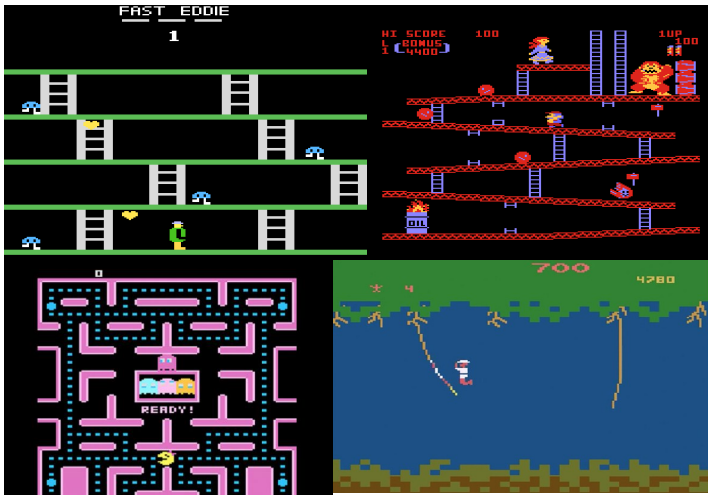
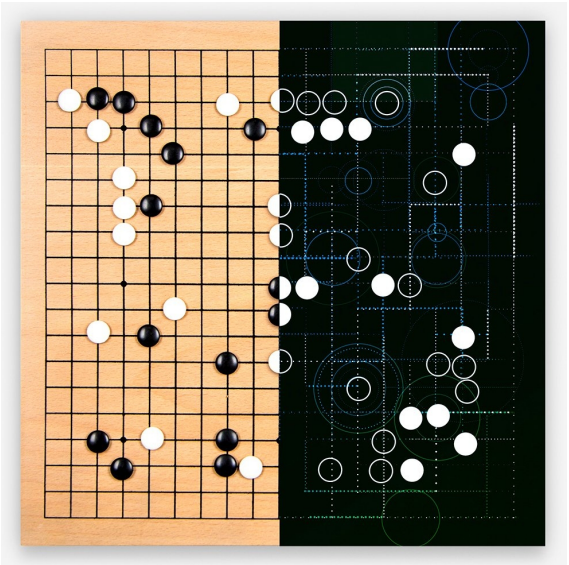
MDPs and Dynamic Programming

Olivier Sigaud

Sorbonne Université
http://people.isir.upmc.fr/sigaud

Why this class (1)?



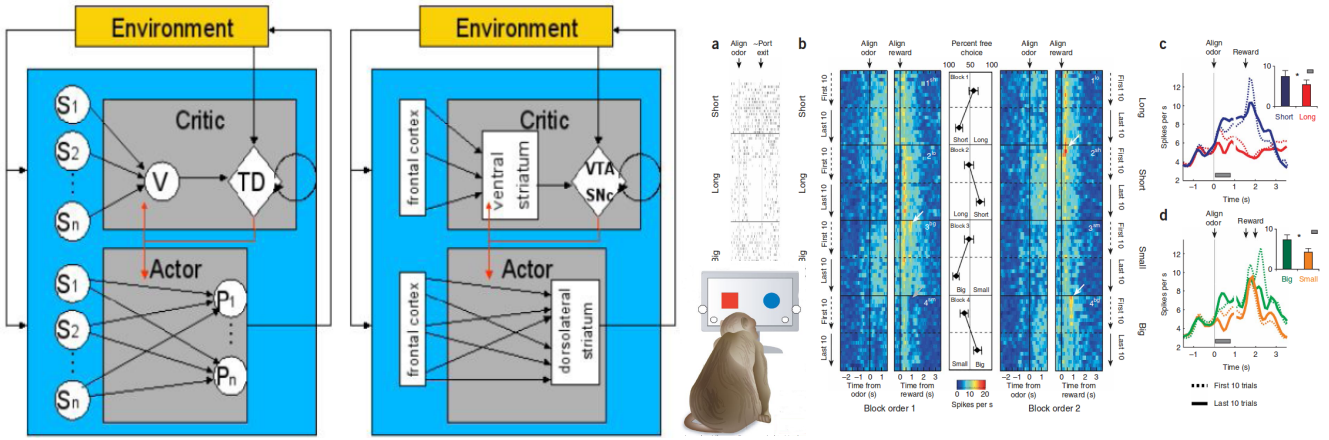


- A lot of buzz about deep reinforcement learning as an engineering tool

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015) Human-level control through deep reinforcement learning. *Nature*, 518(7540), 529–533.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. (2017) Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359

Why this class (2)?

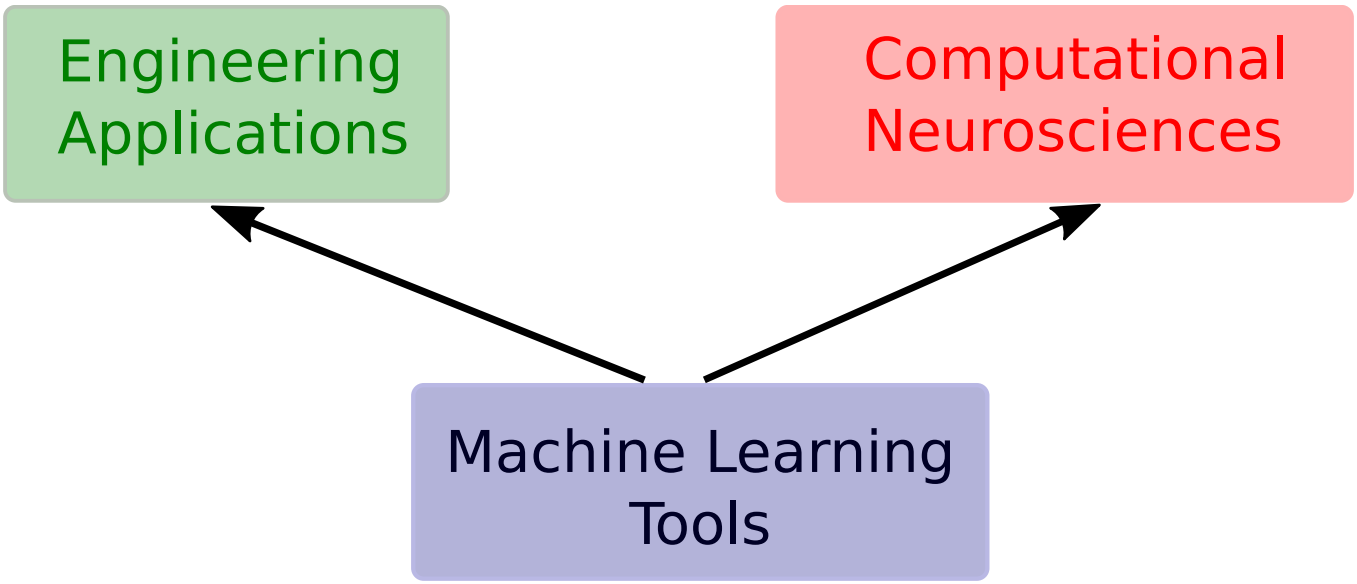


- The reinforcement learning framework is relevant in computational neuroscience
- This aspect will be left out

Takahashi, Y., Schoenbaum, G., & Niv, Y. (2008) Silencing the critics: understanding the effects of cocaine sensitization on dorsolateral and ventral striatum in the context of an actor/critic model. *Frontiers in neuroscience*, 2:14

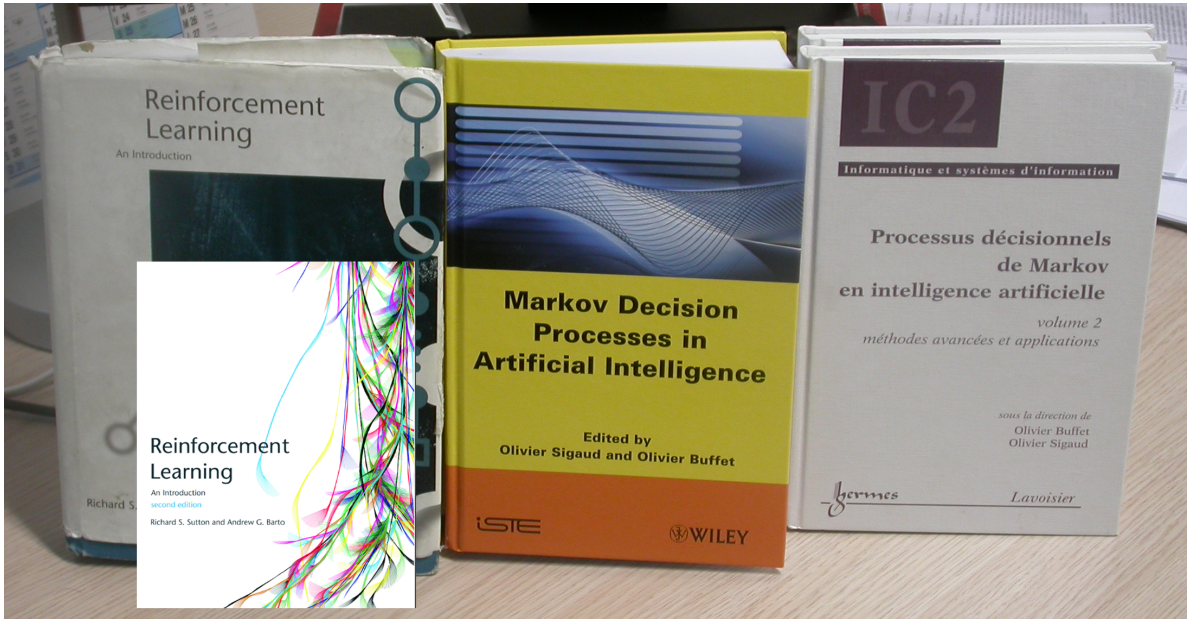
Roesch, M. R., Calu, D. J., & Schoenbaum, G. (2007) Dopamine neurons encode the better option in rats deciding between differently delayed or sized rewards. *Nature Neuroscience*, 10(12):1615–1624

General goal



- Provide the machine learning background
- Explain basic concepts from the discrete case
- Provide an introduction to deep RL algorithms

Introductory books

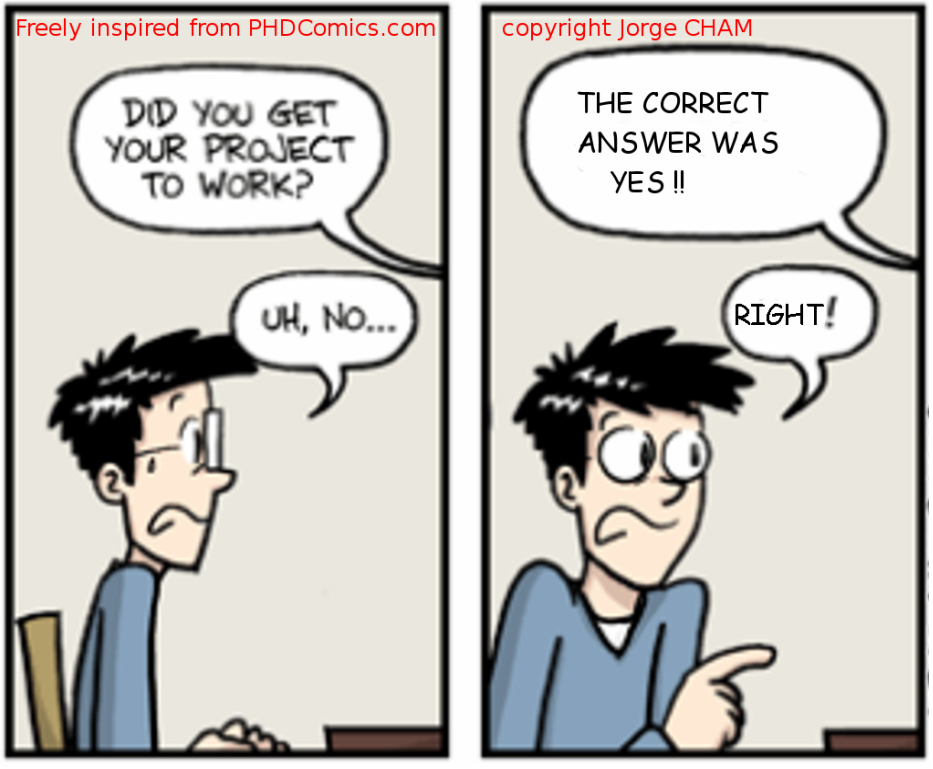


1. [Sutton and Barto, 1998]: the ultimate introduction to the field, in the discrete case
2. New edition available:

   `https://drive.google.com/file/d/1xeUDVGWGUUv1-ccUMAZHJLej2C7aAFWY/view`
3. [Buffet and Sigaud, 2008]: in french
4. [Sigaud and Buffet, 2010]: (improved) translation of 3

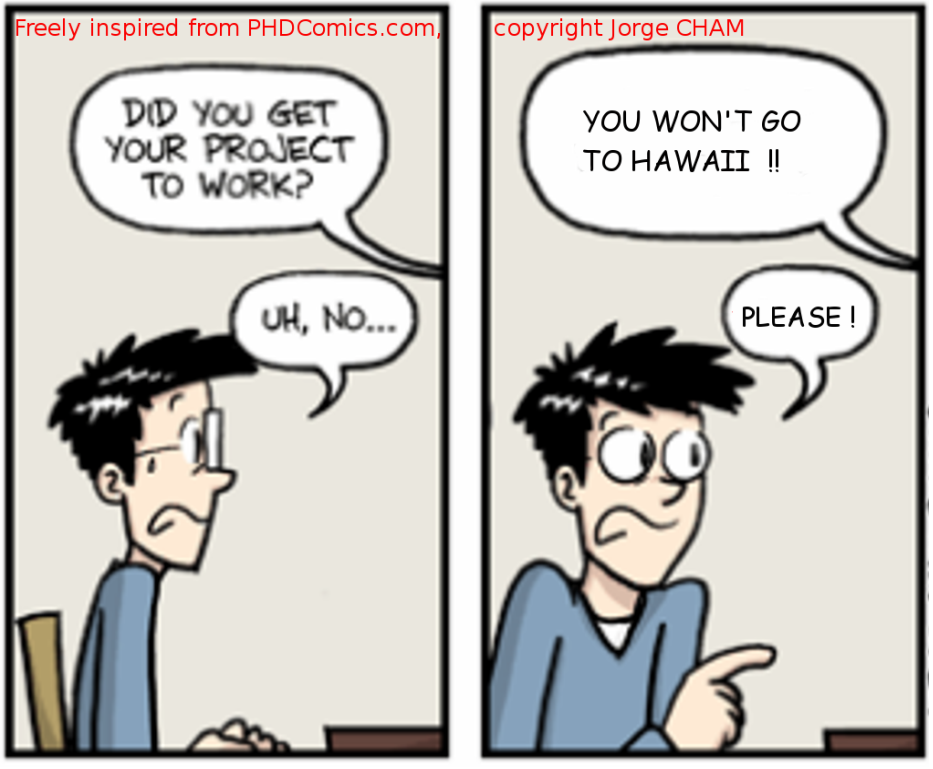
Sutton, R. S. & Barto, A. G. (1998) *Reinforcement Learning: An Introduction*. MIT Press.

Supervised learning



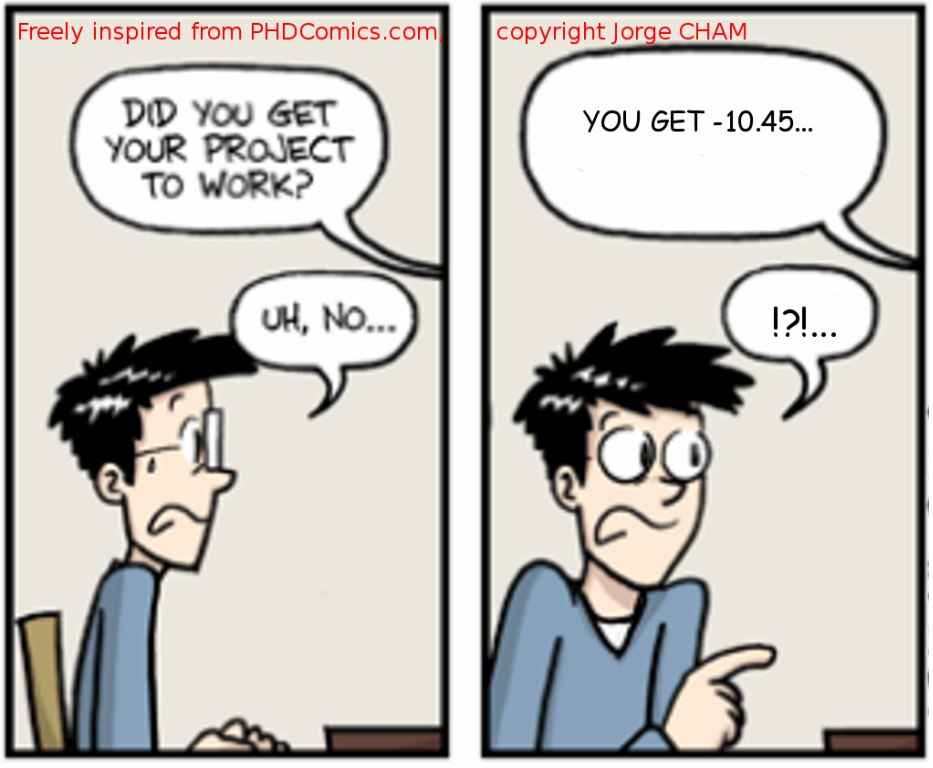
- The supervisor indicates to the agent the expected answer
- The agent corrects a model based on the answer
- Typical mechanism: gradient backpropagation, RLS
- Applications: classification, regression, function approximation...

Cost-Sensitive Learning



- The environment provides the value of action (reward, penalty)
- Application: behaviour optimization

Reinforcement learning



- In RL, the value signal is given as a scalar
- How good is -10.45?
- Necessity of exploration

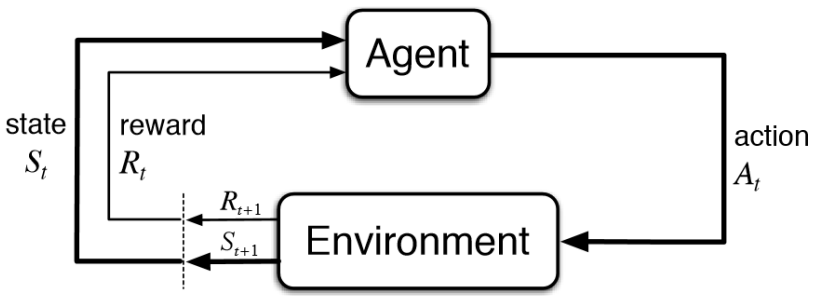
The exploration/exploitation trade-off

- Exploring can be (very) harmful
- Shall I exploit what I know or look for a better policy?
- Am I optimal? Shall I keep exploring or stop?
- Decrease the rate of exploration along time
- ϵ-*greedy*: take the best action most of the time, and a random action from time to time

What's next?

1. Markov Decision Processes
2. Dynamic programming
3. Model-free Reinforcement Learning
4. Advanced discrete Reinforcement Learning

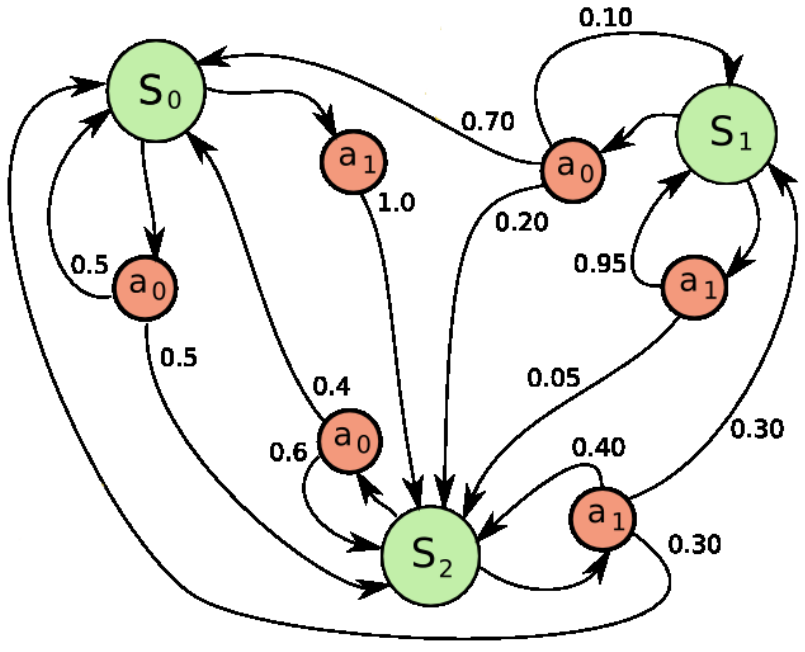
Markov Decision Processes



- S: state space
- A: action space
- $T : S \times A \to \Pi(S)$: transition function
- $r : S \times A \to \mathbb{R}$: reward function

- An MDP describes a problem, not a solution to that problem

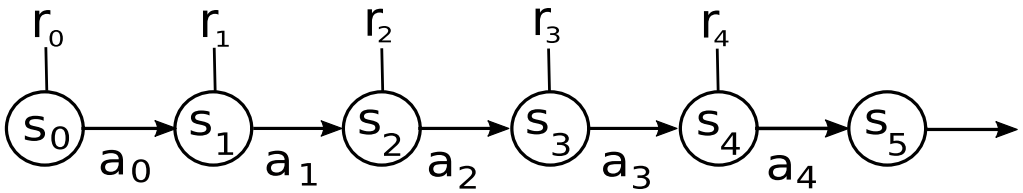
Stochastic transition function



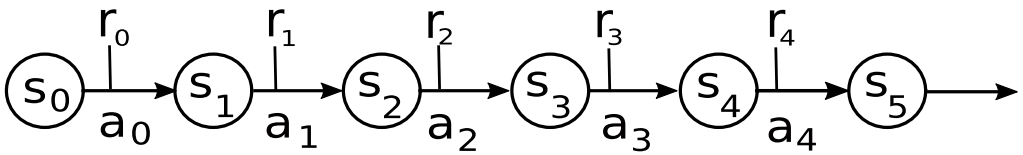
- Deterministic problem = special case of stochastic
- $T(s^t, a^t, s^{t+1}) = p(s'|s, a)$

Rewards: over states or action?

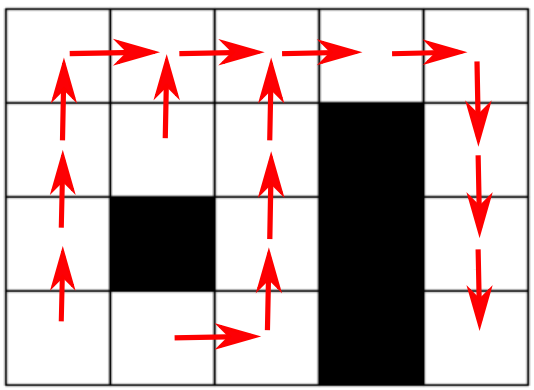
- Reward over states

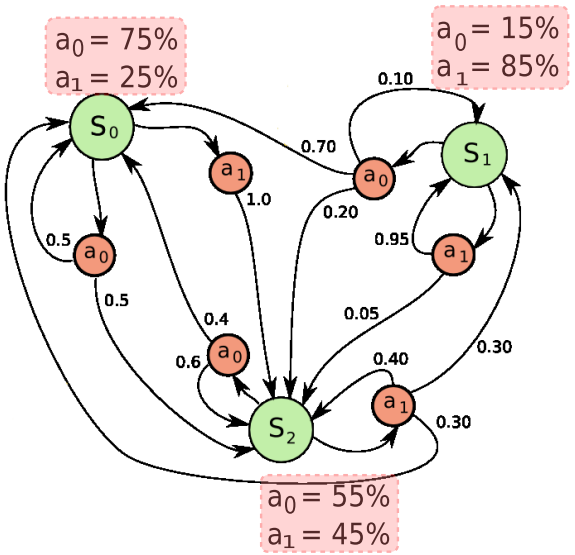


- Reward over actions in states



Deterministic versus stochastic policy

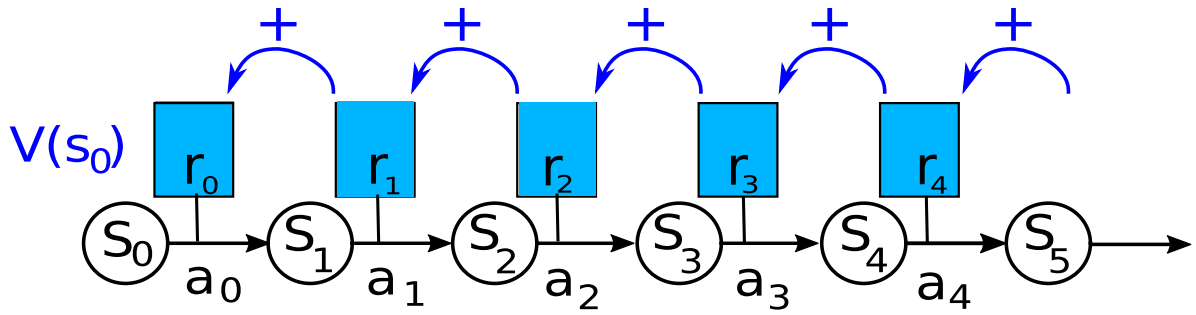




- Goal: find a policy $\pi : S \to A$ maximizing an agregation of rewards on the long run
- Important theorem: for any MDP, there exists a deterministic policy that is optimal

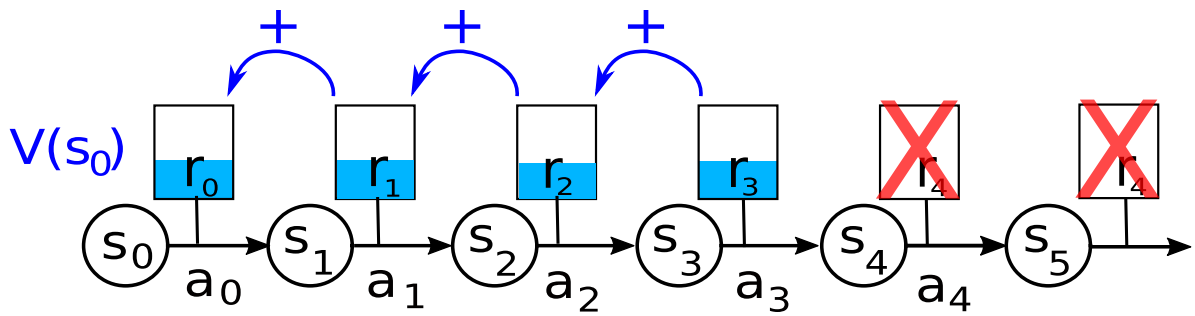
Agregation criterion: mere sum

- The computation of value functions assumes the choice of an agregation criterion (discounted, average, etc.)



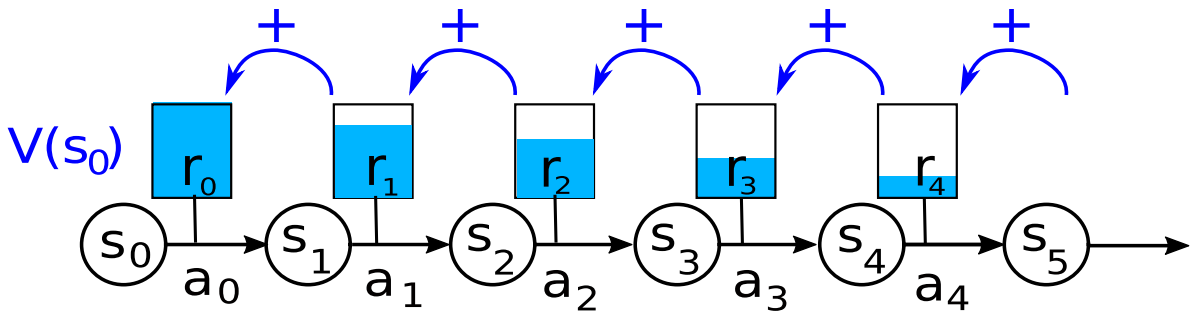
- The sum over a infinite horizon may be infinite, thus hard to compare
- Mere sum (finite horizon N): $V^{\pi}(S_0) = r_0 + r_1 + r_2 + \ldots + r_N$

Agregation criterion: average over a window



- Average criterion on a window: $V^{\pi}(S_0) = \frac{r_0+r_1+r_2}{3}$...

Agregation criterion: discounted

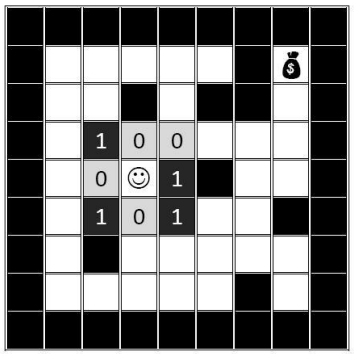


- Discounted criterion: $V^{\pi}(s_{t_0}) = \sum_{t=t_0}^{\infty} \gamma^t r(s_t, \pi(s_t))$
- $\gamma \in [0, 1]$: discount factor
  - if $\gamma = 0$, sensitive only to immediate reward
  - if $\gamma = 1$, future rewards are as important as immediate rewards
- The discounted case is the most used

Markov Property

- An MDP defines s^{t+1} and r^{t+1} as $f(s_t, a_t)$
- Markov property : $p(s^{t+1}|s^t, a^t) = p(s^{t+1}|s^t, a^t, s^{t-1}, a^{t-1}, ...s^0, a^0)$
- In an MDP, a memory of the past does not provide any useful advantage
- Reactive agents $a_{t+1} = f(s_t)$, without internal states nor memory, can be optimal

Markov property: Limitations

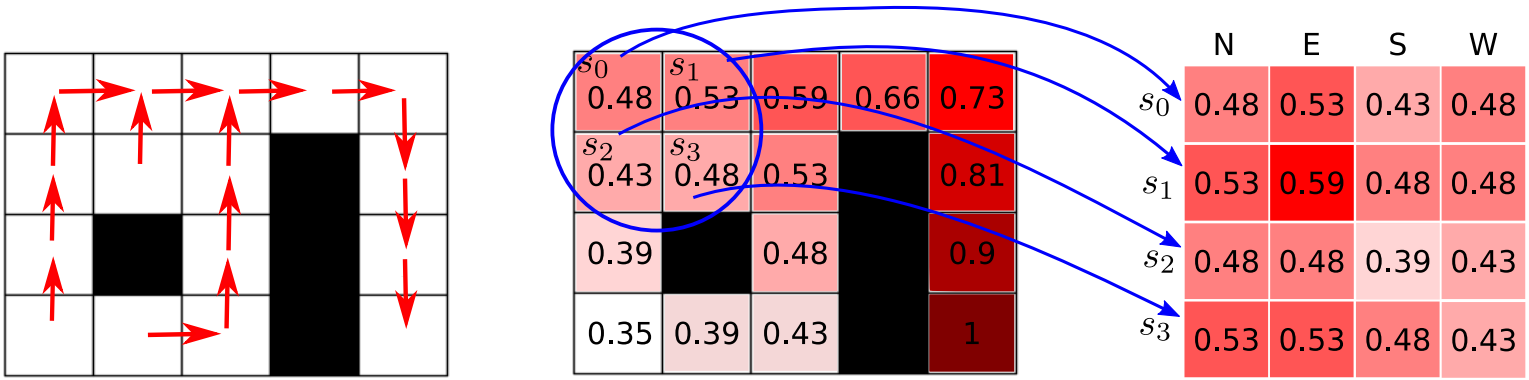


- Markov property is not verified if:
  - the state does not contain all useful information to take decisions (POMDPs)
  - or if the next state depends on decisions of several agents (Dec-MDPs, Dec-POMDPs, Markov games)
  - or if transitions depend on time

Dynamic Programming

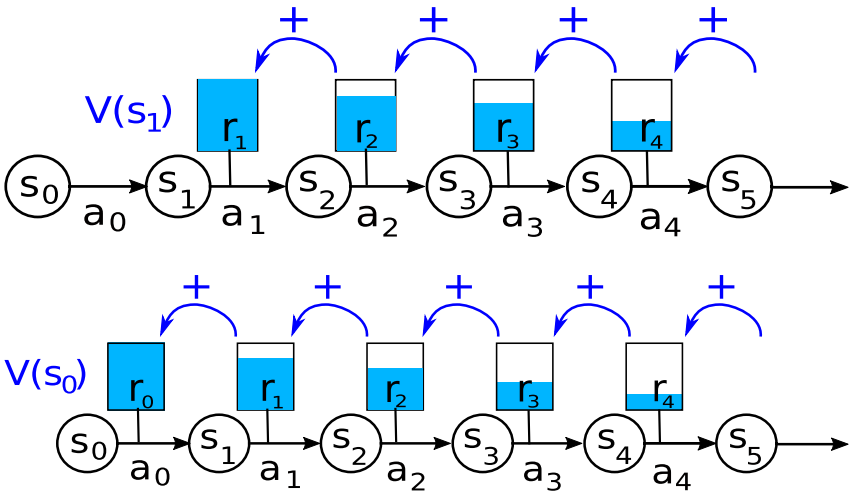
- Once we have defined an MDP
- Dynamic programming methods can find the optimal policy
- Assuming they know everything about the MDP
- Reinforcement Learning applies when the transition and reward functions are unknown
- To define dynamic programming methods, we need value functions

Value and action value functions



- The value function $V^\pi : S \to \mathbb{R}$ records the agregation of reward on the long run for each state (following policy π). It is a vector with one entry per state
- The action value function $Q^\pi : S \times A \to \mathbb{R}$ records the agregation of reward on the long run for doing each action in each state (and then following policy π). It is a matrix with one entry per state and per action
- In the remainder, we focus on V, trivial to transpose to Q

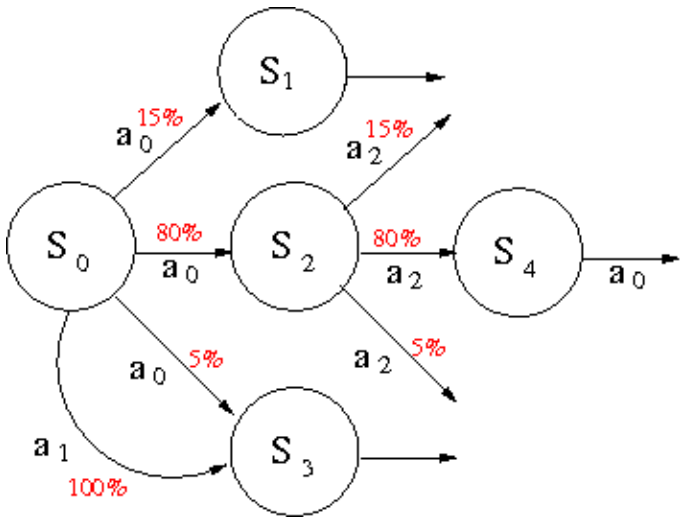
Bellman equation over a Markov chain: recursion



Given the discounted reward agregation criterion:

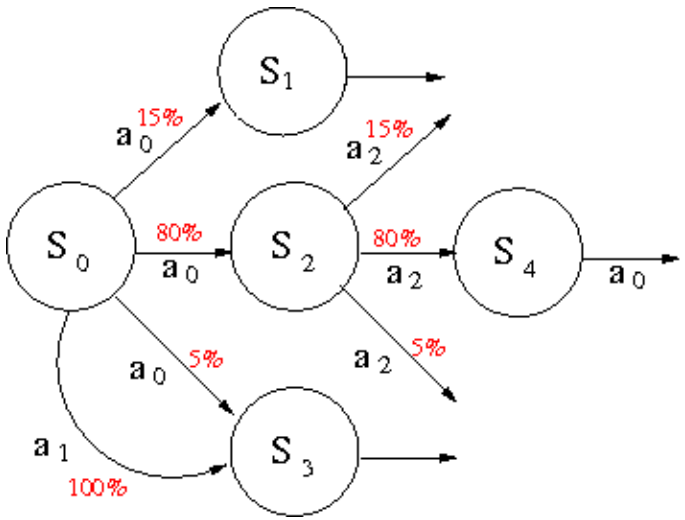
- $V(s_0) = r_0 + \gamma r_1 + \gamma^2 r_2 + \gamma^3 r_3 + ...$
- $V(s_0) = r_0 + \gamma(r_1 + \gamma r_2 + \gamma^2 r_3 + ...)$
- $V(s_0) = r_0 + \gamma V(s_1)$
- More generally $V(s_t) = r_t + \gamma V(s_{t+1})$

Bellman equation: general case



- Generalisation of $V(s_t) = r_t + \gamma V(s_{t+1})$ over all possible trajectories
- The expectation of a random variable is the sum of the realizations weighted by their probabilities
- The realizations are the next states
- Deterministic π: $V^\pi(s) = r(s, \pi(s)) + \gamma \sum_{s'} p(s'|s, \pi(s)) V^\pi(s')$

Bellman equation: general case



- Generalisation of $V(s_t) = r_t + \gamma V(s_{t+1})$ over all possible trajectories
- The expectation of a random variable is the sum of the realizations weighted by their probabilities
- The realizations are the next states
- Stochastic π: $V^\pi(s) = \sum_a \pi(a|s)[r(s,a) + \gamma \sum_{s'} p(s'|s,a)V^\pi(s')]$

Recursive operators and convergence

- If we define an operator T such that $X_{n+1} \leftarrow TX_n$
- It T is contractive, then through repeated application of T, X_n will converge to some fixed point
- For instance, if T divides by 2, X_n converges to 0

The Bellman optimality operator (Value Iteration)

- We call Bellman optimality operator (noted T^*) the application

$$V_{n+1}(s) \leftarrow \max_{a \in A} \Big[r(s,a) + \gamma \sum_{s'} p(s'|s,a) V_n(s') \Big]$$

- If $\gamma < 1$, T^* is contractive
- By iterating, computes the value of the current policy
- The optimal value function is the fixed-point of T^*: $V^* = T^* V^*$
- Value iteration: $V_{n+1} \leftarrow T^* V_n$

Puterman, M. L. (2014) *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons.

The Bellman operator (Policy Iteration)

- We call Bellman operator (noted T^π) the application

$$V^\pi_{n+1}(s) \leftarrow r(s, \pi(s)) + \gamma \sum_{s'} p(s'|s, \pi(s)) V^\pi_n(s')$$

- If $\gamma < 1$, T is contractive
- Converges to optimal value and policy
- Policy Iteration:
  - Policy evaluation:
    $V^\pi_{n+1} \leftarrow T^\pi V^\pi_n$
  - Policy improvement:
    $\forall s \in S, \pi'(s) \leftarrow \arg\max_{a \in A} \sum_{s'} p(s'|s,a)[r(s,a) + \gamma V^\pi_n(s')]$
    or
    $\forall s \in S, \pi'(s) \leftarrow \arg\max_{a \in A} [r(s,a) + \gamma \sum_{s'} p(s'|s,a) V^\pi_n(s')]$
- Note: $\sum_{s',r} p(s', r|s, a)[r + \gamma V(s')] = r + \gamma \sum_{s'} p(s'|s, a) V(s')$

Value Iteration: the algorithm

Value Iteration, for estimating $\pi \approx \pi_*$

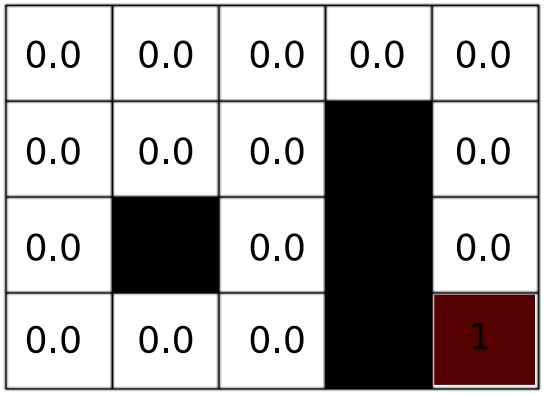
Algorithm parameter: a small threshold $\theta > 0$ determining accuracy of estimation
Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(terminal) = 0$

Loop:
| $\Delta \leftarrow 0$
| Loop for each $s \in \mathcal{S}$:
| $v \leftarrow V(s)$
| $V(s) \leftarrow \max_a \sum_{s',r} p(s', r|s, a)\big[r + \gamma V(s')\big]$
| $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$

Output a deterministic policy, $\pi \approx \pi_*$, such that
$\pi(s) = \arg\max_a \sum_{s',r} p(s', r|s, a)\big[r + \gamma V(s')\big]$

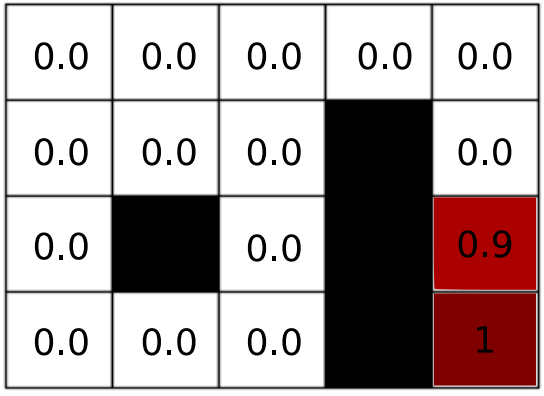
- Taken from Sutton & Barto, 2018, p. 83
- Reminder: $\sum_{s',r} p(s', r|s, a)[r + \gamma V(s')] = r + \gamma \sum_{s'} p(s'|s, a)V(s')$

Value Iteration in practice



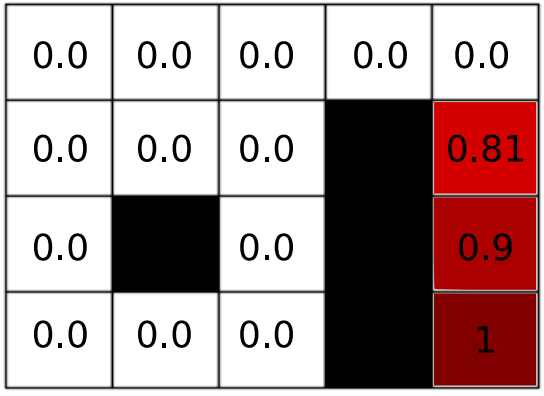
$$\forall s \in S, V_{i+1}(s) \leftarrow \max_{a \in A} \Big[r(s,a) + \gamma \sum_{s'} p(s'|s,a) V_i(s') \Big]$$

Value Iteration in practice



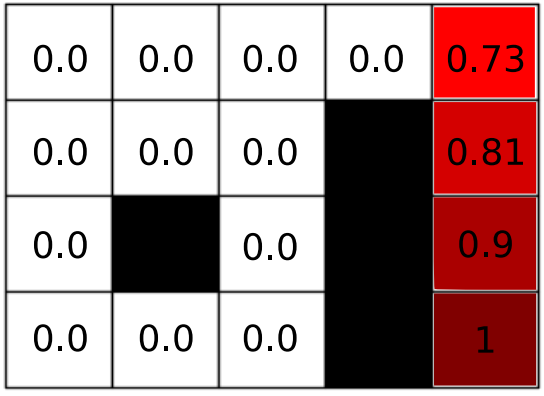
$$\forall s \in S, V_{i+1}(s) \leftarrow \max_{a \in A} \Big[r(s,a) + \gamma \sum_{s'} p(s'|s,a) V_i(s') \Big]$$

Value Iteration in practice



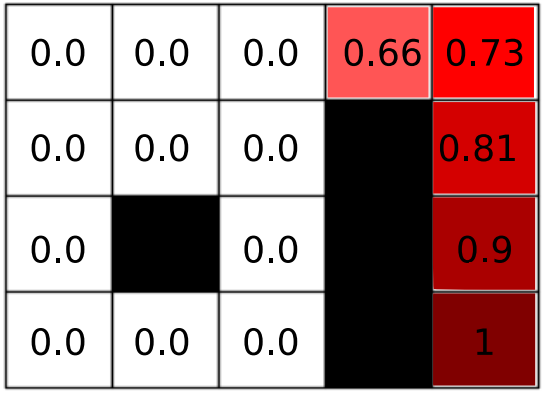
$$\forall s \in S, V_{i+1}(s) \leftarrow \max_{a \in A} \Big[r(s,a) + \gamma \sum_{s'} p(s'|s,a) V_i(s') \Big]$$

Value Iteration in practice



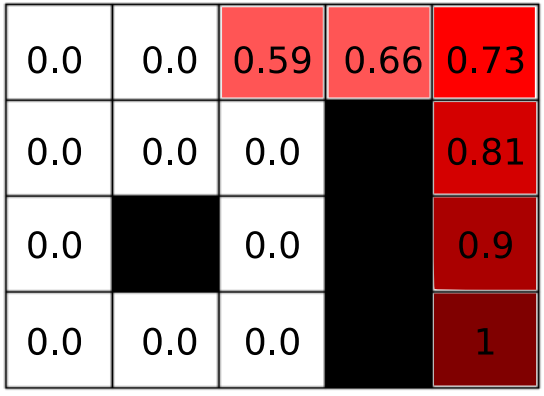
$$\forall s \in S, V_{i+1}(s) \leftarrow \max_{a \in A} \Big[r(s,a) + \gamma \sum_{s'} p(s'|s,a) V_i(s') \Big]$$

Value Iteration in practice



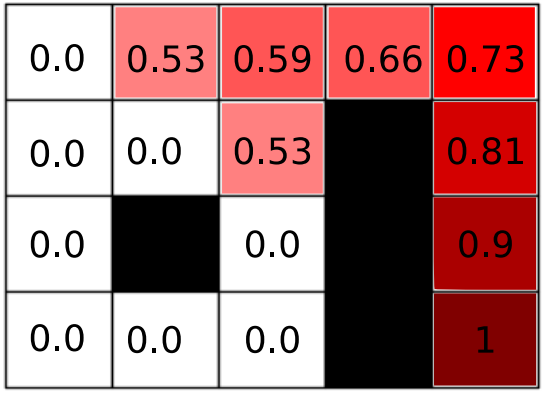
$$\forall s \in S, V_{i+1}(s) \leftarrow \max_{a \in A} \Big[r(s,a) + \gamma \sum_{s'} p(s'|s,a) V_i(s') \Big]$$

Value Iteration in practice



$$\forall s \in S, V_{i+1}(s) \leftarrow \max_{a \in A} \Big[r(s,a) + \gamma \sum_{s'} p(s'|s,a) V_i(s') \Big]$$

Value Iteration in practice

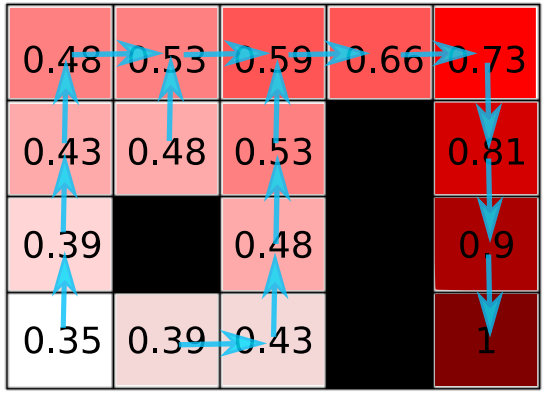


$$\forall s \in S, V_{i+1}(s) \leftarrow \max_{a \in A} \Big[r(s,a) + \gamma \sum_{s'} p(s'|s,a) V_i(s') \Big]$$

Value Iteration in practice

$$\forall s \in S, V_{i+1}(s) \leftarrow \max_{a \in A} \Big[r(s,a) + \gamma \sum_{s'} p(s'|s,a) V_i(s') \Big]$$

Value Iteration in practice



We have iterated on values, and determined a policy out of it (without necessarily representing it if using $Q(s, a)$)

Policy Iteration: the algorithm

Policy Iteration (using iterative policy evaluation) for estimating $\pi \approx \pi_*$

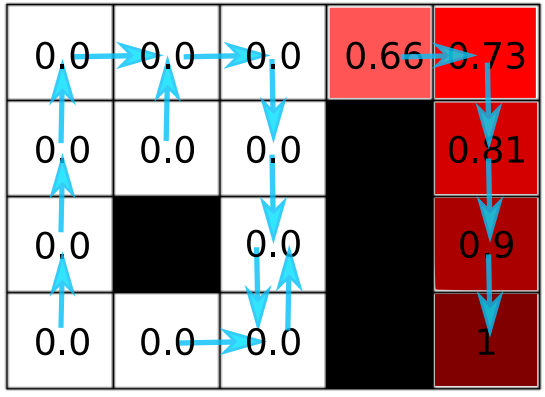
1. Initialization
   $V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

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

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

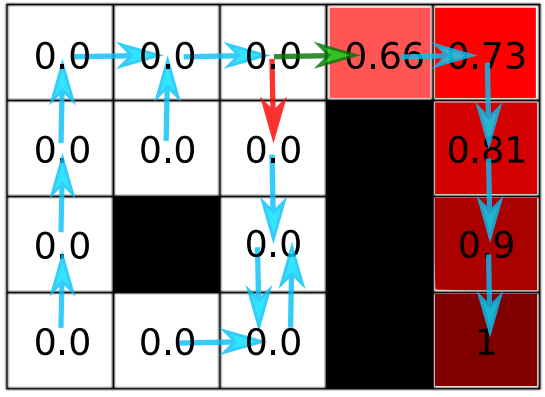
- Taken from Sutton & Barto, 2018, p. 80
- Note: $\sum_{s',r} p(s', r|s, a)[r + \gamma V(s')] = r + \gamma \sum_{s'} p(s'|s, a)V(s')$

Policy Iteration in practice



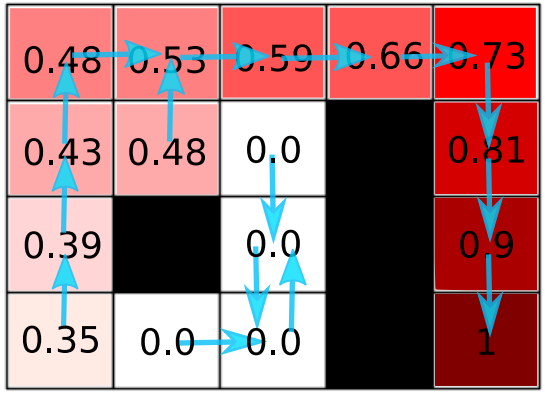
$$\forall s \in S, V_i(s) \leftarrow evaluate(\pi_i(s))$$

Policy Iteration in practice



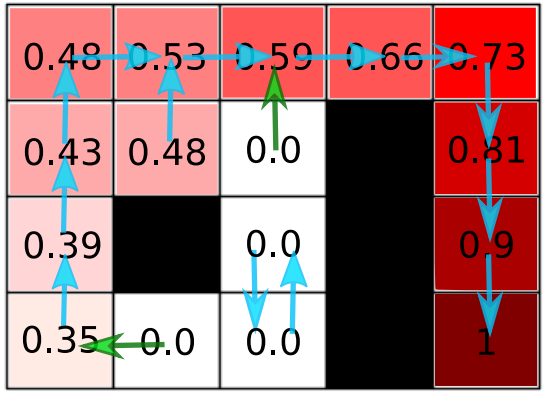
$$\forall s \in S, \pi_{i+1}(s) \leftarrow improve(\pi_i(s), V_i(s))$$

Policy Iteration in practice



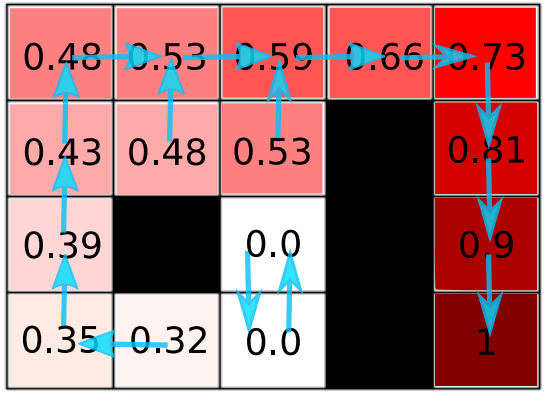
$$\forall s \in S, V_i(s) \leftarrow evaluate(\pi_i(s))$$

Policy Iteration in practice



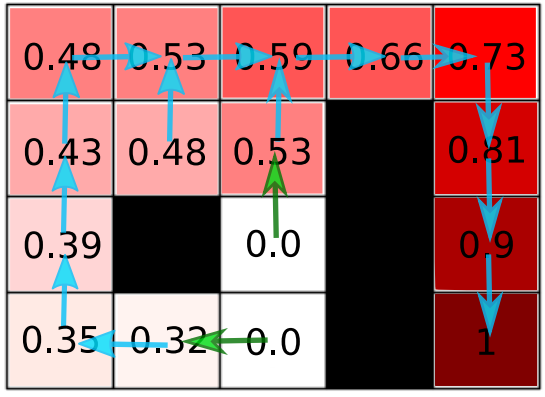
$$\forall s \in S, \pi_{i+1}(s) \leftarrow improve(\pi_i(s), V_i(s))$$

Policy Iteration in practice



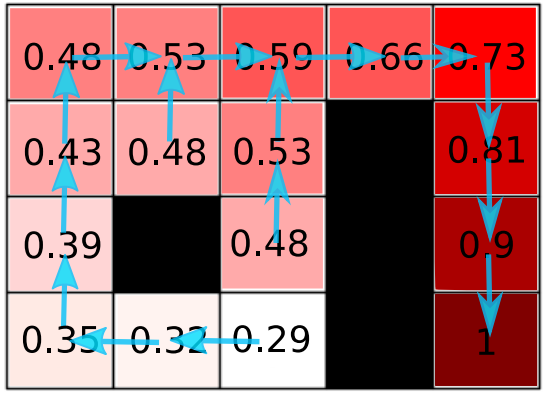
$$\forall s \in S, V_i(s) \leftarrow evaluate(\pi_i(s))$$

Policy Iteration in practice



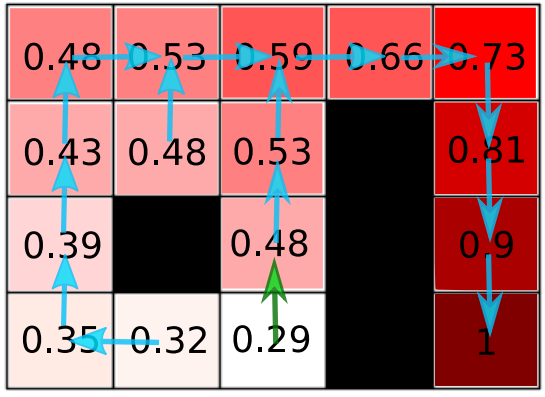
$$\forall s \in S, \pi_{i+1}(s) \leftarrow improve(\pi_i(s), V_i(s))$$

Policy Iteration in practice



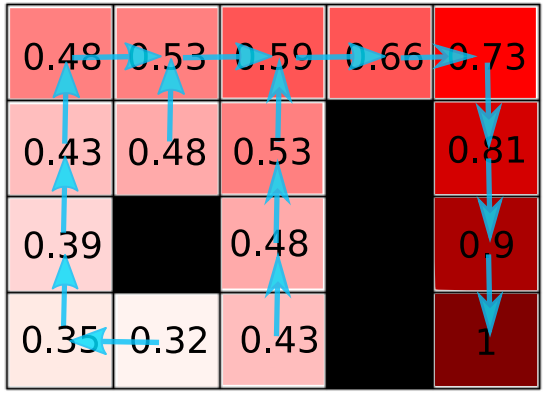
$$\forall s \in S, V_i(s) \leftarrow evaluate(\pi_i(s))$$

Policy Iteration in practice



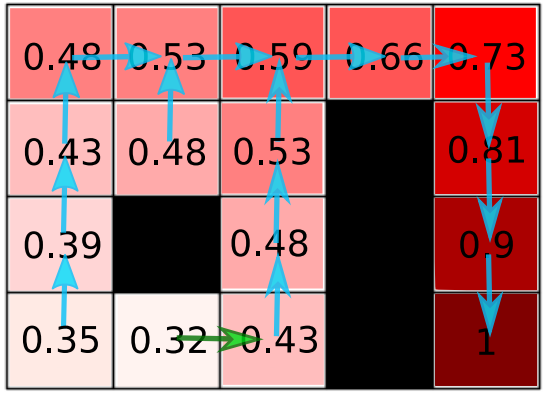
$$\forall s \in S, \pi_{i+1}(s) \leftarrow improve(\pi_i(s), V_i(s))$$

Policy Iteration in practice



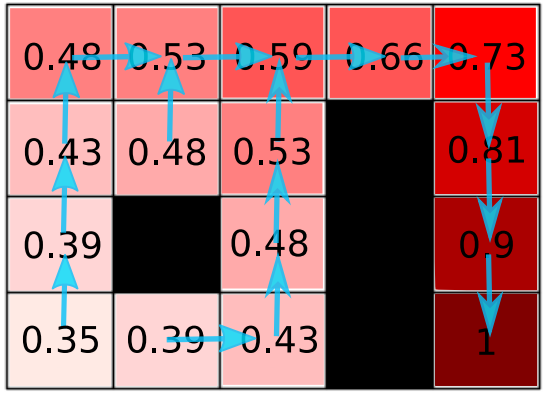
$$\forall s \in S, V_i(s) \leftarrow evaluate(\pi_i(s))$$

Policy Iteration in practice



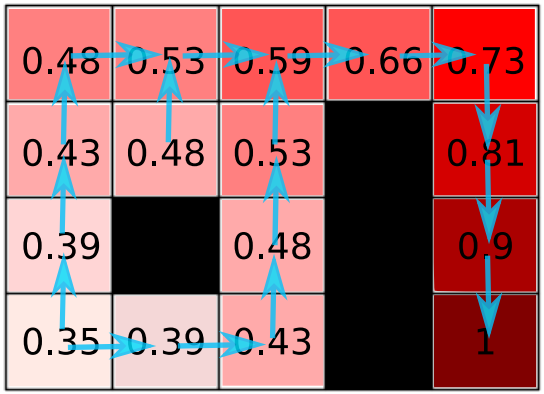
$$\forall s \in S, \pi_{i+1}(s) \leftarrow improve(\pi_i(s), V_i(s))$$

Policy Iteration in practice



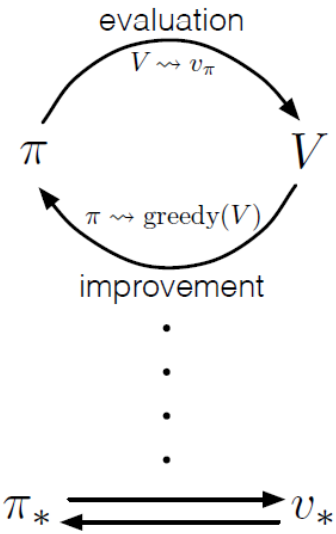
$$\forall s \in S, V_i(s) \leftarrow evaluate(\pi_i(s))$$

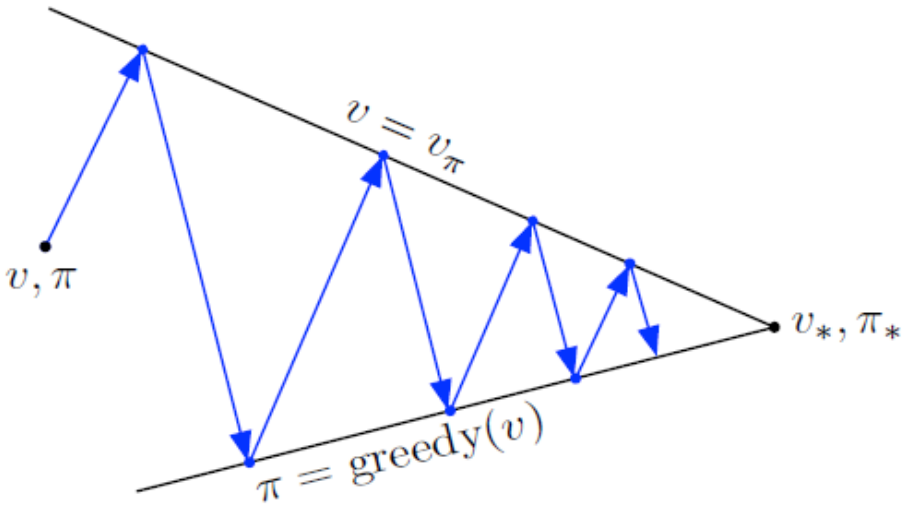
Policy Iteration in practice



Here we have managed a policy and a value representations at all steps

Generalized Policy Iteration





- Policy iteration evaluates each intermediate policy up to convergence. This is slow.
- Instead, evaluate the policy for N iterations, or even not for all states.
- Asynchronous dynamics programming: decoupling policy evaluation and improvement
- Taken from Sutton & Barto, 2018

Any question?

Send mail to: Olivier.Sigaud@upmc.fr


Buffet, O. and Sigaud, O. (2008).
Processus décisionnels de Markov en intelligence artificielle.
Lavoisier.


Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015).
Human-level control through deep reinforcement learning.
Nature, 518(7540):529–533.


Puterman, M. L. (2014).
Markov decision processes: discrete stochastic dynamic programming.
John Wiley & Sons.


Roesch, M. R., Calu, D. J., and Schoenbaum, G. (2007).
Dopamine neurons encode the better option in rats deciding between differently delayed or sized rewards.
Nature Neuroscience, 10(12):1615–1624.


Sigaud, O. and Buffet, O. (2010).
Markov Decision Processes in Artificial Intelligence.
iSTE - Wiley.


Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., et al. (2017).
Mastering chess and shogi by self-play with a general reinforcement learning algorithm.
arXiv preprint arXiv:1712.01815.


Sutton, R. S. and Barto, A. G. (1998).
Reinforcement Learning: An Introduction.
MIT Press.


Takahashi, Y., Schoenbaum, G., and Niv, Y. (2008).
Silencing the critics: understanding the effects of cocaine sensitization on dorsolateral and ventral striatum in the context of an actor/critic model.
Frontiers in neuroscience, 2:14.