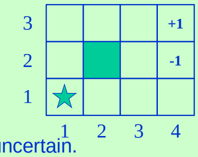
Markov Decision Process and Reinforcement Learning

- Objectives
  - ❑ Markov Decision Process (MDP)
  - ❑ Utility of State
  - ❑ Value Iteration
  - ❑ Passive Reinforcement Learning
  - ❑ Active Reinforcement Learning
- Reference
  - ❑ Russell & Norvig: Chapter 17 & 21

Sequential Decision Making

- Ex Grid world
  - ❑ Start and terminal states
  - ❑ Actions and rewards
  - ❑ Fully observable
  - ❑ Stochastic: Effects of actions are uncertain.



- If agent performs (n,n,e,e,e), what is the probability of reaching (4,3)?
- A broad class of problems have a similar nature.
  - ❑ Robot navigation, project management, planning a complex operation, …

Markov Decision Process (MDP)

- An MDP consists of
  - ❑ a set S of states including an initial state s_0,
  - ❑ a transition model T(s, a, s'), and
  - ❑ a reward function R(s).
- What should a solution to an MDP look like?
- A policy π specifies the action $\pi(s)$ for each state s.
- An optimal policy π^* is a policy that yields the highest expected utility.

Utility Function

- Which policy is optimal depends on utility function.
- Denote the utility over history $[s_0, s_1, \ldots, s_n]$ as $U_h([s_0, s_1, \ldots, s_n])$.
  - ❑ Many alternatives for $U_h([s_0, s_1, \ldots, s_n])$ exist.
- Additive rewards
  - ❑ The utility of state sequence $[s_0, s_1, \ldots, s_n]$ is
    $U_h([s_0, s_1, \ldots, s_n]) = \Sigma_i R(s_i)$.

Optimal Policies for Grid World

- Optimal policy depends on chosen utility function which in turn depends on reward function R(s).
- Ex Possible range of R(s) for s ≠ (4,2) or (4,3) and corresponding optimal policy
  a) R(s) = -0.04
  b) R(s) < -1.6284
  c) R(s) ∈ (-0.0221, 0)
  d) R(s) > 0

	1	2	3	4
3	→	→	→	+1
2	↑		↑	-1
1	↑	←	←	←

Horizon

- Horizon specifies the maximum length of history that matters.
  - ❑Finite horizon N
    $U_h([s_0, s_1, \ldots, s_{N+k}]) = U_h([s_0, s_1, \ldots, s_N])$ for k>0.
  - ❑Infinite horizon
- Optimal policy for a finite horizon is non-stationary.
  - ❑Ex At (3,1) of grid world
- For infinite horizon, the optimal policy is stationary.

Utility as Discounted Rewards

- With infinite horizon and utility as additive rewards, utility of any infinite state sequence is infinite.
  - ❑How can alternative sequences be compared?
- Discounted rewards
  - ❑The utility of state sequence $[s_0, s_1, \ldots]$ is
    $U_h([s_0, s_1, \ldots]) = \Sigma_i \gamma^i R(s_i)$, where $\gamma \in (0,1]$.
  - ❑If rewards are bounded and $\gamma < 1$, then utility of an infinite sequence $[s_0, s_1, \ldots]$ is finite.
  - ❑Additive rewards is a special case.
- We assume infinite horizon and discounted rewards below.

Utility of State

- Idea to compute optimal policy
  - ❑Compute the utility of each state and use state utilities to select optimal action for each state.
- Utility of a state s wrt policy π is
  $U^\pi(s) = E(\Sigma_i \gamma^i R(s_i) \mid \pi, s_0 = s)$.
- Utility U(s) of a state s is its utility wrt an optimal policy π^*, i.e., $U(s) = U^{\pi^*}(s)$.
- Ex State utilities with R(s) = -0.04 and γ=1

Ex State Utilities of Grid World

- $R(s) = -0.04$ and $\gamma=1$

	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

State Utility and Optimal Policy

- From MEU principle, optimal action $\pi^*(s)$ at state s satisfies the following:

  $\forall a \;\; \Sigma_{s'} T(s,\pi^*(s),s')U(s') \geq \Sigma_{s'} T(s,a,s')U(s')$
- Ex What is the optimal action at (1,1)?
- Reflection: If state utility is known for each state, then the optimal policy can be obtained.
  - How can state utilities be obtained?
  - Can they be obtained based on state utility definition?

Bellman Equation

- Bellman equation: Utility of a state is given as

  $U(s) = R(s) + \gamma \max_a \Sigma_{s'} T(s, a, s')U(s')$.
  - Ex Utility for state (1,1)
  - The equation suggests an iterative approach to compute state utility.
- Bellman update
  - Given transition model $T(s, a, s')$ and reward function $R(s)$, obtain state utilities by

    $U_{i+1}(s) = R(s) + \gamma \max_a \Sigma_{s'} T(s,a,s')U_i(s')$,

  where $i=0,1,\ldots$ and $U_0(s)=0$.

Value Iteration Algorithm

```
valueIteration(ε) {
  U'[]: vector of utilities for states, initially zero;
  δ: max utility change of any state in one round;
  repeat
     U = U'; δ = 0;
     for each state s, do
        U'[s] = R[s] + γ max_a Σ_s' T(s,a,s')U[s'];
        if |U'[s] - U[s]| > δ, then δ = |U'[s] - U[s]|;
  until δ < ε;
  return U;
}
```

Reinforcement Learning (RL)

- What has value iteration achieved?
- Reality in many agent environments
  - What agent knows
    - Set of states, start and terminal states
  - What agent does not know
    - Transition model T(s, a, s')
    - Reward function R(s)
  - What agent can perceive
    - Current state
    - Reward received at the current state
- Objective of RL: Learn an optimal policy for the env from observed rewards.

Passive Reinforcement Learning

- Task: Learn utility of each state s wrt a fixed policy π, i.e., $U^{\pi}(s) = E(\Sigma_i \gamma^i R(s_i) \mid \pi, s_0 = s)$.
- In passive RL, agent performs a set of trials.
- In each trial, agent starts from s, executes policy π, experiences a sequence of state transitions, receives reward at each state, until reaching a terminal state.
- Ex A typical trial

$(1,1)_{-.04} \rightarrow (1,2)_{-.04} \rightarrow (1,3)_{-.04} \rightarrow (1,2)_{-.04} \rightarrow (1,3)_{-.04} \rightarrow (2,3)_{-.04} \rightarrow (3,3)_{-.04} \rightarrow (4,3)_{+1}$

Direct State Utility Estimation

- $U^{\pi}(s)$ is expected total reward from state s onward.
- Each trial provides a sample of expected total reward for each state visited.
- Ex The grid world trial
- Method to estimate $U^{\pi}(s)$
  - After each trial, update average total reward for each state visited.
  - As number of trials approaches infinity, average total reward for s converges to $U^{\pi}(s)$.

Motivation of Adaptive Dynamic Programming (ADP)

- Limitation of direct state utility estimation
  - It treats states as if they are independent of each other.
  - It often converges very slowly.
- Idea for improvement
  - Handle state dependency with Bellman equation.
  1) But Bellman equation is for π^*, not any π!
  2) Where do R(s) and T(s,a,s') come from?

Utility of State Revisited

- Utility of a state s wrt policy π is
  $U^{\pi}(s) = E(\Sigma_i \gamma^i R(s_i) \mid \pi, s_0 = s)$.
- Bellman equation: Utility of a state is
  $U(s) = R(s) + \gamma \max_a \Sigma_{s'} T(s, a,s')U(s')$.
- Simplified Bellman equation:
  $U^{\pi}(s) = R(s) + \gamma \Sigma_{s'} T(s, \pi(s),s')U^{\pi}(s')$.
  - Utility of a state wrt policy π equals its own reward plus expected utility of its successor states.

Simplified Value Iteration

- Source of R(s)
  - For each state s experienced, its R(s) is observed.
- Where does $T(s, \pi(s),s')$ come from?
  - Estimate T(s,a,s') from the frequency with which s' is reached when a is executed at s.
  - Ex Two trials in grid world
- Simplified value iteration
  $U^{\pi}_{i+1}(s) = R(s) + \gamma \Sigma_{s'} T(s, \pi(s),s')U^{\pi}_{i}(s')$.
  - Repeat for K times, where K is a constant.

ADP Agent

- At each step, agent
  - performs an action according to policy π,
  - perceives new state s',
  - Receives reward r',
  - updates its transition model T(), and
  - updates state utilities by simplified value iteration.

ADP Agent Initialization

- Set S of states, fixed policy π, and discount factor γ
- Set U of state utilities, initialized to 0
- Set R of state rewards, initialized to 0
- Nsa: repetition counters for state-action pairs
  - For each pair, initialize counter to 0.
- Nsas': repetition counters for s-a-s' triples
  - For each triple, initialize counter to 0.
- Transition model T: Init each transition prob to 0.
- Previous state s and action a: initialized to null

```
passiveADP(s', r') {
  static s, a, π, γ;
  if 1st visit of s', then U[s'] = r'; R[s'] = r';
  if s ≠ null, do
      Nsa[s,a]++; Nsas'[s,a,s']++;
      for each t such that Nsas'[s,a,t] ≠ 0, do
          T[s,a,t] = Nsas'[s,a,t] / Nsa[s,a];
      U = simplifiedValueIteration(S, R, T, π, γ);
  if terminal(s'), then s=null; a=null;
  else s = s'; a = π(s');
  return a;
}
```


Active Reinforcement Learning

- In general, agent knows neither which policy to use nor state utilities. How should it act?
- Explore env to learn transition model and state utilities.
- Follow the best policy derived from learned model.
- As more is known about env, the best policy will converge to the optimal policy.

Active ADP Agent

1. Optimistic value iteration: Replace simplified value iteration in passiveADP() by the optimistic estimate $U^+(s) = R(s) + \gamma \max_a f(\Sigma_{s'} T(s,a,s')U^+(s'), N(s,a))$, where

$$f(u,n) = \begin{cases} R^+ & \text{if } n < M; \\ u & \text{otherwise.} \end{cases}$$

2. At state s, take action a* s.t. the following holds:

$$\forall a \quad f(\Sigma_{s'} T(s,a^*,s')U^+(s'), N(s,a^*)) \geq f(\Sigma_{s'} T(s,a,s')U^+(s'), N(s,a))$$

- The activeADP algorithm
- Execution and properties


```
activeADP(s', r') {
  static s, a, γ;
  if s' visited 1st time, then U[s'] = r'; R[s'] = r';
  if s ≠ null, do
      Nsa[s,a]++; Nsas'[s,a,s']++;
      for each t such that Nsas'[s,a,t] ≠ 0, do
          T[s,a,t] = Nsas'[s,a,t] / Nsa[s,a];
      U = optimisticValueIteration(S, R, T, γ, Nsa);
  if terminal(s'), then s=null; a=null;
  else s = s'; a = getBestAction(s', T, U, Nsa);
  return a;
}
```