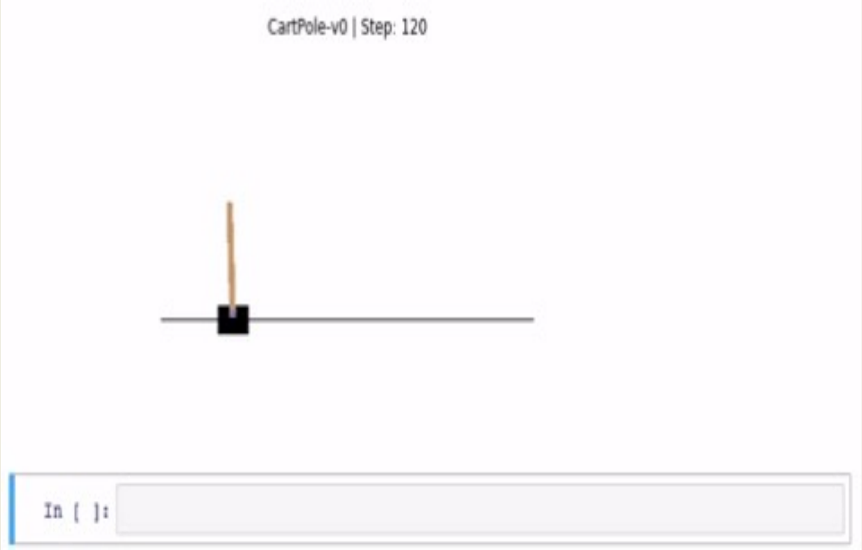
finding some structure in the dataset, grouping data.





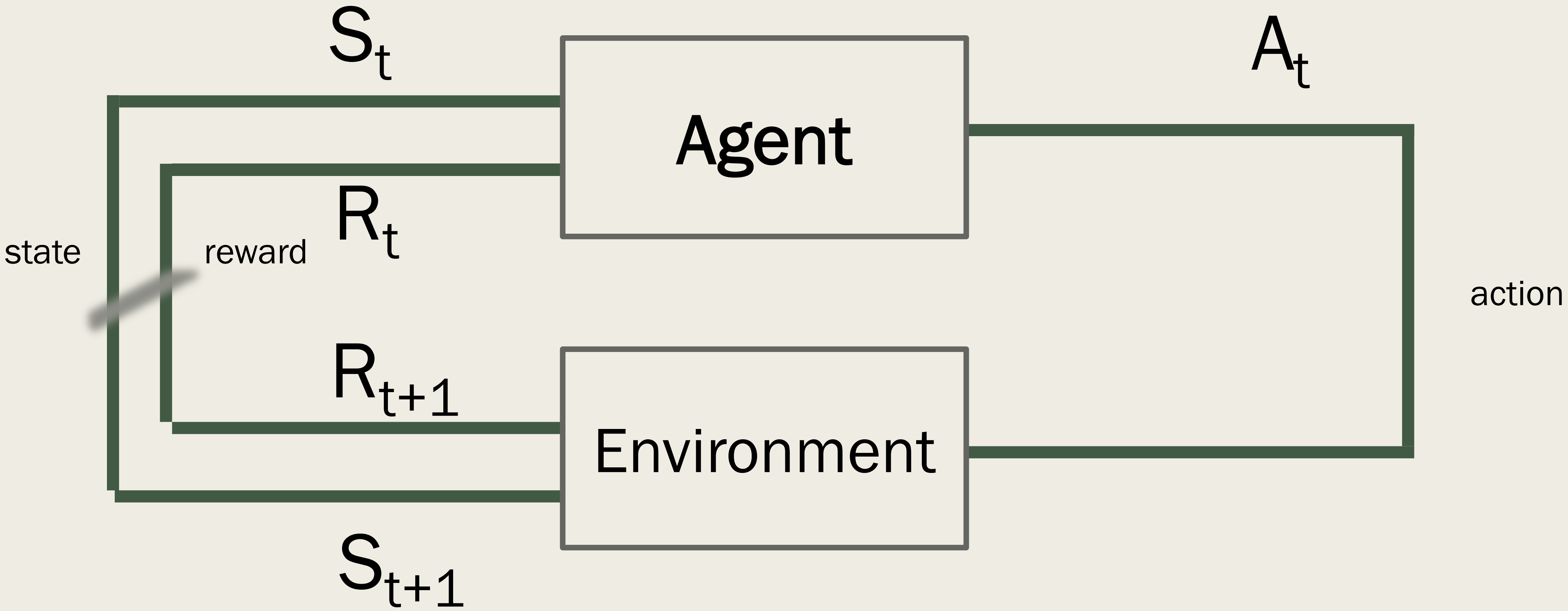
an « agent » interacts with an « environment » to learn what sequences of actions will maximize the rewards it will receive.

RL - example



Cartpole :

- the agent controls the cart, trying to keep the pole up. linteraction ends when the pole falls down
- 2 actions :
  - left,
  - right
- State :
  - position,
  - cart velocity,
  - cart / pole angle,
  - top of pole velocity



trajectory : $S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, \ldots R_n, S_n, A_n$

Decision process

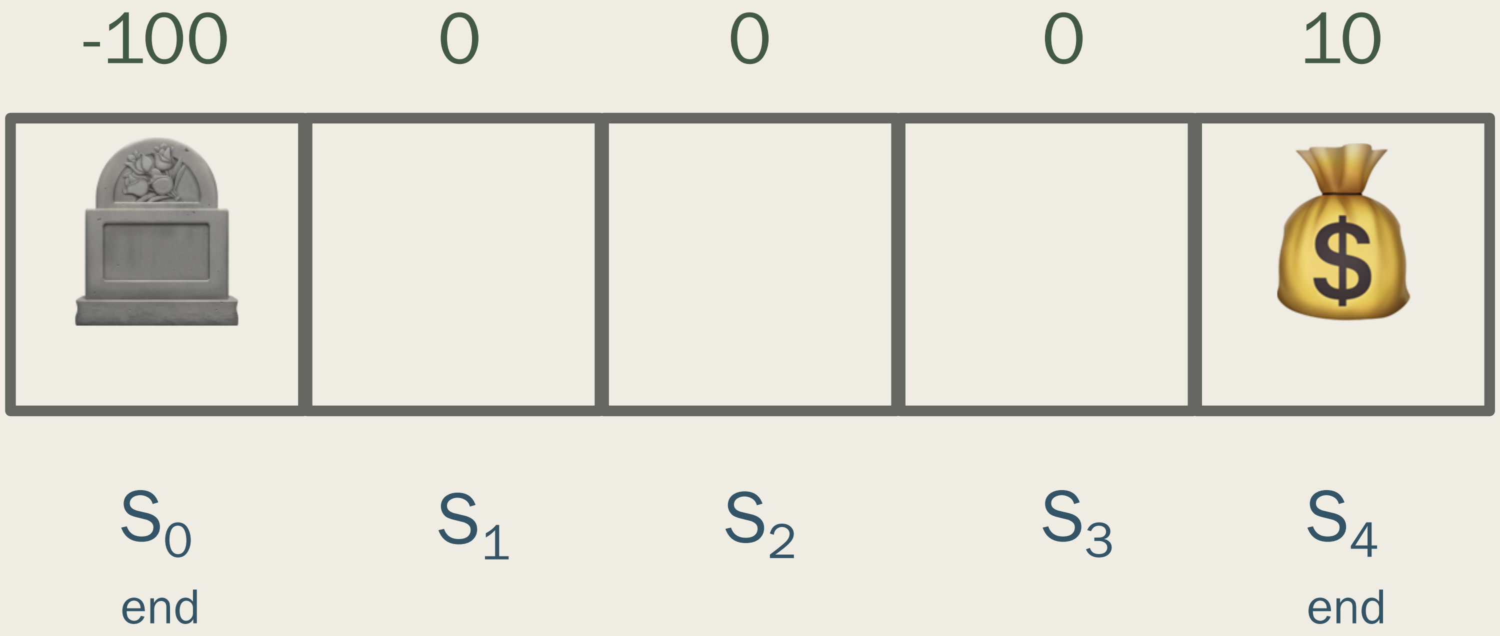
$$s \, , a \, , p \, , r$$

$$P(s'|s,a) = P(S_{t+1} \mid S_t = s \, , A_t = a)$$

Notes:

- markovian process (futur depends on present, not past)
- p,r : are given (model based) or not (model free)

Example : Row GridWorld



Reward function : $r(s, a, s') = E[R_{t+1} \mid S_t = s, A_t = a, S_{t+1} = s']$

ex :

- $r(S_1, \leftarrow, S_0) = -100$
- $r(S_4, \leftarrow, S_3) = 0$
- $r(S_4, \rightarrow, S_5) = 10$

policy : $\boldsymbol{\pi}$

- a policy $\boldsymbol{\pi}$ decribes the behaviour of the agent (for each state in the environment)
- Resolving the environment mean finding the policy that maxmizes the future return
- this will be the optimal policy $\boldsymbol{\pi}_*$

Return is defined by : $G_t = R_{t+1} + R_{t+2} + R_{t+3} + \ldots$

(more often : $G_t = R_{t+1} + \boldsymbol{\gamma} R_{t+2} + \boldsymbol{\gamma}^2 R_{t+3} + \boldsymbol{\gamma}^3 R_{t+4} + \ldots$)

note : $G_t = R_{t+1} + \boldsymbol{\gamma} G_{t+1}$

For a given policy a state *s* defines an action *a* : $\boldsymbol{\pi}(s) = a$

The state value function measures the return of a policy at each state :

$$v_{\boldsymbol{\pi}}(s) = E_{\boldsymbol{\pi}}[G_t \mid S_t = s]$$

policy : $\boldsymbol{\pi}$

- Optimal policy $\boldsymbol{\pi}_*$ is associated to $\mathbf{v}_*(s) = \max_{\boldsymbol{\pi}} (v_{\boldsymbol{\pi}}(s)$ for any s
- Now resolving the environment if finding $\mathbf{v}_*$
- Resolving the environment is finding the policy that maxmizes the future return

$$v_{\boldsymbol{\pi}}(s) = \boldsymbol{\Sigma}_a \boldsymbol{\pi}(a|s) \, \boldsymbol{\Sigma}_{s'} \, p(s'|s,a)[r(s,a,s') + \boldsymbol{\gamma} v_{\boldsymbol{\pi}}(s')]$$

Which lead to the Bellman Equation (that give an algo for solving :

$$v_*(s) = \max_a [\, \boldsymbol{\Sigma}_a \boldsymbol{\pi}(a|s) \, \boldsymbol{\Sigma}_{s'} \, p(s'|s,a)[r(s,a,s') + \boldsymbol{\gamma} v_{\boldsymbol{\pi}}(s')]]$$

try all policies, finding v max will give the optimal policy

```
input π
init v(s) = 0 for all s
repeat :
    for each s do :
        v(s) ← Σaπ(a|s) Σs' p(s'|s,a)[r(s,a,s') + γvπ (s')
    end
untill change of v is lower than a threshold
return v (=approx vπ)
```