Chap. 17

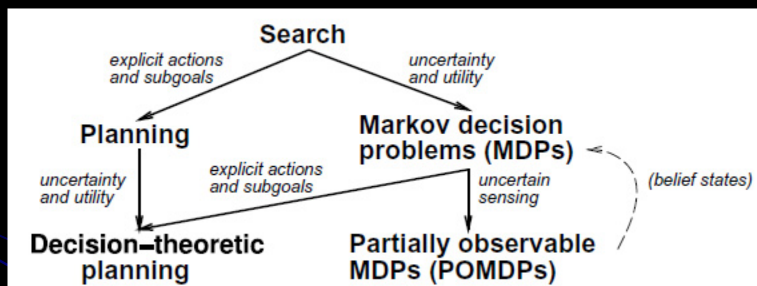
Making Complex Decisions

Decision making methods of what to do today,
given that we may decide again tomorrow
– sequential decision problem.

Outline

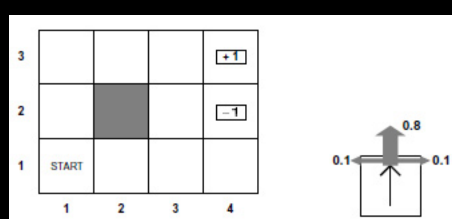
- Sequential Decision problems
  - The agent's utility depends on a sequence of decisions.
  - $\supset$ utilities, uncertainty, sensing.
  - A stochastic generalization of search/planning problem.
  - How sequential decision problem is defined?
- Value iteration
  - How to solve them to produce *optimal behavior* that balances the risks and rewards of acting in an uncertain environment
    - i.e. how to find an optimal policy.
- Policy iteration
  - An alternative way to find optimal policies.

Sequential decision problems



	Deterministic	Stochastic
Fully Observable	A*, DFS, BFS, etc.	MDP
Partially Observable		POMDP

Markov Decision Process(MDP)



- The specification of a *sequential decision problem* for a *fully observable environment* with a *Markovian Transition Model* and additive *Rewards.*
- Components: a set of states S (initial state=s_0), a set of ACTIONS(s), $\forall s \in S$,
  transition model M^a_{ij}, *reward function* $R(i)$
  - Model $M^a_{ij} \equiv P(j|i, a)$ = probability that doing a in i leads to j : Markov transition
  - Utility function will depend on a *sequence of states* (i.e. an environment history) rather than a single state.
  - Each state has a *reward* $R(i)$
    = -0.04 (small penalty) for nonterminal states; = ±1 for terminal states
  - Utility of an environment history ≈ the *sum* of the rewards received.

Solving MDPs

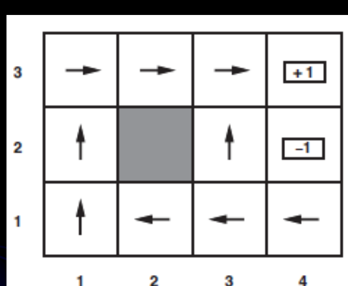
- In search problems, aim is to find an optimal *sequence of actions.*
- In MDPs, aim is to find an optimal *policy* π

  i.e. best action for *every possible state*

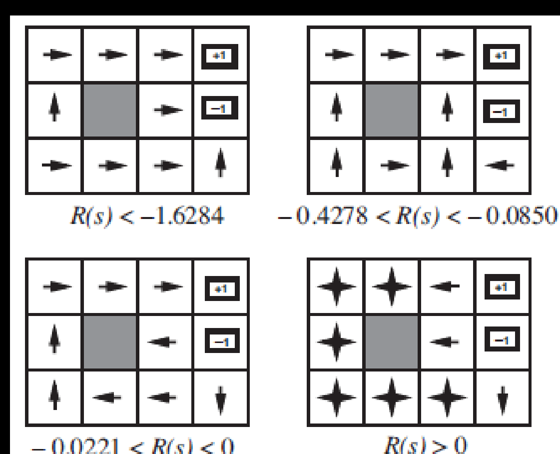
  (because can't predict where one will end up)

  $\pi(s)$: the action recommended by the policy π for state s.
- The quality (i.e. value) of a policy is measured by the *expected utility* of the possible environment histories generated by that policy.
- An *optimal policy* π^* : a policy with the *highest expected utility (MEU).*
- Given π^*, the agent decides what to do by consulting its current percept, which tells it the current state s, and then executing the action $\pi^*(s)$.
- The balance of risk and reward changes depends on the value of $R(s)$ for the nonterminal states.
- The careful *balancing of risk & reward* is a characteristic of MDPs which doesn't arise in deterministic search problem but in real-world decision problem.

5



An optimal policy for the stochastic environment with R(s) = -.04 in the nonterminal states



Optimal policies for four different ranges of R(s).

6

Utility

- The performance of agent is measured by a sum or rewards for the states visited.
- The choices for the utility function on environment histories.
- Is there a finite horizon or an infinite horizon for decision making?
  - $U_h([s_0, s_1, .., s_{N+k}]) = U_h([s_0, s_1, .., s_N)]$, for all $k > 0$.
  - With a finite horizon (with a fixed time N), the optimal action in a given state could change over time.

    --the optimal policy for a finite horizon is non-stationary.
  - With a infinite horizon, the optimal action depends only on the current state and is stationary.
  - Policies for the infinite horizon (i.e. no fixed deadline) case are simpler.
- How to calculate the utility of state sequences?
  - A question in multiattribute utility theory where s_i is viewed as attribute of the state sequence $[s_0, s_1, ..,]$.
  - Assumption: the agent's preferences b/t state sequences are stationary:

7

Utility: cont.

- In *sequential* decision problems, preferences are expressed between *sequences* of states.
- *Preference-independence* Assumption: agent's preferences between state sequences are *stationary*.
  - If $s_0=s_0'$ in two sequence $[s_0, s_1, s_2, \ldots]$ and $[s_0', s_1', s_2', \ldots]$,

    then two sequences should be preference-ordered the same way

    as $[s_1, s_2, \ldots]$ and $[s_1', s_2', \ldots]$,
- *Additive Rewards :* Usually uses an *additive* utility function

  $$U_h([s_0, s_1, \ldots , s_n]) = R(s_0)+R(s_1)+R(s_2)+ \ldots +R(s_n)$$

  (cf. path cost in search problems).
- *Discounted Rewards*:

  $$U_h([s_0, s_1, \ldots , s_n]) = R(s_0)+\gamma R(s_1)+\gamma^2 R(s_2)+ \ldots +\gamma^n R(s_n)$$
  $$= \sum_{t=0}^{n} \gamma^t R(s_t).$$

  where γ: discount factor $\in [0, 1]$,

  the preference of an agent for current rewards over future rewards. 8

  $\gamma \to 0$, rewards in the distant future are insignificant; $\gamma=1$, additive rewards.

Utility: cont.

- If the environment contains no terminal state or if the agent never reaches one → infinite horizon.
- Solutions for some problems with infinite horizons:
  - Utility of an infinite sequence is finite with discounted rewards:
    $U_h([s_0, s_1, \ldots, s_n]) = \sum_{t=0}^{\infty} \gamma^t R(s_t) \leq \sum_{t=0}^{\infty} \gamma^t R_{max} = R_{max} / (1 - \gamma)$.
  - If the environment contains the terminal states and if the agent is guaranteed to get to one eventually, then no need to compare infinite sequences.
    - A *Proper policy*: a policy that is guaranteed to reach a terminal state.
    - An improper policy can cause the standard algorithms for solving MDPs to fail with additive rewards – good to use discounted rewards.
  - Compare infinite sequences in terms of the average reward per time step.

9

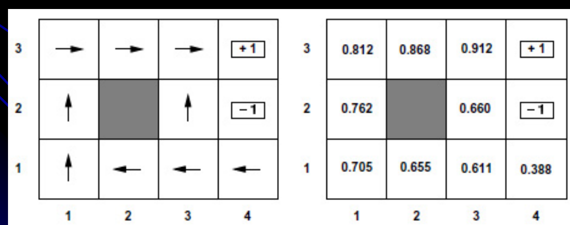
Utilities : cont.

- Compare policies, (π1, π2), by their EU when executing them: $U^{\pi 1}$, $U^{\pi 2}$.
- *Utility of a state* (a.k.a. its value) : the EU by executing a policy π in s
  $$U^{\pi}(s) = E\left[\Sigma_{t=0}^{\infty} \gamma^t R(s_t) \mid \pi, s_0 = s\right]$$
  where the expectation is w.r.t. the probability distribution over state sequence decided by s and π.
- How to choose b/t policies?
  - A given policy generates a whole range of possible state sequences
  - The *value of a policy* is the *Expected Sum* of disounted rewards obtained where the expectation is taken over all possible state sequences that could occur: Expected Utility of a policy.
  - An *optimal policy* : $\pi_s^* = \text{argmax}_{\pi}\, U^{\pi}(s) = \text{argmax}_{\pi}\, E[\, \Sigma_{t=0}^{\infty} \gamma^t R(s_t) \mid \pi \,]$
    where s is the start state, s_t is the state the agent is in after executing π for t steps.
  - The optimal policy is independent of the starting state, using the discounted utilities with infinite horizons.

10

Utility: cont.

- True Utility of a state (a.k.a. its value):
  $U_h(s)$ = expected sum of discounted rewards until termination
  under the optimal policy $= U^{\pi^*}(s)$
  cf) $R(s)$: the short-term reward for being in s
  $U(s)$: the long-term total reward from s onwards (in a sequence).
- Given the utilities of the states $U(s)$, choosing the best action is just MEU:
  choose the action such that the EU of the immediate successors is highest:
  $\pi^*(i) = \mathrm{argmax}_a \sum_j U(j)\, M^a_{ij}$ where $M^a_{ij} = P(j|i, a), a \in A(i)$
- Optimal policy and state values ($U(s)$) for the given $R(s) = -.04$:



11

Bellman equation

- How to find the optimal policy?
- *Value Iteration Algorithm*: Calculate the utility of each state and
  use the state utilities to select an optimal action in each state.
- *the utility of a state* = the immediate reward for that state
  \+ the expected discounted utility of the next state
  assuming that the agent chooses the optimal action.
- *Bellman equation* (1957):
  $U(i) = R(i) + \gamma \cdot max_a \sum_j U(j) M^a_{ij}$

$$
\begin{aligned}
U(1,1) = -0.04 &+ \max\{0.8U(1,2) + 0.1U(2,1) + 0.1U(1,1), && \text{(Up)}\\
&0.9U(1,1) + 0.1U(1,2), && \text{(Left)}\\
&0.9U(1,1) + 0.1U(2,1), && \text{(Down)}\\
&0.8U(2,1) + 0.1U(1,2) + 0.1U(1,1)\} && \text{(Right)}
\end{aligned}
$$

One equation per state = n (nonlinear) equation in n unknowns

12

Value Iteration Algorithm

- Iterative approach using n Bellman equations for n possible states.
- Idea: Start with arbitrary utility values

  Update to make them *locally consistent* with Bellman eqn.

  Everywhere locally consistent $\Rightarrow$ global optimality

  repeat until "equilibrium"

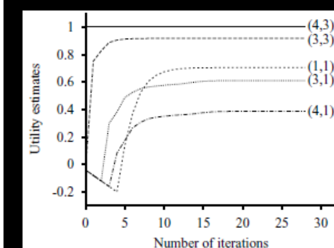
  $U_k(i) \leftarrow R(i) + \gamma \cdot \max_a \sum_j U_k(j) M^a_{ij}$ for all i, at step k.

  -- a unique solution to Bellman equation. $\Rightarrow$ optimal policy

  -- a propagation of information through the state space b.m.o. local updates.

function VALUE-ITERATION(mdp, ϵ) **returns** a utility function
inputs: mdp, an MDP with states S, actions $A(s)$, transition model $P(s' \mid s, a)$, rewards $R(s)$, discount γ
ϵ, the maximum error allowed in the utility of any state
local variables: U, U', vectors of utilities for states in S, initially zero
δ, the maximum change in the utility of any state in an iteration

repeat
$U \leftarrow U'$; $\delta \leftarrow 0$
for each state s **in** S **do**
$U'[s] \leftarrow R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s' \mid s, a)\, U[s']$
if $|U'[s] - U[s]| > \delta$ **then** $\delta \leftarrow |U'[s] - U[s]|$
until $\delta < \epsilon(1-\gamma)/\gamma$
return U



13

Policy iteration (Howard, 1960)

- Idea: search for optimal policy and utility values simultaneously
- Policy Evaluation

  Given a policy π_k, calculate $U_k = U^{\pi_k}$, the utility of each state if π_k were to be executed.

  \+ Policy Improvement

  Calculate a new MEU policy π_{k+1}, using one-step look-ahead based on U_k.
- Algorithm:

  $\pi_i \leftarrow$ an arbitrary initial policy

  repeat until no change in π

  compute utilities U_i given π_i

  update π_i as if utilities were correct (i.e. local MEU)
- To compute utilities given a fixed π

  $U(i) = R(i) + \gamma \sum_j U(j) M^{\pi(i)}_{ij}$ for all i where $M^{\pi(i)}_{ij} = P(j \mid i, \pi(i))$,

  where $\pi(i)$ specifies the action $\pi(i)$ in state i under the policy π

  $U_k(1,1) = -0.04 + 0.8U_k(1,2) + 0.1U_k(2,1) + 0.1U_k(1,1)$,

  $U_k(1,2) = -0.04 + 0.8U_k(1,3) + 0.2U_k(1,2)$,

  …..

14

Policy iteration (Howard, 1960)

function POLICY-ITERATION(mdp) **returns** a policy
  inputs: mdp, an MDP with states S, actions $A(s)$, transition model $P(s' \mid s, a)$
  local variables: U, a vector of utilities for states in S, initially zero
    π, a policy vector indexed by state, initially random

  repeat
    $U \leftarrow$ POLICY-EVALUATION(π, U, mdp)
    $unchanged? \leftarrow$ true
    for each state s **in** S **do**
      if $\max_{a \in A(s)} \sum_{s'} P(s' \mid s, a)\ U[s'] > \sum_{s'} P(s' \mid s, \pi[s])\ U[s']$ **then do**
        $\pi[s] \leftarrow \operatorname{argmax}_{a \in A(s)} \sum_{s'} P(s' \mid s, a)\ U[s']$
        $unchanged? \leftarrow$ false
  until $unchanged?$
  return π

- n simultaneous *linear* equations in n unknowns, solve in $O(n^3)$ by standard linear algebra methods.
- The simplified Bellman update: often more efficient
  $U_{k+1}(i) = R(i) + \gamma \sum_j U_k(j)\, M^{\pi(i)}_{ij}$ for all i where $M^{\pi(i)}_{ij} = P(j|i, \pi(i))$,

15