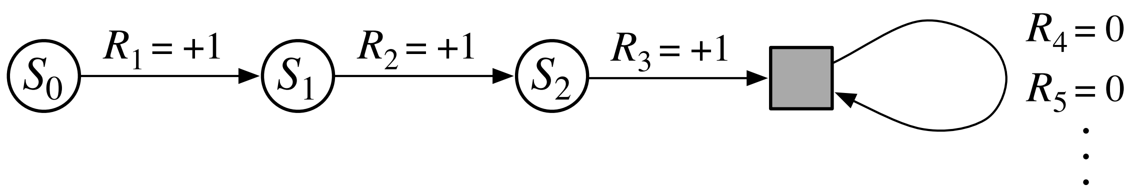
# A Trick to Unify Notation for Returns

- In episodic tasks, we number the time steps of each episode starting from zero.
- We usually do not have to distinguish between episodes, so instead of writing $S_{t,j}$ for states in episode $j$, we write just $S_t$
- Think of each episode as ending in an absorbing state that always produces reward of zero:




- We can cover <u>all</u> cases by writing $G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$,

where $\gamma$ can be 1 only if a zero reward absorbing state is always reached.

# Rewards and returns

- The objective in RL is to maximize long-term future reward
- That is, to choose $A_t$ so as to maximize $R_{t+1}, R_{t+2}, R_{t+3}, \ldots$
- But what exactly should be maximized?
- The discounted *return* at time t:

the *discount rate*

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \cdots \qquad \gamma \in [0,1)$$

| $\gamma$ | Reward sequence | Return |
|---|---|---|
| 0.5(or any) | 1 0 0 0… | 1 |
| 0.5 | 0 0 2 0 0 0… | 0.5 |
| 0.9 | 0 0 2 0 0 0… | 1.62 |
| 0.5 | -1 2 6 3 2 0 0 0… | 2 |

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \cdots \qquad \gamma \in [0, 1)$$

- Suppose $\gamma = 0.5$ and the reward sequence is

$$R_1 = 1, R_2 = 6, R_3 = -12, R_4 = 16, \text{ then zeros for } R_5 \text{ and later}$$

- What are the following returns?

$$G_4 = 0 \quad G_3 =$$

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \cdots \qquad \gamma \in [0,1)$$

- Suppose $\gamma = 0.5$ and the reward sequence is

$$R_1 = 1, R_2 = 6, R_3 = -12, R_4 = 16, \text{ then zeros for } R_5 \text{ and later}$$

- What are the following returns?

$$G_4 = 0 \quad G_3 = 16 \quad G_2 =$$

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \cdots \qquad \gamma \in [0,1)$$

- Suppose $\gamma = 0.5$ and the reward sequence is

$$R_1 = 1, R_2 = 6, R_3 = -12, R_4 = 16, \text{ then zeros for } R_5 \text{ and later}$$

- What are the following returns?

$$G_4 = 0 \quad G_3 = 16 \quad G_2 = -4 \quad G_1 =$$

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \cdots \qquad \gamma \in [0,1)$$

- Suppose $\gamma = 0.5$ and the reward sequence is

$$R_1 = 1, R_2 = 6, R_3 = -12, R_4 = 16, \text{ then zeros for } R_5 \text{ and later}$$

- What are the following returns?

$$G_4 = 0 \quad G_3 = 16 \quad G_2 = -4 \quad G_1 = 4 \quad G_0 =$$

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \cdots \qquad \gamma \in [0, 1)$$

- Suppose $\gamma = 0.5$ and the reward sequence is

$$R_1 = 1, R_2 = 6, R_3 = -12, R_4 = 16, \text{ then zeros for } R_5 \text{ and later}$$

- What are the following returns?

$$G_4 = 0 \quad G_3 = 16 \quad G_2 = -4 \quad G_1 = 4 \quad G_0 = 3$$

- Suppose $\gamma = 0.5$ and the reward sequence is all 1s.

$$G =$$

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \cdots \qquad \gamma \in [0, 1)$$

- Suppose $\gamma = 0.5$ and the reward sequence is

$$R_1 = 1, R_2 = 6, R_3 = -12, R_4 = 16, \text{ then zeros for } R_5 \text{ and later}$$

- What are the following returns?

$$G_4 = 0 \quad G_3 = 16 \quad G_2 = -4 \quad G_1 = 4 \quad G_0 = 3$$

- Suppose $\gamma = 0.5$ and the reward sequence is all 1s.

$$G = \frac{1}{1 - \gamma}$$

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \cdots \qquad \gamma \in [0, 1)$$

- Suppose $\gamma = 0.5$ and the reward sequence is

$$R_1 = 1, R_2 = 6, R_3 = -12, R_4 = 16, \text{ then zeros for } R_5 \text{ and later}$$

- What are the following returns?

$$G_4 = 0 \quad G_3 = 16 \quad G_2 = -4 \quad G_1 = 4 \quad G_0 = 3$$

- Suppose $\gamma = 0.5$ and the reward sequence is all 1s.

$$G = \frac{1}{1-\gamma} = 2$$

# What we learned so far

- Finite Markov decision processes!
  - States, actions, and rewards
  - And returns
  - And time, discrete time
  - They capture essential elements of life — state, causality
- The goal is to optimize expected returns
  - returns are *discounted sums* of *future* rewards
- Thus we are interested in *values* — expected returns

# 4 value functions

| | state values | action values |
|---|---|---|
| prediction | $v_\pi$ | $q_\pi$ |
| control | $v_*$ | $q_*$ |

- All theoretical objects, mathematical ideals (expected values)
- Distinct from their estimates:

$$V_t(s) \qquad Q_t(s, a)$$

# Values are *expected* returns

- The value of a state, given a policy:

$$v_\pi(s) = \mathbb{E}\{G_t \mid S_t = s, A_{t:\infty} \sim \pi\} \qquad v_\pi : \mathcal{S} \to \Re$$

- The value of a state-action pair, given a policy:

$$q_\pi(s,a) = \mathbb{E}\{G_t \mid S_t = s, A_t = a, A_{t+1:\infty} \sim \pi\} \qquad q_\pi : \mathcal{S} \times \mathcal{A} \to \Re$$

- The optimal value of a state:

$$v_*(s) = \max_\pi v_\pi(s) \qquad v_* : \mathcal{S} \to \Re$$

- The optimal value of a state-action pair:

$$q_*(s,a) = \max_\pi q_\pi(s,a) \qquad q_* : \mathcal{S} \times \mathcal{A} \to \Re$$

- Optimal policy: $\pi_*$ is an optimal policy if and only if

$$\pi_*(a|s) > 0 \text{ only where } q_*(s,a) = \max_b q_*(s,b) \qquad \forall s \in \mathcal{S}$$

  - in other words, $\pi_*$ is optimal iff it is *greedy* wrt $q_*$

# Bellman Equation for a Policy π

The basic idea:

$$
\begin{aligned}
G_t &= R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \cdots \\
&= R_{t+1} + \gamma \left( R_{t+2} + \gamma R_{t+3} + \gamma^2 R_{t+4} + \cdots \right) \\
&= R_{t+1} + \gamma G_{t+1}
\end{aligned}
$$

So:

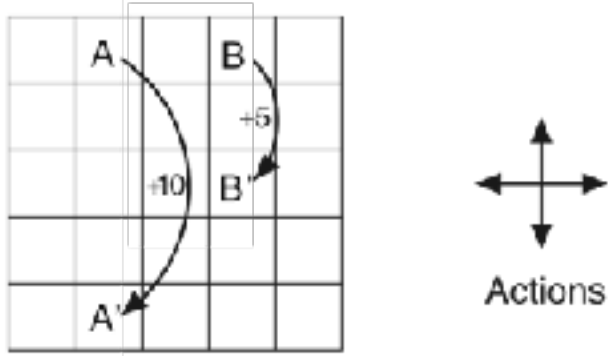
$$
\begin{aligned}
v_\pi(s) &= E_\pi \left\{ G_t \mid S_t = s \right\} \\
&= E_\pi \left\{ R_{t+1} + \gamma v_\pi\left(S_{t+1}\right) \mid S_t = s \right\}
\end{aligned}
$$

Or, without the expectation operator:

$$
v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a) \Big[ r + \gamma v_\pi(s') \Big]
$$

# Gridworld

- Actions: `north`, `south`, `east`, `west`; deterministic.
- If would take agent off the grid: no move but reward = –1
- Other actions produce reward = 0, except actions that move agent out of special states A and B as shown.




(a)

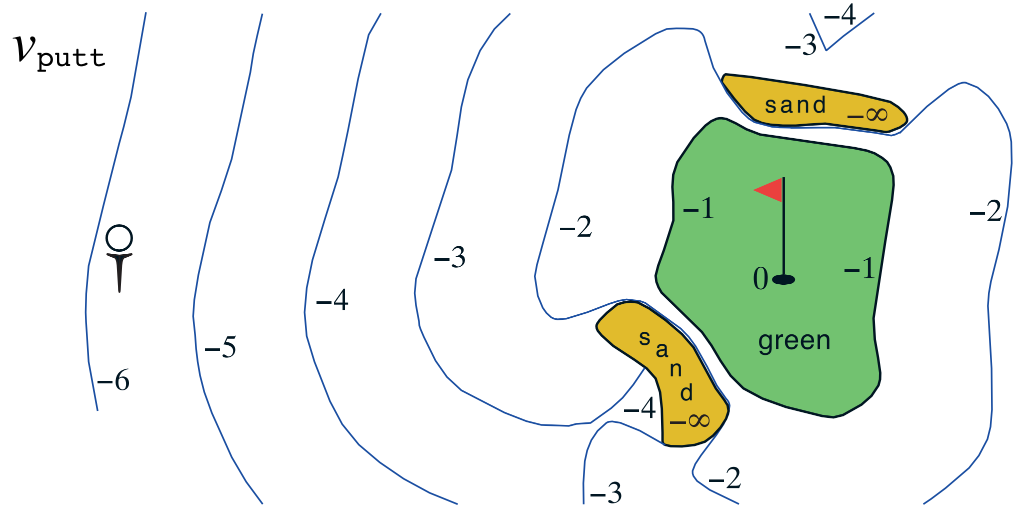
| | | | | |
|---|---|---|---|---|
| 3.3 | 8.8 | 4.4 | 5.3 | 1.5 |
| 1.5 | 3.0 | 2.3 | 1.9 | 0.5 |
| 0.1 | 0.7 | 0.7 | 0.4 | -0.4 |
| -1.0 | -0.4 | -0.4 | -0.6 | -1.2 |
| -1.9 | -1.3 | -1.2 | -1.4 | -2.0 |

(b)

State-value function for equiprobable random policy; $\gamma = 0.9$

# Golf

- State is ball location
- Reward of –1 for each stroke until the ball is in the hole
- Value of a state?
- Actions:
  - `putt` (use putter)
  - `driver` (use driver)
- `putt` succeeds anywhere on the green



# Optimal Value Functions

- For finite MDPs, policies can be **partially ordered**:

$$\pi \geq \pi' \quad \text{if and only if } v_\pi(s) \geq v_{\pi'}(s) \text{ for all } s \in \mathcal{S}$$

- There are always one or more policies that are better than or equal to all the others. These are the **optimal policies**. We denote them all $\pi_*$.
- Optimal policies share the same **optimal state-value function**:

$$v_*(s) = \max_\pi v_\pi(s) \quad \text{for all } s \in \mathcal{S}$$

- Optimal policies also share the same **optimal action-value function**:

$$q_*(s,a) = \max_\pi q_\pi(s,a) \text{ for all } s \in \mathcal{S} \text{ and } a \in \mathcal{A}$$

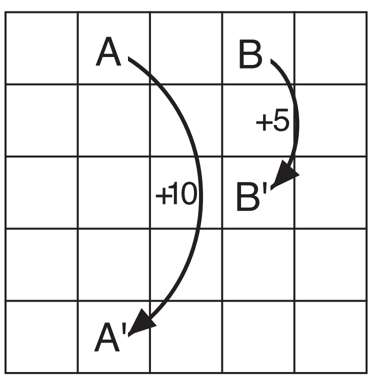
This is the expected return for taking action $a$ in state $s$ and thereafter following an optimal policy.

# Why Optimal State-Value Functions are Useful

**Any policy that is greedy with respect to $v_*$ is an optimal policy.**

Therefore, given $v_*$, one-step-ahead search produces the long-term optimal actions.

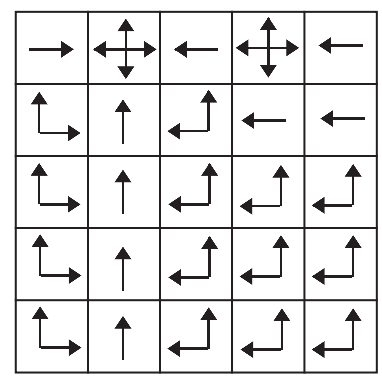
E.g., back to the gridworld:




a) gridworld

| | | | | |
|---|---|---|---|---|
| 22.0 | 24.4 | 22.0 | 19.4 | 17.5 |
| 19.8 | 22.0 | 19.8 | 17.8 | 16.0 |
| 17.8 | 19.8 | 17.8 | 16.0 | 14.4 |
| 16.0 | 17.8 | 16.0 | 14.4 | 13.0 |
| 14.4 | 16.0 | 14.4 | 13.0 | 11.7 |

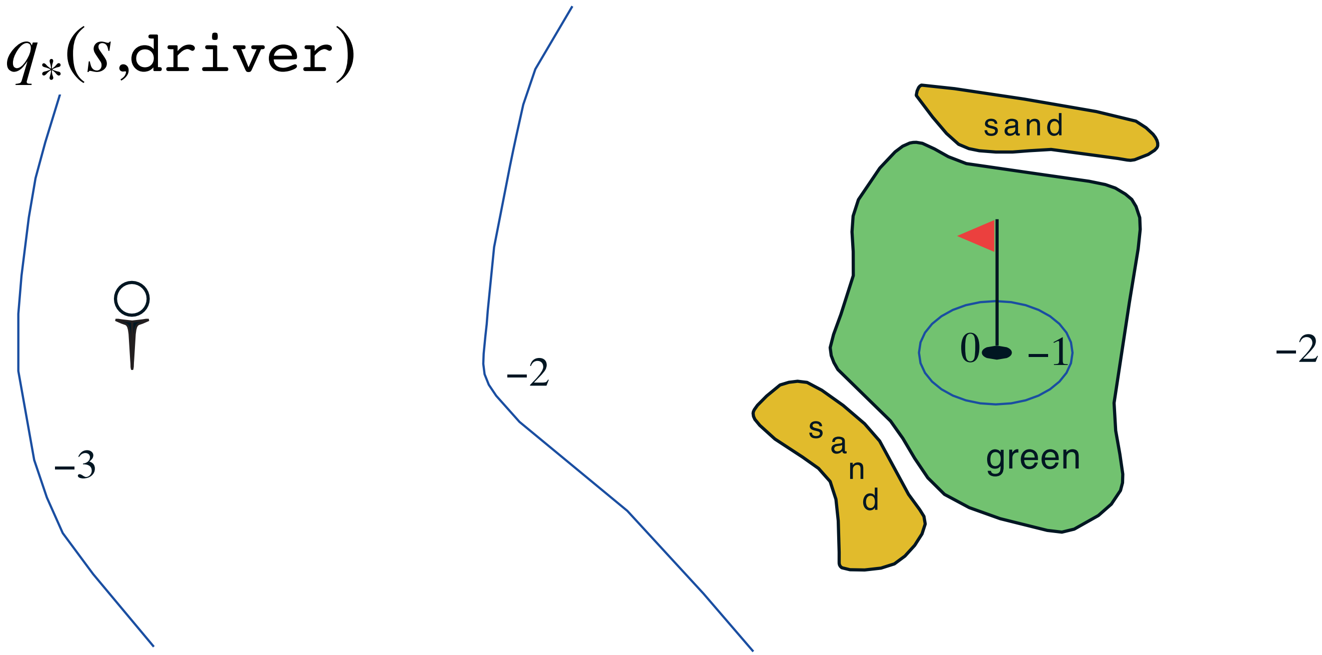
b) $v_*$



c) $\pi_*$

# Optimal Value Function for Golf

- We can hit the ball farther with `driver` than with `putter`, but with less accuracy
- $q_*(s,\texttt{driver})$ gives the value or using `driver` first, then using whichever actions are best



# What About Optimal Action-Value Functions?

Given $q_*$, the agent does not even have to do a one-step-ahead search:

$$\boxed{\pi_*(s) = \arg\max_a q_*(s,a)}$$

# Bellman Equation for a Policy π

The basic idea:

$$
\begin{aligned}
G_t &= R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \cdots \\
&= R_{t+1} + \gamma \left( R_{t+2} + \gamma R_{t+3} + \gamma^2 R_{t+4} + \cdots \right) \\
&= R_{t+1} + \gamma G_{t+1}
\end{aligned}
$$

So:

$$
\boxed{
\begin{aligned}
v_\pi(s) &= E_\pi \left\{ G_t \,\middle|\, S_t = s \right\} \\
&= E_\pi \left\{ R_{t+1} + \gamma v_\pi\left(S_{t+1}\right) \,\middle|\, S_t = s \right\}
\end{aligned}
}
$$

Or, without the expectation operator:

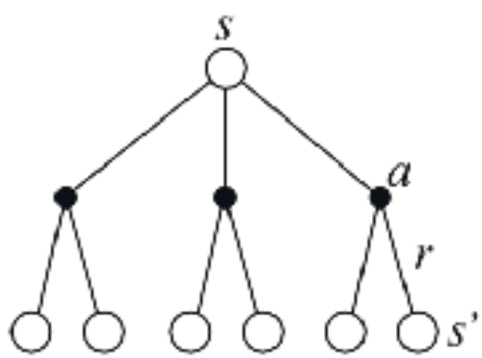
$$
v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)\Big[r + \gamma v_\pi(s')\Big]
$$

# More on the Bellman Equation

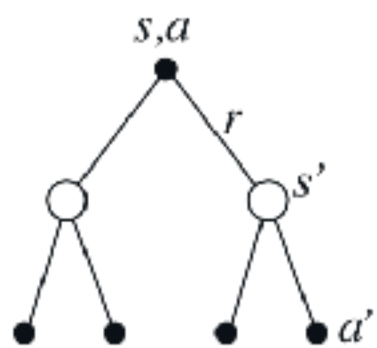
$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)\Big[r + \gamma v_\pi(s')\Big]$$

This is a set of equations (in fact, linear), one for each state.
The value function for $\pi$ is its unique solution.

**Backup diagrams**:




for $v_\pi$



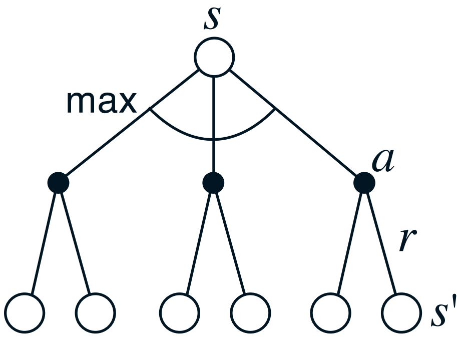

for $q_\pi$

# Bellman Optimality Equation for $v_*$

The value of a state under an optimal policy must equal the expected return for the best action from that state:

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

The relevant backup diagram:

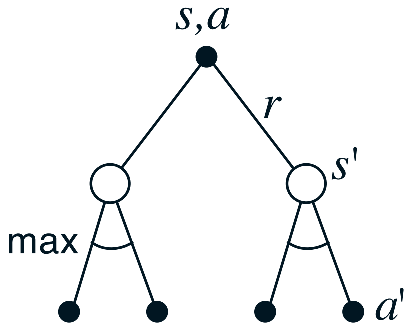



$v_*$ is the unique solution of this system of nonlinear equations.

# Bellman Optimality Equation for $q_*$

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

The relevant backup diagram:




$q_*$ is the unique solution of this system of nonlinear equations.

# Solving the Bellman Optimality Equation

- Finding an optimal policy by solving the Bellman Optimality Equation requires the following:
  - accurate knowledge of environment dynamics;
  - we have enough space and time to do the computation;
  - the Markov Property.
- How much space and time do we need?
  - polynomial in number of states (via dynamic programming methods; Chapter 4),
  - BUT, number of states is often huge (e.g., backgammon has about $10^{20}$ states).
- We usually have to settle for approximations.
- Many RL methods can be understood as approximately solving the Bellman Optimality Equation.

# Summary

- Agent-environment interaction
  - States
  - Actions
  - Rewards
- Policy: stochastic rule for selecting actions
- Return: the function of future rewards agent tries to maximize
- Episodic and continuing tasks
- Markov Property
- Markov Decision Process
  - Transition probabilities
  - Expected rewards
- Value functions
  - State-value function for a policy
  - Action-value function for a policy
  - Optimal state-value function
  - Optimal action-value function
- Optimal value functions
- Optimal policies
- Bellman Equations
- The need for approximation