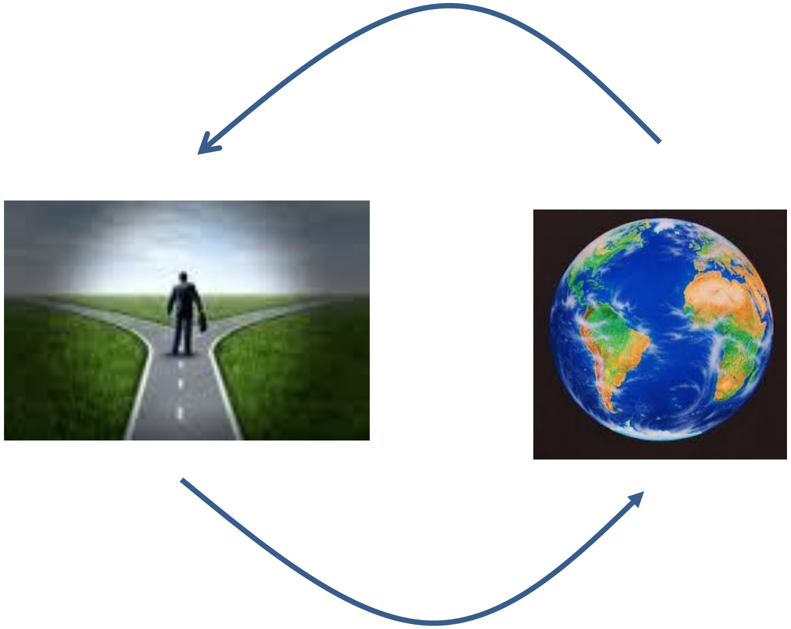
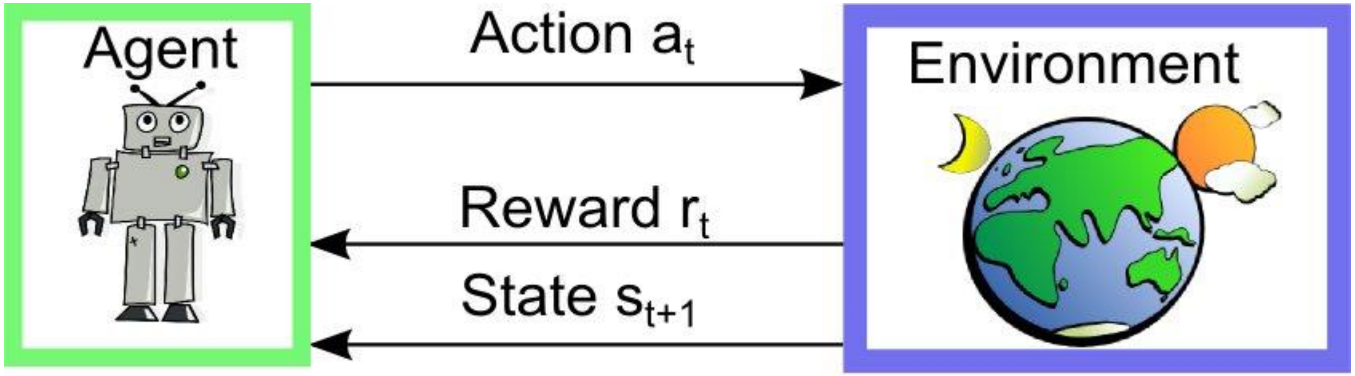
Reinforcement Learning 1

Agent learning to act in an unknown environment



Reinforcement Learning Setup

S_t



Background and setup

- The environment is initially unknown or partially known
- It is also stochastic, the agent cannot fully predict what will happen next

- What is a ‘good’ action to select under these conditions?
- Animals learning seeks to maximize their reward

Background and setup

- The reward, or some of it, can come at the end of a long sequence
-
- Finding good actions that will lead to high overall expected reward

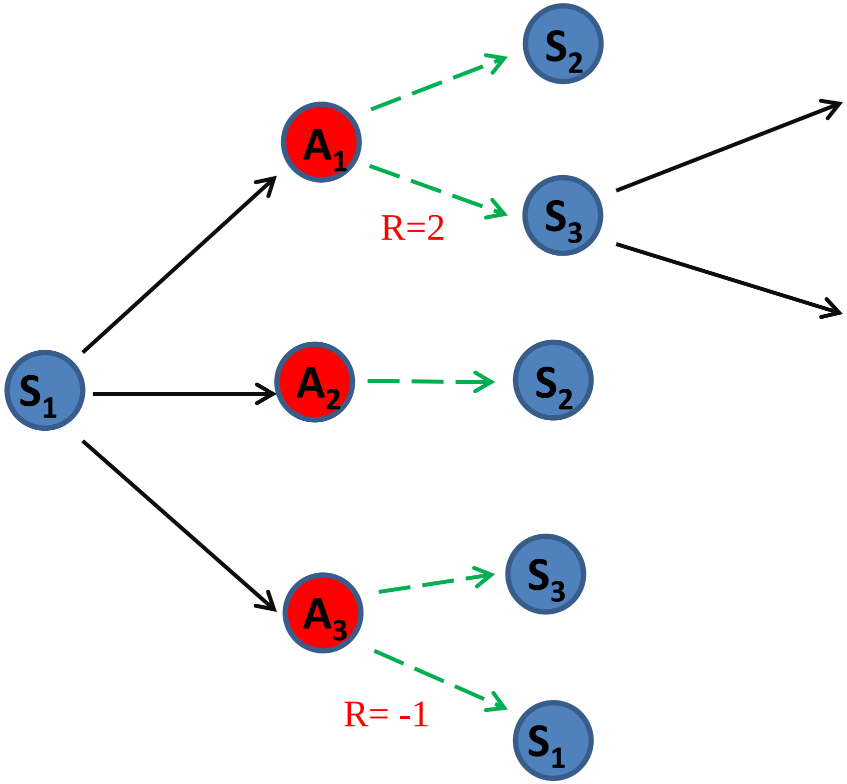
Formal Setup

- The agent is in one of a set of *states*, {S_1, S_2,…S_n }

- At each state, it can take an action from a set of available *actions* {A_1, A_2,…A_k}

- From state S_i taking action A_j –
- A new state S_j and a possible reward

Stochastic transitions



The consequence of an action:

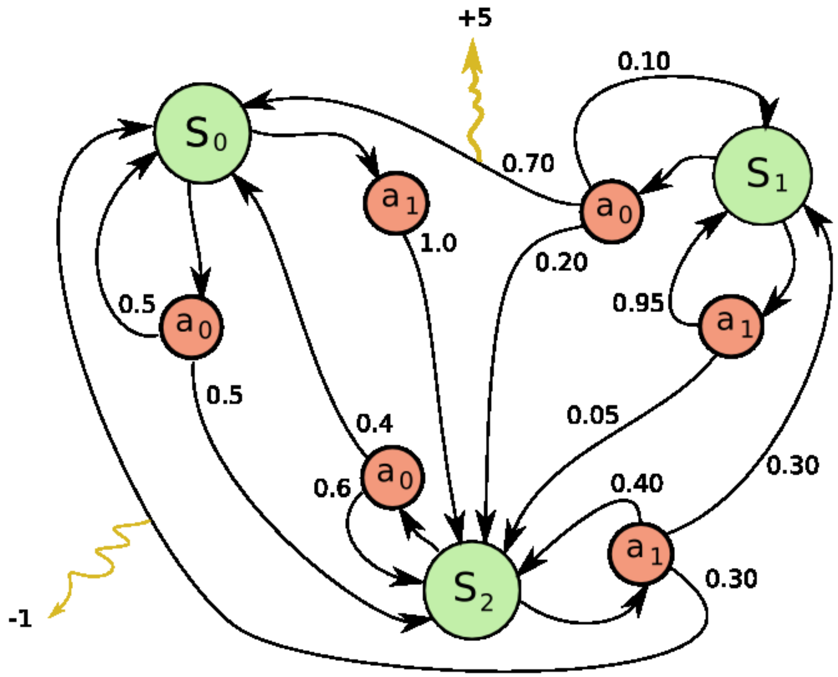
- (S,A) → (S', R)
- Governed by:
- **P(S' | S, A)**
- **P(R | (S, A, S')**
- These probabilities are properties of the world. (‘Contingencies’)
- An assumption that the transitions are *Markovian*

Policy

- The goal is to learn a policy π: S → A
- The policy determines the future of the agent:

$$S_1 \xrightarrow{\pi} A_1 \longrightarrow S_2 \xrightarrow{\pi} A_2 \longrightarrow S_3 \xrightarrow{\pi} A_3 \longrightarrow$$

MDP with 3 states and 2 actions



In this graph, for each state we draw all the actions of this states, so actions can appear multiple times. This is necessary, since transitions depend on the pair (s,a). The yellow arrows point here to Rewards.

Model-based RL

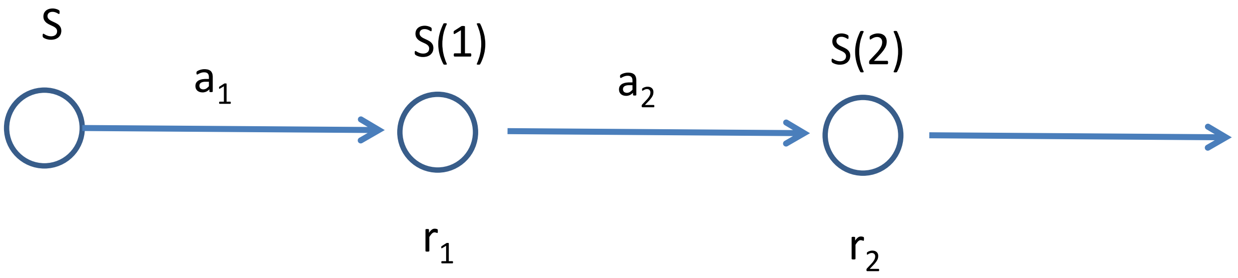
- Model-based methods:
- We assume that:

- P(S' | S, A)
- P(R | (S, A, S')

- Are known and can be used in the planning

- Model-free methods
- The ‘contingencies’ are not known
- Need to be learned by exploration as a part of policy learning

Step 1 defining $V_{\pi}(S)$



- Start from S and just follow the policy π
- We find ourselves in state S(1) and reward r_r etc.

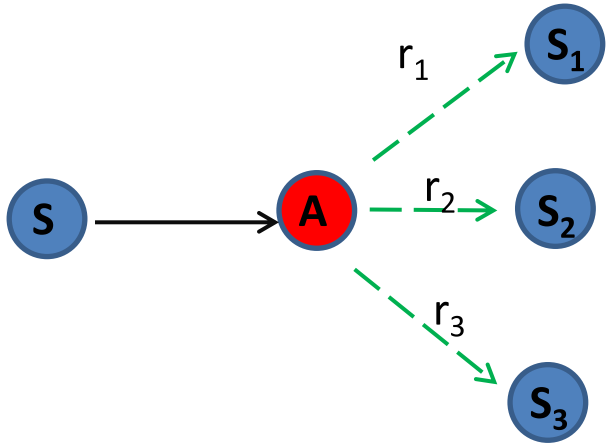
- $V_{\pi}(S) = < r_1 + \gamma r_2 + \gamma^2 r_3 + \dots >$

- The expected (discounted) reward from S on.

Step 2: equations for V(S)

- $V_\pi(S) = < r_1 + \gamma r_2 + \gamma^2 r_3 + \dots >$
- $= V_\pi(S) = < r_1 + \gamma (r_2 + \gamma r_3 + \dots) >$
- $= < r_1 + \gamma V(S') >$
- These are equations relating V(S) for different states.
- Next write the explicitly in terms of the known parameters (contingencies):

Equations for V(S)



- $V_\pi(S) = < r_1 + \gamma V_\pi(S') >$
- $V_\pi(S) = \Sigma(S')\, p(S'|S,a)\ [r(S, a, S') + \gamma V_\pi(S')\ ]$
-
- E.g.:
- $V_\pi(S) = [\ 0.2\ (r_1 + \gamma V_\pi(S_1)\ + 0.5\ (r_2 + \gamma V_\pi(S_2)\ + 0.3\ (r_3 + \gamma V_\pi(S_3)\ \ ]$
- Linear equations, the unknowns are $V(S_i)$

Improving the Policy

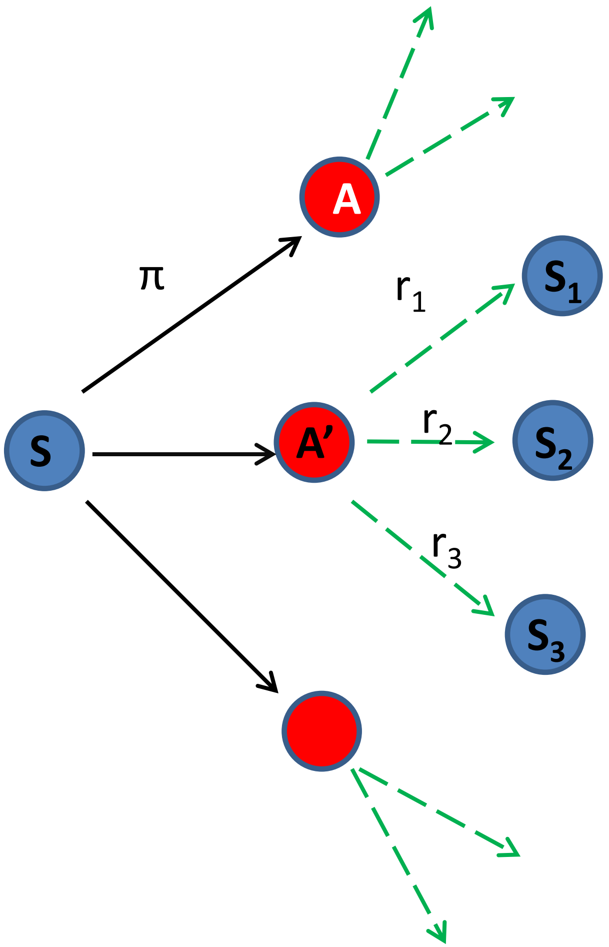
- Given the policy π, we can find the values $V_{\pi}(S)$
- Solving the linear equations, can do this iteratively
- Guaranteed convergence, the system is strongly diagonally dominant.

- Given V(S), we can improve the policy:

$$\pi(s) = \arg\max_a \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V(s') \right]$$

- We can combine these steps to find the optimal policy

Improving the policy

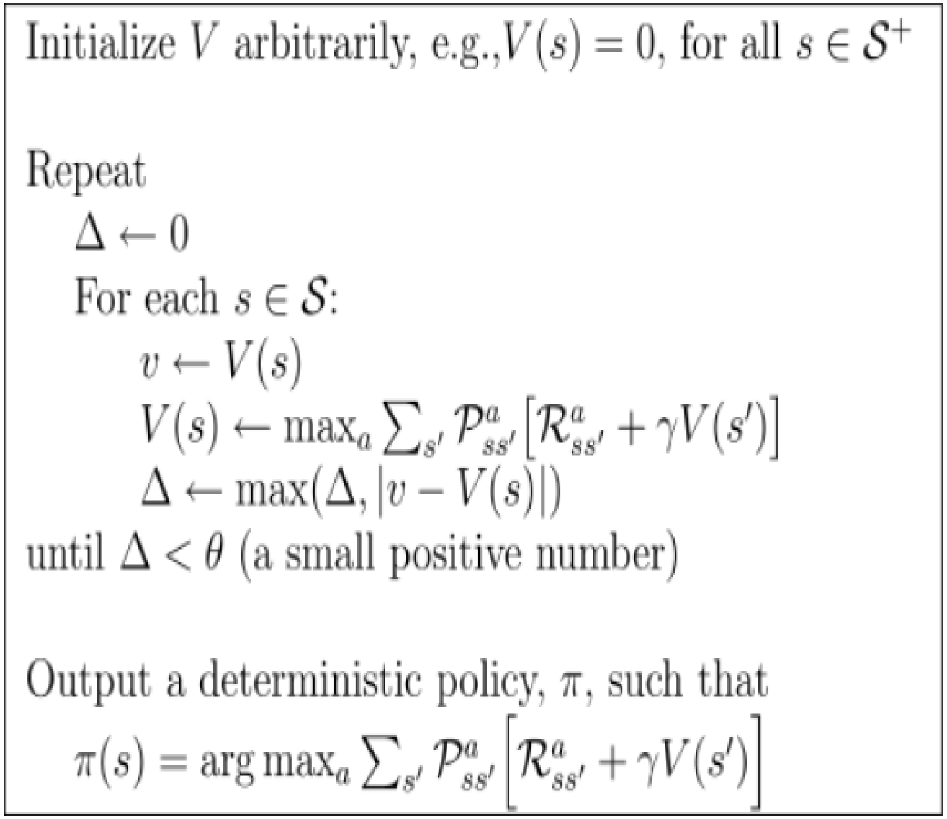


Value Iteration

$$\begin{aligned} V_{k+1}(s) &= \max_a E\left\{r_{t+1} + \gamma V_k(s_{t+1}) \mid s_t = s, a_t = a\right\}, \\ &= \max_a \sum_{s'} \mathcal{P}^a_{ss'}\left[\mathcal{R}^a_{ss'} + \gamma V_k(s')\right], \end{aligned}$$

learning V and π when the ‘contingencies’ are known:

Value Iteration Algorithm



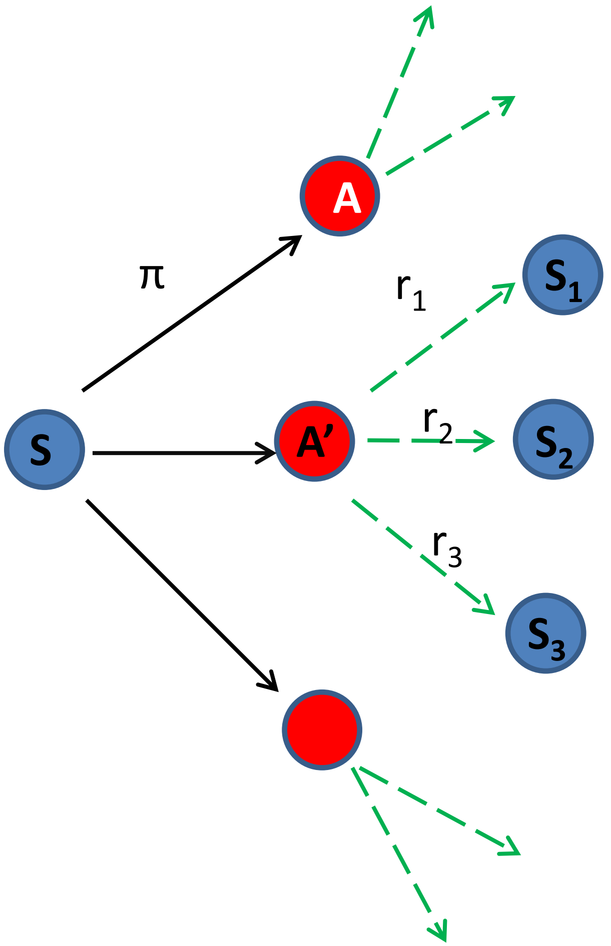
```
Initialize $V$ arbitrarily, e.g.,$V(s) = 0$, for all $s \in \mathcal{S}^+$

Repeat
    $\Delta \leftarrow 0$
    For each $s \in \mathcal{S}$:
        $v \leftarrow V(s)$
        $V(s) \leftarrow \max_a \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V(s')\right]$
        $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$ (a small positive number)

Output a deterministic policy, $\pi$, such that
    $\pi(s) = \arg\max_a \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V(s')\right]$
```

Value iteration is used in the problem set

At the optimal policy:



The action A selected by π will be the best action to choose

Bellman Equation (Dynamic Programming)

$V_{\pi}(S) = \Sigma(S')\ p(S'|S,a)\ [r(S, a, S') + \gamma V_{\pi}(S')\ ]$

$$\begin{aligned} V^*(s) &= \max_a E\left\{r_{t+1} + \gamma V^*(s_{t+1}) \mid s_t = s, a_t = a\right\} \\ &= \max_a \sum_{s'} \mathcal{P}^a_{ss'}\left[\mathcal{R}^a_{ss'} + \gamma V^*(s')\right] \qquad (4.1) \end{aligned}$$

The unknowns are $V^*(S)$, but the equations are non linear

The value iteration algorithm solves the equations