Sequential Decision Problems

Decision Theory

- The science of making decisions to maximize returns.
- A probabilistic view:
  - We have some set of possible actions A.
  - We have a set of possible results S.
  - Assume we know $\boldsymbol{P}(S \mid A)$ – the distribution of results given actions.
  - We assign different value to different states.
    - Expressed with a utility function: $U(S)$.

Blackjack Example

- *A* could be *hit* or *stand* in blackjack.
- *S* could be *blackjack*, *bust*, or some higher point total.
  - U(blackjack) = the pot
  - U(bust) = 0
  - U(higher points) = somewhere in between
- How do we decide which action to take?
  - Maximize probability of getting the highest possible return?
  - Minimize the change of getting the lowest possible utility?
  - Maximize expected utility – amount we will win on average?

Expected Utility

- The amount that we expect to receive for a given action:

$$EU(a) = \sum_{s \in S} U(s) P(s|a)$$

- Maximizing expected utility:

$$\underset{a \in A}{argmax} \sum_{s \in S} U(s) P(s|a)$$

Sequential Decisions

- Previous discussion only pertains to making a single decision.
- More generally, we might need to make a series of decisions that lead us from one state to the next:
  - $s_0, s_1, \ldots, s_N$

Markov Decision Problems

- Specified by two functions,
  - Transition model: $P(s' \mid s, a)$ expresses the probability that the system will end up in state s' if action a is taken in state s.
  - Reward function: $R(s)$ expresses the immediate reward associated with each state.
- Our goal is to find $\pi^*(s)$, a mapping from states to actions that results in the highest utility.
- How do we define utility for a sequence of states?
  - $U([s_0, s_1, \ldots, s_N])$

Utility of a State Sequence

- One possibility, sum of rewards:
  - $U([s_0, s_1, ..., s_N]) = R(s_0) + R(s_1) + ...$
  - Doesn't make sense for infinitely long sequences.
- A second possibility, discounted reward:

$$U([s_0, s_1, \ldots]) = R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \ldots$$

- γ is a discount factor that ranges from 0 to 1.
- It has the nice property that (if $\gamma < 1$) the sum will be finite.

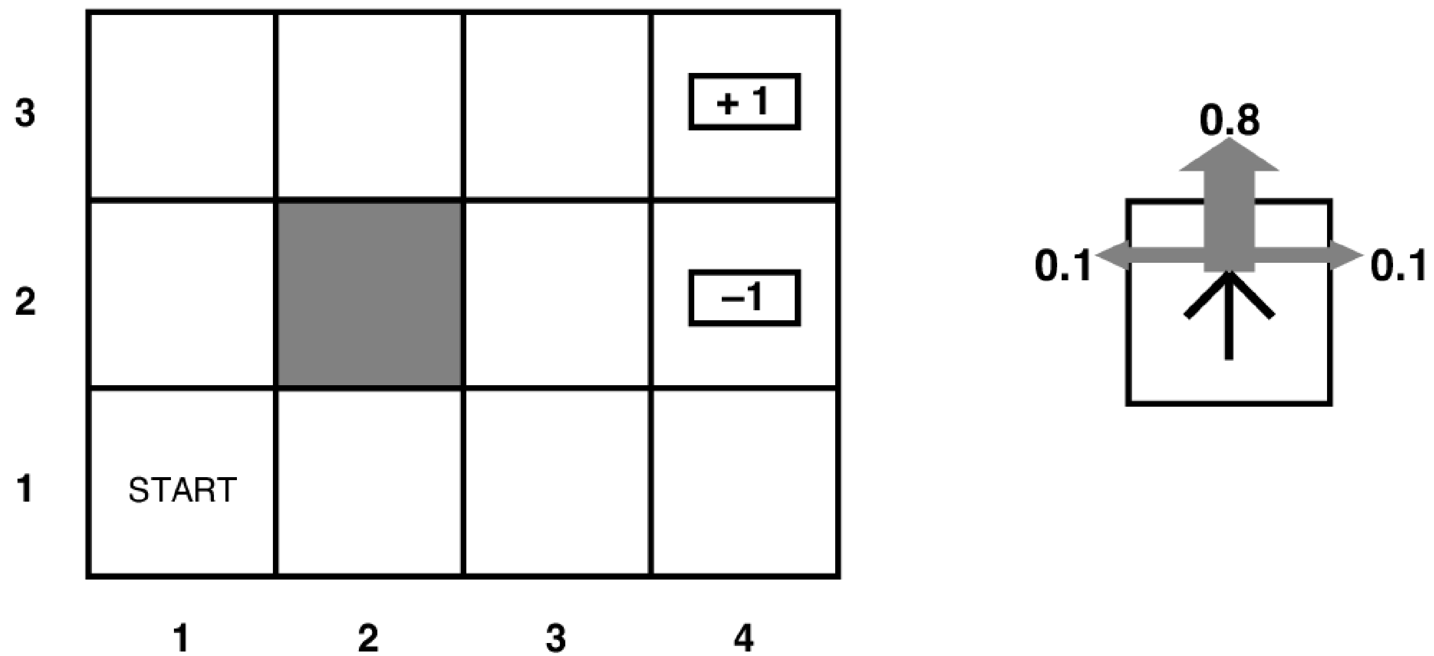
Optimal Policies

- Now we can specify what we mean by an “optimal policy”.

$$\pi^* = \underset{\pi}{argmax} = E\left[\sum_{t=0}^{\infty} \gamma^t R(s_t) \mid \pi\right]$$

- In other words, we want the policy with the highest expected sum of discounted reward.

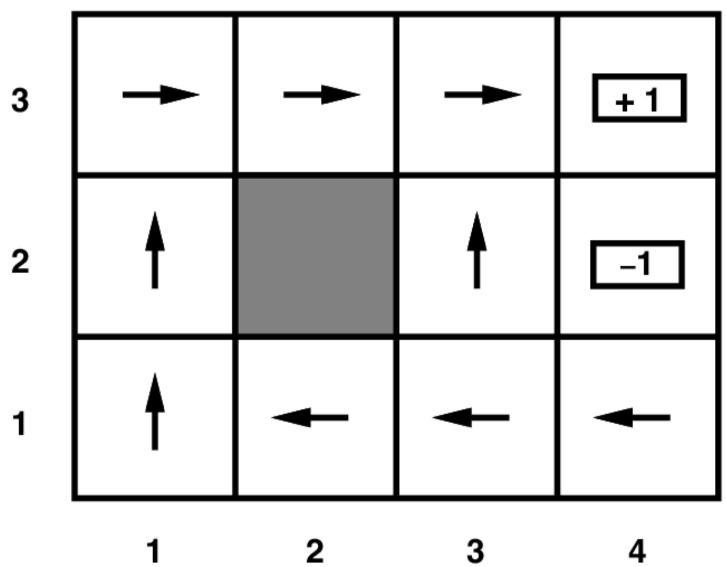
Simple MDP



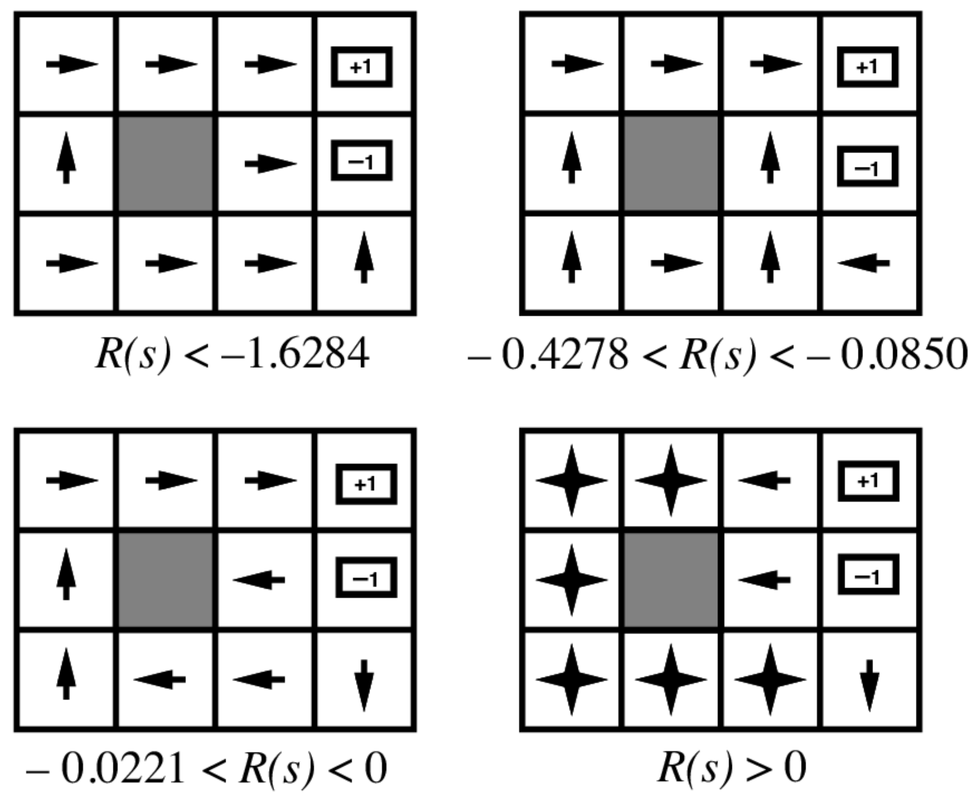
(a) (b)

R(s) = -.04 for all non-terminal states.

Optimal Policies



(a)



(b)

State Utility

- First we define the utility of a state with respect to a policy:

$$U^{\pi}(s) = E\left[\sum_{t=0}^{\infty} \gamma^t R(s_t) \mid \pi, s_0 = s\right]$$

- The utility of *s* is equal to the expected discounted reward we will receive if we start in *s*.
- What we *really* want is:

$$U(s) = U^{\pi^*}(s) = \max_{\pi} E\left[\sum_{t=0}^{\infty} \gamma^t R(s_t) \mid \pi, s_0 = s\right]$$

State Utilities

	1	2	3	4
3	0.812	0.868	0.918	+ 1
2	0.762		0.660	–1
1	0.705	0.655	0.611	0.388

R(s) = -.04 for all non-terminal states.

$\gamma = 1$

Optimal Policy

- If we know *U(s)*, we can get π^* , specifically:

$$\pi^*(s) = \underset{a}{argmax} \sum_{s'} P(s'|a,s) U^*(s')$$

- All we need now is *U(s)*.

The Bellman Equation

- We can write the utility of a given state as follows:

$$U(s) = R(s) + \gamma \max_a \sum_{s'} P(s'|a, s') U(s')$$

  - The value of a state is equal to the immediate reward plus the expected discounted utility of the next state, assuming we choose the best action.
- If there are *N* states, we have *N* instances of the equation above.
- *N* equations in *N* unknowns!
- Unfortunately, they are non-linear equations.

Value Iteration Algorithm

- We can find a solution iteratively.
- First, guess *U(s)* for all *s*.
- Then repeat the following until satisfied:
  - for each state s:

$$U_{i+1}(s) \leftarrow R(s) + \gamma \max_{a} \sum_{s'} P(s'|a, s') U_i(s')$$

  - where *i* is the iteration number.
- This is guaranteed to converge to the true *U(s)*.
- Converges more quickly for small γ.