

## TP N°2 - Reinforcement learning

### Exercice 1 — Jack's Car Rental

Use policy iteration algorithm to solve de Jack's Car Rental problem with the following simplification : we assume that the number of car returned is deterministic, each days there is 3 car returned at location 1 and 2 car returned at location 2.

### Exercice 2 — Value iteration.

Implement the value iteration algorithm to solve Gambler's problem.

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

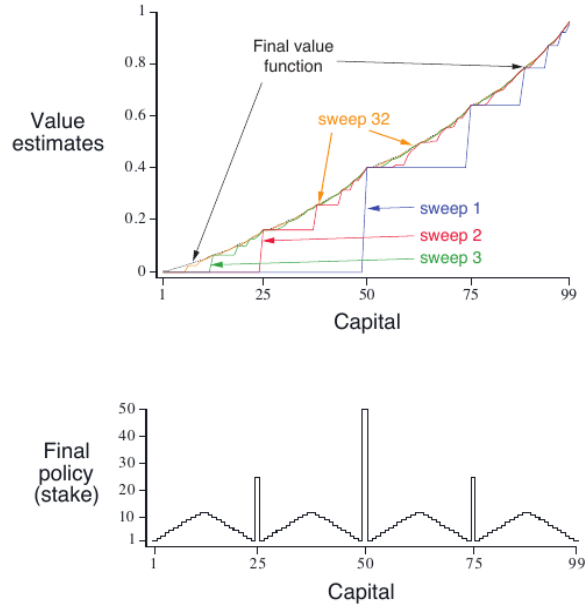
Algorithm parameter: a small threshold  $\theta > 0$  determining accuracy of estimation  
Initialize  $V(s)$ , for all  $s \in \mathcal{S}^+$ , arbitrarily except that  $V(\text{terminal}) = 0$

Loop:

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

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

**Example 4.3: Gambler’s Problem** A gambler has the opportunity to make bets on the outcomes of a sequence of coin flips. If the coin comes up heads, he wins as many dollars as he has staked on that flip; if it is tails, he loses his stake. The game ends when the gambler wins by reaching his goal of \$100, or loses by running out of money. On each flip, the gambler must decide what portion of his capital to stake, in integer numbers of dollars. This problem can be formulated as an undiscounted, episodic, finite MDP. The state is the gambler’s capital,  $s \in \{1, 2, \dots, 99\}$  and the actions are stakes,  $a \in \{0, 1, \dots, \min(s, 100 - s)\}$ . The reward is zero on all transitions except those on which the gambler reaches his goal, when it is +1. The state-value function then gives the probability of winning from each state. A policy is a mapping from levels of capital to stakes. The optimal policy maximizes the probability of reaching the goal. Let  $p_h$  denote the probability of the coin coming up heads. If  $p_h$  is known, then the entire problem is known and it can be solved, for instance, by value iteration. Figure 4.3 shows the change in the value function over successive sweeps of value iteration, and the final policy found, for the case of  $p_h = 0.4$ . This policy is optimal, but not unique. In fact, there is a whole family of optimal policies, all corresponding to ties for the argmax action selection with respect to the optimal value function. Can you guess what the entire family looks like?



**Figure 4.3:** The solution to the gambler’s problem for  $p_h = 0.4$ . The upper graph shows the value function found by successive sweeps of value iteration. The lower graph shows the final policy.

**Example 4.2: Jack’s Car Rental** Jack manages two locations for a nationwide car rental company. Each day, some number of customers arrive at each location to rent cars. If Jack has a car available, he rents it out and is credited \$10 by the national company. If he is out of cars at that location, then the business is lost. Cars become available for renting the day after they are returned. To help ensure that cars are available where they are needed, Jack can move them between the two locations overnight, at a cost of \$2 per car moved. We assume that the number of cars requested and returned at each location are Poisson random variables, meaning that the probability that the number is  $n$  is  $\frac{\lambda^n}{n!} e^{-\lambda}$ , where  $\lambda$  is the expected number. Suppose  $\lambda$  is 3 and 4 for rental requests at the first and second locations and 3 and 2 for returns. To simplify the problem slightly, we assume that there can be no more than 20 cars at each location (any additional cars are returned to the nationwide company, and thus disappear from the problem) and a maximum of five cars can be moved from one location to the other in one night. We take the discount rate to be  $\gamma = 0.9$  and formulate this as a continuing finite MDP, where the time steps are days, the state is the number of cars at each location at the end of the day, and the actions are the net numbers of cars moved between the two locations overnight. Figure 4.2 shows the sequence of policies found by policy iteration starting from the policy that never moves any cars.