# UNIT II

Markov Decision Process, The Agent-Environment Interface , Goals and Rewards-Returns, Unified Notation for Episodic and Continuing Tasks, The Markov Property, Markov Decision Processes, Value Functions, Optimal Value Functions, Optimality and Approximation, Bellman expectation equations, Bellman optimality equations, Markov Reward Process

# Markov- Decision Process

- *MDPs are a classical formalization of sequential decision making, where actions influence not just immediate rewards, but also subsequent situations, or states, and through those future rewards.*
- *Thus MDPs involve delayed reward and the need to tradeoff immediate and delayed reward. Whereas in bandit problems we estimated the value $q_*(a)$ of each action a, in MDPs we estimate the value $q_*(s, a)$ of each action "a" in each state s, or we estimate the value $v_*(s)$ of each state given optimal action selections.*
- *These state-dependent quantities are essential to accurately assigning credit for long-term consequences to individual action selections.*

# *The Agent–Environment Interface*

- The learner and decision maker is called the agent.
- The thing it interacts with, comprising everything outside the agent, is called the environment.
- These interact continually, the agent selecting actions and the environment responding to these actions and presenting new situations to the agent.
- The environment also gives rise to rewards, special numerical values that the agent seeks to maximize over time through its choice of actions.
- *The MDP and agent together thereby give rise to a sequence or trajectory that begins like this:*

$$S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, R_3, \ldots$$

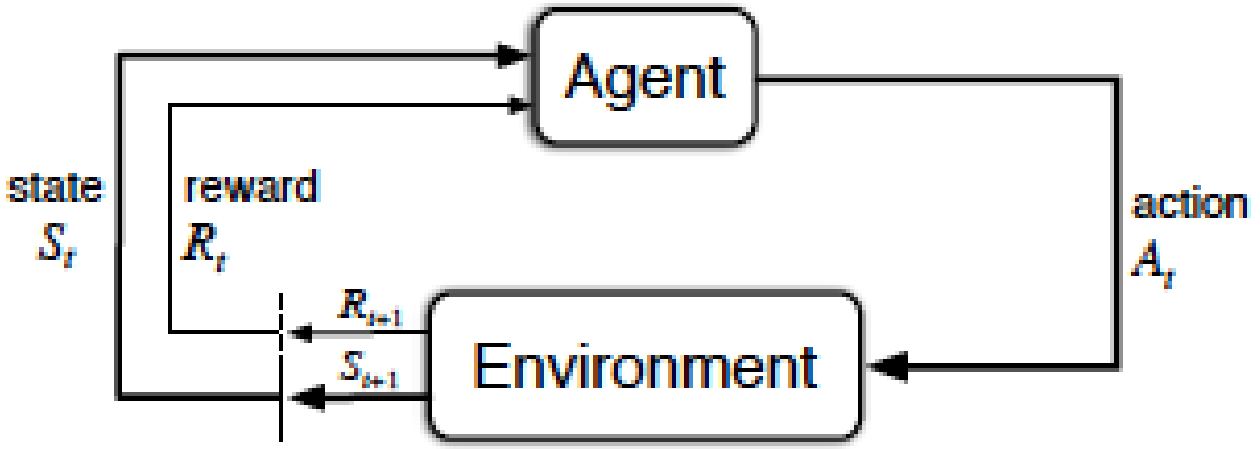
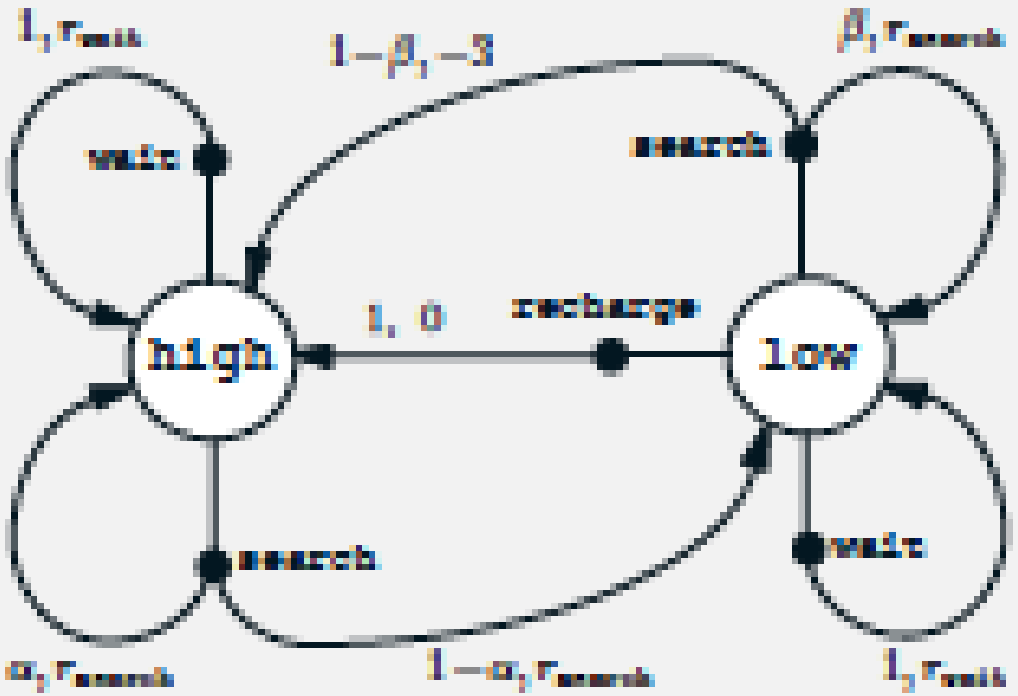



*Fig: The agent–environment interaction in a Markov decision process*

# Recycling Robot

| $s$ | $a$ | $s'$ | $p(s'\|s,a)$ | $r(s,a,s')$ |
|---|---|---|---|---|
| high | search | high | $\alpha$ | $r_{search}$ |
| high | search | low | $1-\alpha$ | $r_{search}$ |
| low | search | high | $1-\beta$ | $-3$ |
| low | search | low | $\beta$ | $r_{search}$ |
| high | wait | high | 1 | $r_{wait}$ |
| high | wait | low | 0 | - |
| low | wait | high | 0 | - |
| low | wait | low | 1 | $r_{wait}$ |
| low | recharge | high | 1 | 0 |
| low | recharge | low | 0 | - |



## Goals and Rewards

- In reinforcement learning, the purpose or goal of the agent is formalized in terms of a special signal, called the reward, passing from the environment to the agent.
- At each time step, the reward is a simple number $R_t \in \mathbb{R}$.
- The use of a reward signal to formalize the idea of a goal is one of the most distinctive features of reinforcement learning.

# Returns and Episodes

- To maximize the expected return, where the return, denoted Gt, is defined as some specific function of the reward sequence.
- In the simplest case the return is the sum of the rewards:

$$G_t \doteq R_{t+1} + R_{t+2} + R_{t+3} + \cdots + R_T$$

where T is a final time step

*Discounting*

According to this approach, the agent tries to select actions so that the sum of the discounted rewards it receives over the future is maximized. In particular, it chooses $A_t$ to maximize the expected discounted return:

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1},$$

where $\gamma$ is a parameter, $0 \leq \gamma \leq 1$, called the *discount rate*.

Returns at successive time steps are related to each other in a way that is important for the theory and algorithms of reinforcement learning:

$$\begin{aligned} G_t &\doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \cdots \\ &= R_{t+1} + \gamma\big(R_{t+2} + \gamma R_{t+3} + \gamma^2 R_{t+4} + \cdots\big) \\ &= R_{t+1} + \gamma G_{t+1} \end{aligned} \tag{3.9}$$

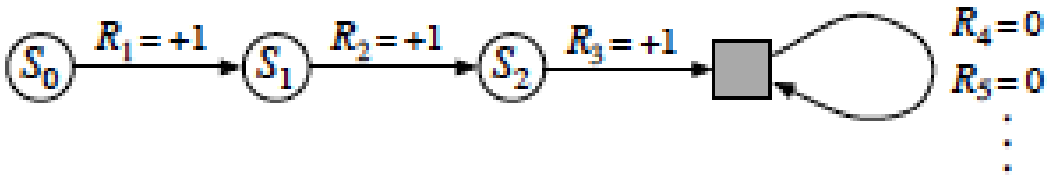
Note that this works for all time steps t < T, even if termination occurs at t + 1, if we define GT = 0. This often makes it easy to compute returns from reward sequences. For example, if the reward is a constant +1, then the return is

$$G_t = \sum_{k=0}^{\infty} \gamma^k = \frac{1}{1-\gamma}.$$

# *Unified Notation for Episodic and Continuing Tasks*

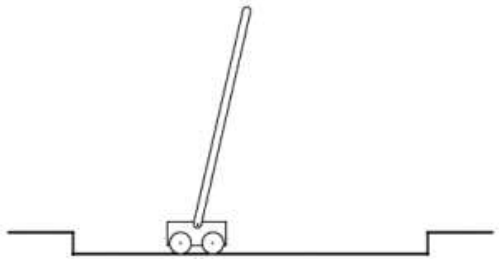
- To obtain a single notation that covers both episodic and continuing tasks. We have defined the return as a sum over a finite number of terms in one case and as a sum over an infinite number of terms in the other.
- These two can be unified by considering episode termination to be the entering of a special absorbing state that transitions only to itself and that generates only rewards of zero.

For example, consider the state transition diagram:




$$G_t \doteq \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k,$$

# An Example



Avoid **failure:** the pole falling beyond a critical angle or the cart hitting end of track.

As an **episodic task** where episode ends upon failure:

reward = +1 for each step before failure

$\Rightarrow$ return = number of steps before failure

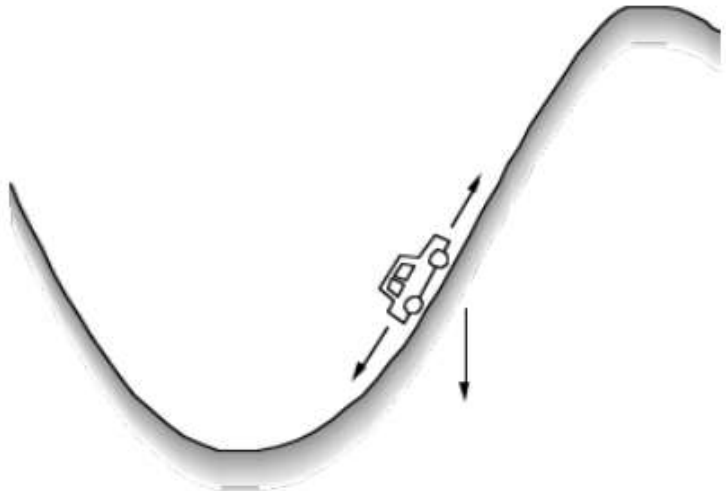
As a **continuing task** with discounted return:

reward = −1 upon failure; 0 otherwise

$\Rightarrow$ return = $-\gamma^k$, for $k$ steps before failure

In either case, return is maximized by avoiding failure for as long as possible.

# Another Example



Get to the top of the hill
as quickly as possible.

reward = –1 for each step where **not** at top of hill

⇒ return = – number of steps before reaching top of hill

Return is maximized by minimizing
number of steps to reach the top of the hill.

# The Markov Property

- A property of environments and their state signals that is of particular interest, called the Markov property.
- the state --> whatever information is available to the agent
- the state is given by some preprocessing system that is nominally part of the environment.
- A state signal that summarizes past sensations compactly, yet in such a way that all relevant information is retained.
- This normally requires more than the immediate sensations, but never more than the complete history of all past sensations. A state signal that succeeds in retaining all relevant information is said to be *Markov*, or to have *the Markov property.*

The dynamics can be defined only by specifying the complete probability distribution:

$$Pr\{s_{t+1} = s', r_{t+1} = r \mid s_t, a_t, r_t, s_{t-1}, a_{t-1}, \ldots, r_1, s_0, a_0\},$$

for all $s'$, $r$, and all possible values of the past events: $s_t, a_t, r_t, \ldots, r_1, s_0, a_0$. If the state signal has the *Markov property*, on the other hand, then the environment's response at $t+1$ depends only on the state and action representations at $t$, in which case the environment's dynamics can be defined by specifying only

$$Pr\{s_{t+1} = s', r_{t+1} = r \mid s_t, a_t\},$$

If an environment has the Markov property, then its one-step dynamics enable us to predict the next state and expected next reward given the current state and action.

## Policies and Value Functions

- Value functions—>functions of states (or of state–action pairs) that estimate how good it is for the agent to be in a given state (or how good it is to perform a given action in a given state).
- how good means future rewards that can be expected, or, to be precise, in terms of expected return
- Value functions are defined with respect to particular ways of acting, called policies.
- A policy is a mapping from states to probabilities of selecting each possible action.

The value function of a state s under a policy $\pi$→ is the expected return when starting in *s* and following $\pi$ thereafter. The *state-value function for policy* $\pi$ is

$$v_\pi(s) \doteq \mathbb{E}_\pi[G_t \mid S_t = s] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \,\middle|\, S_t = s\right], \text{ for all } s \in \mathcal{S},$$

$\mathbb{E}_\pi[\cdot]$ ➪ *denotes the expected value of a random variable given that the agent follows policy* $\pi$*, and t is any time step.*

*The action-value function for policy* $\pi$

$$q_\pi(s,a) \doteq \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \,\middle|\, S_t = s, A_t = a\right]$$

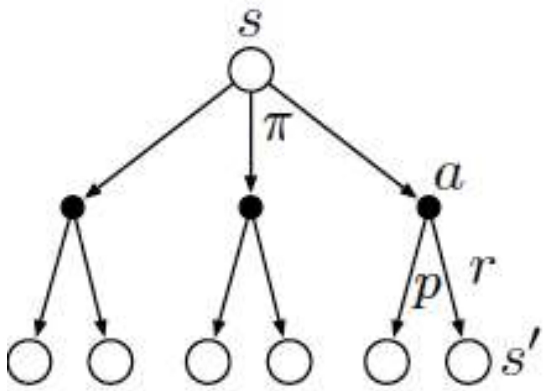
- *The value functions $v_\pi$ and $q_\pi$ can be estimated from experience.*
- *For example, if an agent follows policy $\pi$ and maintains an average, for each state encountered, of the actual returns that have followed that state, then the average will converge to the state's value, $v_\pi(s)$, as the number of times that state is encountered approaches infinity.*
- *If separate averages are kept for each action taken in each state, then these averages will similarly converge to the action values, $q_\pi(s, a)$.*
- *We call estimation methods of this kind Monte Carlo methods because they involve averaging over many random samples of actual returns*

For any policy **π** and any state s, the following consistency condition holds between the value of s and the value of its possible successor states:

$$
\begin{aligned}
v_\pi(s) &\doteq \mathbb{E}_\pi[G_t \mid S_t = s] \\
&= \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} \mid S_t = s] \\
&= \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r|s, a) \Big[ r + \gamma \mathbb{E}_\pi[G_{t+1} | S_{t+1} = s'] \Big] \\
&= \sum_a \pi(a|s) \sum_{s', r} p(s', r|s, a) \Big[ r + \gamma v_\pi(s') \Big], \quad \text{for all } s \in \mathcal{S},
\end{aligned}
$$

This is the Bellman equation for $v_{\boldsymbol{\pi}}$

- It expresses a relationship between the value of a state and the values of its successor states.



Backup diagram for $v_\pi$

# Optimal Policies and Optimal Value Functions

- Value functions define a partial ordering over policies.
- A policy $\pi$ is defined to be better than or equal to a policy $\pi'$ if its expected return is greater than or equal to that of $\pi'$ for all states.
- In other words, $\pi \geq \pi'$ if and only if $v_{\pi}(s) \geq v_{\pi'}(s)$ for all s $\epsilon$S.
- There is always at least one policy that is better than or equal to all other policies. This is an optimal policy.
- Although there may be more than one, we denote all the optimal policies by $\pi_*$.
- They share the same state-value function, called the optimal state-value function, denoted $v_*$, and defined a

$$v_*(s) \doteq \max_{\pi} v_{\pi}(s),$$

for all $s \in \mathcal{S}$.

Optimal policies also share the same *optimal action-value function*, denoted $q_*$, and defined as

$$q_*(s,a) \doteq \max_\pi q_\pi(s,a), \tag{3.16}$$

for all $s \in \mathcal{S}$ and $a \in \mathcal{A}(s)$. For the state–action pair $(s,a)$, this function gives the expected return for taking action $a$ in state $s$ and thereafter following an optimal policy. Thus, we can write $q_*$ in terms of $v_*$ as follows:

$$q_*(s,a) = \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1}) \mid S_t = s, A_t = a]. \tag{3.17}$$

- Because $v_*$ is the value function for a policy, it must satisfy the self-consistency condition given by the Bellman equation for state values.
- Because it is the optimal value function, however, $v_*$'s consistency condition can be written in a special form without reference to any specific policy.
- This is the Bellman equation for $v_*$, or the Bellman optimality equation.

- Intuitively, the Bellman optimality equation expresses the fact that the value of a state under an optimal policy must equal the expected return for the best action from that state:

$$
\begin{aligned}
v_*(s) &= \max_{a \in \mathcal{A}(s)} q_{\pi_*}(s,a) \\
&= \max_a \mathbb{E}_{\pi_*}[G_t \mid S_t = s, A_t = a] \\
&= \max_a \mathbb{E}_{\pi_*}[R_{t+1} + \gamma G_{t+1} \mid S_t = s, A_t = a] \\
&= \max_a \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1}) \mid S_t = s, A_t = a] \\
&= \max_a \sum_{s',r} p(s', r \mid s, a)\big[r + \gamma v_*(s')\big].
\end{aligned}
$$

The last two equations are two forms of the Bellman optimality equation for $v_*$. The Bellman optimality equation for $q_*$ is

$$
\begin{aligned}
q_*(s,a) &= \mathbb{E}\Big[R_{t+1} + \gamma \max_{a'} q_*(S_{t+1}, a') \;\Big|\; S_t = s, A_t = a\Big] \\
&= \sum_{s',r} p(s', r \mid s, a)\Big[r + \gamma \max_{a'} q_*(s', a')\Big].
\end{aligned}
$$

The Bellman optimality equation is actually a system of equations, one for each state, so if there are n states, then there are n equations in n unknowns.

If the dynamics p of the environment are known, then in principle one can solve this system of equations for $v_*$ using any one of a variety of methods for solving systems of nonlinear equations. One can solve a related set of equations for $q_*$.

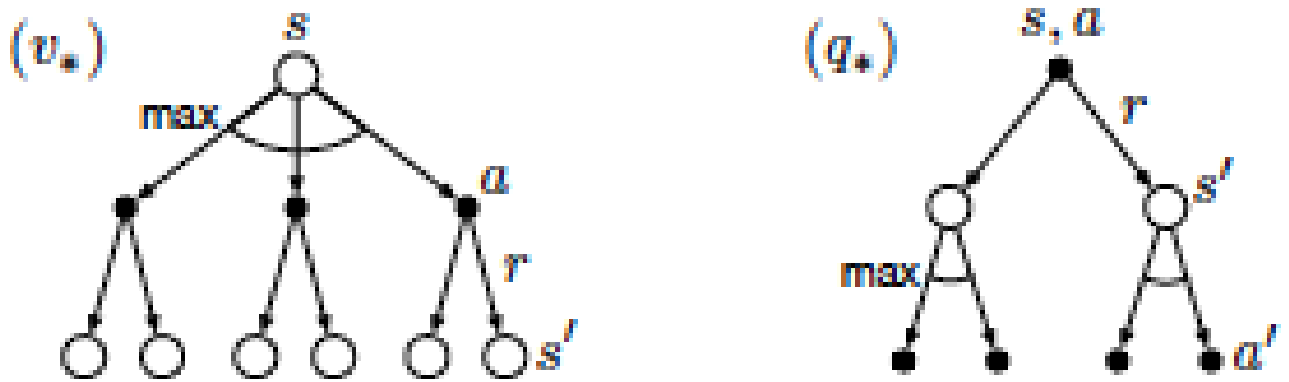



**Figure 3.4:** Backup diagrams for $v_*$ and $q_*$

# GRID WORLD PROBLEM

# GOLF AS A REINFORCEMENT LEARNING

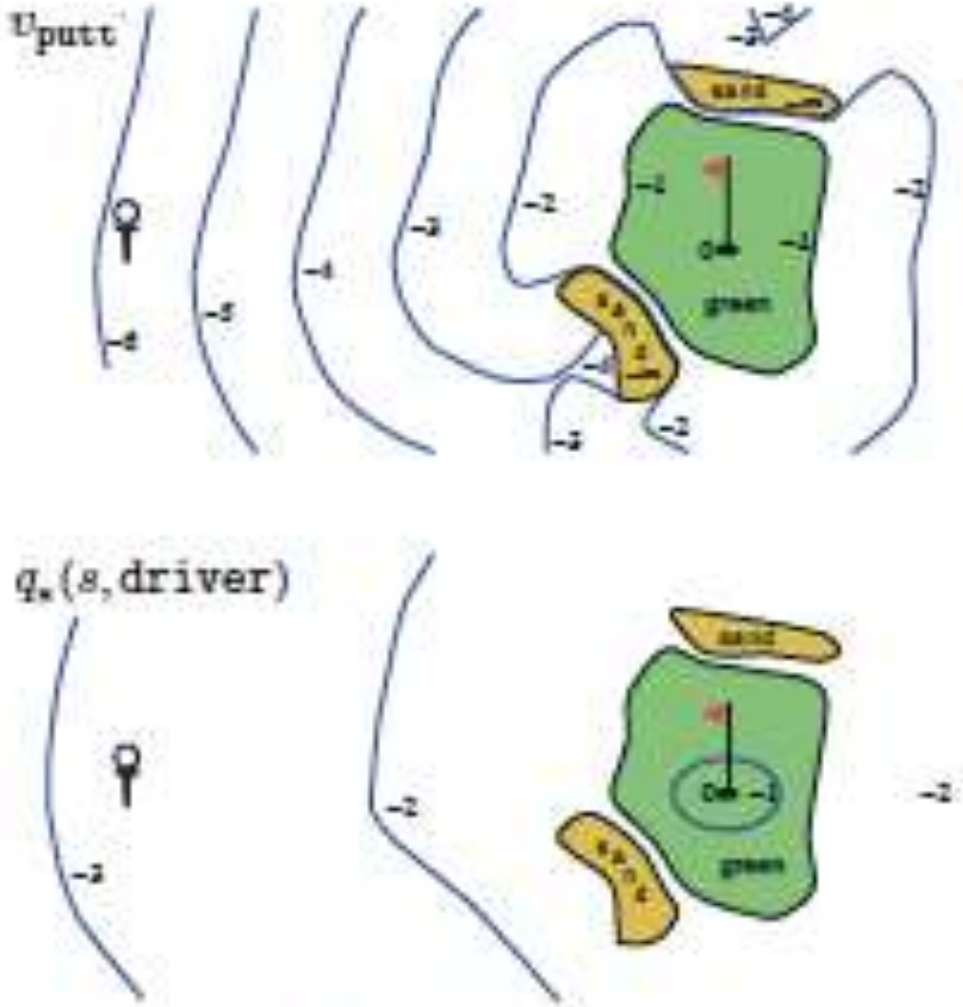




**Figure 3.3:** A golf example: the state-value function for putting (upper) and the optimal action-value function for using the driver (lower). ■

# Bellman Optimality Equations for the Recycling Robot

Abbreviate the states `high` and `low`, and the actions `search`, `wait`, and `recharge` respectively by `h`, `l`, `s`, `w`, and `re`. Because there are only two states, the Bellman optimality equation consists of two equations.

$$
\begin{aligned}
v_*(\mathtt{h}) &= \max\left\{ \begin{array}{l} p(\mathtt{h}|\mathtt{h},\mathtt{s})[r(\mathtt{h},\mathtt{s},\mathtt{h}) + \gamma v_*(\mathtt{h})] + p(\mathtt{l}|\mathtt{h},\mathtt{s})[r(\mathtt{h},\mathtt{s},\mathtt{l}) + \gamma v_*(\mathtt{l})], \\ p(\mathtt{h}|\mathtt{h},\mathtt{w})[r(\mathtt{h},\mathtt{w},\mathtt{h}) + \gamma v_*(\mathtt{h})] + p(\mathtt{l}|\mathtt{h},\mathtt{w})[r(\mathtt{h},\mathtt{w},\mathtt{l}) + \gamma v_*(\mathtt{l})] \end{array} \right\} \\
&= \max\left\{ \begin{array}{l} \alpha[r_{\mathtt{s}} + \gamma v_*(\mathtt{h})] + (1-\alpha)[r_{\mathtt{s}} + \gamma v_*(\mathtt{l})], \\ 1[r_{\mathtt{w}} + \gamma v_*(\mathtt{h})] + 0[r_{\mathtt{w}} + \gamma v_*(\mathtt{l})] \end{array} \right\} \\
&= \max\left\{ \begin{array}{l} r_{\mathtt{s}} + \gamma[\alpha v_*(\mathtt{h}) + (1-\alpha) v_*(\mathtt{l})], \\ r_{\mathtt{w}} + \gamma v_*(\mathtt{h}) \end{array} \right\}.
\end{aligned}
$$

Following the same procedure for $v_*(\mathtt{l})$ yields the equation

$$
v_*(\mathtt{l}) = \max\left\{ \begin{array}{l} \beta r_{\mathtt{s}} - 3(1-\beta) + \gamma[(1-\beta) v_*(\mathtt{h}) + \beta v_*(\mathtt{l})], \\ r_{\mathtt{w}} + \gamma v_*(\mathtt{l}), \\ \gamma v_*(\mathtt{h}) \end{array} \right\}.
$$

For any choice of $r_{\mathtt{s}}$, $r_{\mathtt{w}}$, $\alpha$, $\beta$, and $\gamma$, with $0 \leq \gamma < 1$, $0 \leq \alpha, \beta \leq 1$, there is exactly one pair of numbers, $v_*(\mathtt{h})$ and $v_*(\mathtt{l})$, that simultaneously satisfy these two nonlinear equations. ■

## Optimality and Approximation

- Clearly, an agent that learns an optimal policy has done very well, but in practice this rarely happens.
- For the kinds of tasks in which we are interested, optimal policies can be generated only with extreme computational cost.
- The memory available is also an important constraint. A large amount of memory is often required to build up approximations of value functions, policies, and models.
- In tasks with small, finite state sets, it is possible to form these approximations using arrays or tables with one entry for each state