Artificial Intelligence

Topic 7

Sequential Decision Problems

- ◇ Introduction to sequential decision problems
- ◇ Value iteration
- ◇ Policy iteration
- ◇ Longevity in agents

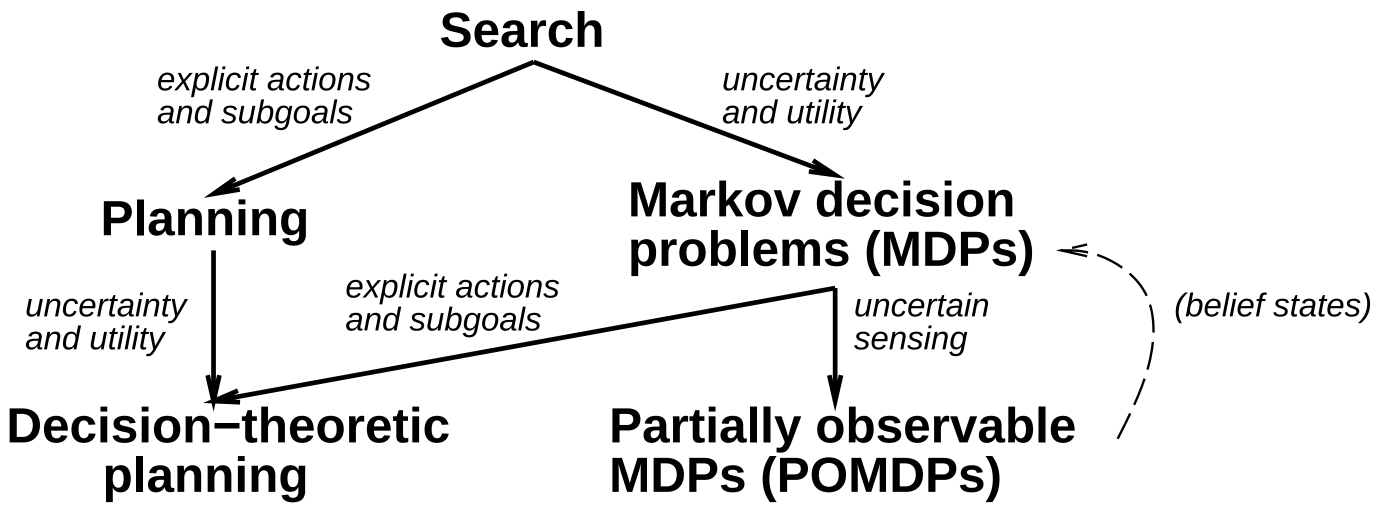
Reading: Russell and Norvig, Chapter 17, Sections 1–3

1. Sequential decision problems

Previously concerned with single decisions, where utility of each action's outcome is known.

This section — *sequential decision problems*
— utility depends on a sequence of decisions

Sequential decision problems which include utilities, uncertainty, and sensing, generalise search and planning problems. . .


Search
explicit actions and subgoals
uncertainty and utility
Planning
Markov decision problems (MDPs)
uncertainty and utility
explicit actions and subgoals
uncertain sensing
(belief states)
Decision−theoretic planning
Partially observable MDPs (POMDPs)

1.1 From search algorithms to policies

Sequential decision problems in known, accessible, deterministic domains

tools — search algorithms

outcome — sequence of actions that leads to good state

Sequential decision problems in *uncertain* domains

tools — techniques originating from control theory, operations research, and decision analysis

outcome — *policy*

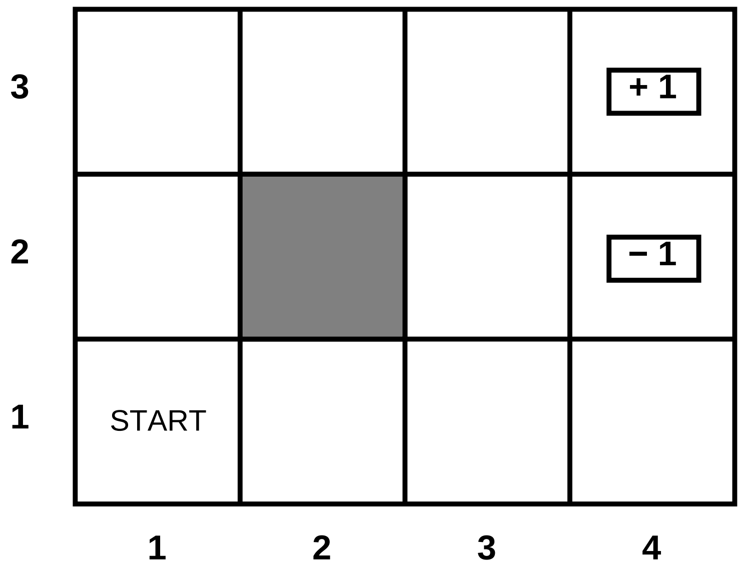
$$policy = set\ of\ state\text{-}action\ \text{“}rules\text{”}$$

- tells agent what is the best (MEU) action to try in any situation
- derived from utilities of states

This section is about finding optimal policies.

1.2 From search algorithms to policies – example

Consider the environment:



Problem

Utilities only known for terminal states

$\Rightarrow$ even for deterministic actions, depth-limited search fails!

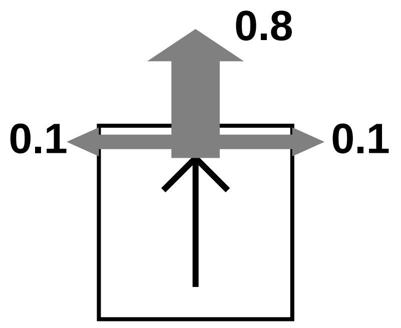
Utilities for other states will depend on sequence (or *environment history*) that leads to terminal state.

1.2 From search algorithms to policies – example

Indeterminism

deterministic version — each action (N,S,E,W) moves one square in intended direction (bumping into wall results in no change)

stochastic version
— actions are unreliable. . .



transition model — probabilities of actions leading to transitions between states

$$M_{ij}^a \equiv P(j|i,a) = \text{probability that doing } a \text{ in } i \text{ leads to } j$$

Cannot be certain which state an action leads to ($cf.$ game playing).

$\Rightarrow$ generating sequence of actions in advance then executing unlikely to succeed

1.2 From search algorithms to policies – example

Policies

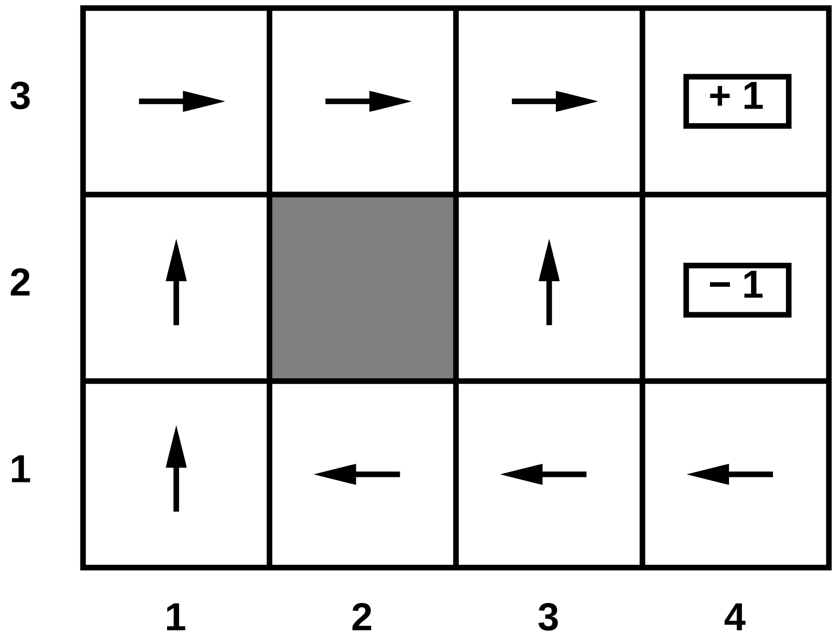
But, if

- we know what state we've reached (accessible)
- we can calculate best action for each state

⇒ *always know what to do next!*

Mapping from states to actions is called a *policy*

eg. Optimal policy for step costs of 0.04. . .



Note: small step cost ⇒ conservative policy (eg. state (3,1))

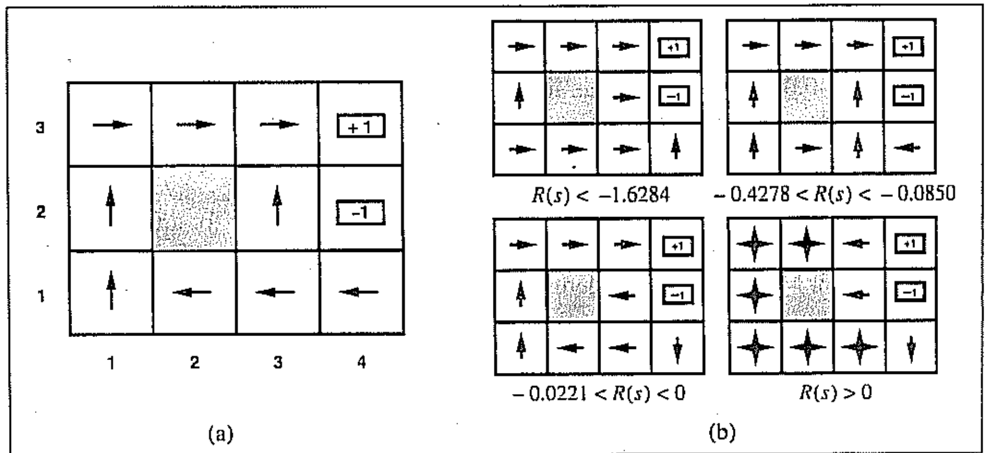


Figure 17.2 (a) An optimal policy for the stochastic environment with $R(s) = -0.04$ in the nonterminal states. (b) Optimal policies for four different ranges of $R(s)$.

heads directly away from the –1 state so that it cannot fall in by accident, even though this means banging its head against the wall quite a few times. Finally, if $R(s) > 0$, then life is positively enjoyable and the agent avoids *both* exits. As long as the actions in (4,1), (3,2), and (3,3) are as shown, every policy is optimal, and the agent obtains infinite total reward because it never enters a terminal state. Surprisingly, it turns out that there are six other optimal policies for various ranges of $R(s)$; Exercise 17.7 asks you to find them.

The careful balancing of risk and reward is a characteristic of MDPs that does not arise in deterministic search problems; moreover, it is a characteristic of many real-world decision problems. For this reason, MDPs have been studied in several fields, including AI, operations research, economics, and control theory. Dozens of algorithms have been proposed for calculating optimal policies. In sections 17.2 and 17.3 we will describe two of the most important algorithm families. First, however, we must complete our investigation of utilities and policies for sequential decision problems.

Optimality in sequential decision problems

In the MDP example in Figure 17.1, the performance of the agent was measured by a sum of rewards for the states visited. This choice of performance measure is not arbitrary, but it is not the only possibility. This section investigates the possible choices for the performance measure—that is, choices for the utility function on environment histories, which we will

NONSTATIONARY POLICY

STATIONARY POL

STATIONARY PREFERENCE

ADDITIVE REWARD

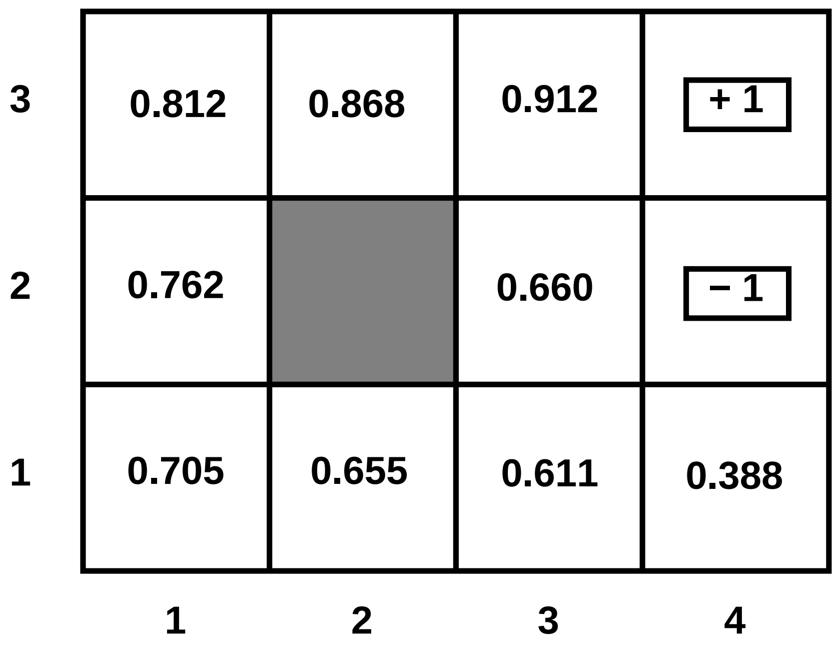
DISCOUNTED REWARDS

DISCOUNT FACTOR

1.2 From search algorithms to policies – example

Expected Utilities

Given a policy, can calculate expected utilities. . .



Aim is therefore not to find action sequence, but to find optimal policy — ie. policy that maximises expected utilities.

1.2 From search algorithms to policies – example

Policy represents agent function explicitly

utility-based agent $\mapsto$ *simple reflex agent!*

```
function SIMPLE-POLICY-AGENT(percept) returns an action
    static: M, a transition model
            U, a utility function on environment histories
            P, a policy, initially unknown

    if P is unknown then P ← the optimal policy given U, M
    return P[percept]
```

Problem of calculating an optimal policy in an accessible, stochastic environment with a known transition model is called a *Markov decision problem*.

Markov property — transition probabilities from a given state depend only on the state (not previous history)

How can we calculate optimal policies. . . ?

2. Value Iteration

Basic idea:

- calculate utility of each state $U(state)$
- use state utilities to select optimal action

Sequential problems usually use an *additive* utility function ($cf.$ path cost in search problems):

$$\begin{aligned} U([s_1, s_2, \ldots, s_n]) &= R(s_1) + R(s_2) + \cdots + R(s_n) \\ &= R(s_1) + U([s_2, \ldots, s_n]) \end{aligned}$$

where $R(i)$ is *reward* in state i (eg. +1, -1, -0.04).

Utility of a *state* (a.k.a. its *value*):

$U(s_i) =$ $\underline{\text{expected sum of rewards until termination}}$
$\underline{\text{assuming optimal actions}}$

Difficult to express mathematically. Easier is recursive form. . .

$\underline{\text{expected sum of rewards}} = \underline{\text{current reward}}$
$+ \underline{\text{expected sum of rewards after taking best action}}$

2.1 Dynamic programming

Bellman equation (1957)

$$U(i) = R(i) + \max_a \sum_j M^a_{ij} U(j)$$

eg.

$$\begin{aligned} U(1,1) = -0.04 & \\ + \max\{&0.8U(1,2) + 0.1U(2,1) + 0.1U(1,1), && \textit{up} \\ &0.9U(1,1) + 0.1U(1,2) && \textit{left} \\ &0.9U(1,1) + 0.1U(2,1) && \textit{down} \\ &0.8U(2,1) + 0.1U(1,2) + 0.1U(1,1)\} && \textit{right} \end{aligned}$$

One equation per state $= n$ <u>nonlinear</u> equations in n unknowns

Given utilities of the states, choosing best action is just maximum expected utility (MEU) — choose action such that the expected utility of the immediate successors is highest.

$$\mathit{policy}(i) = \arg\max_a \sum_j M^a_{ij} U(j)$$

Proven optimal (Bellman & Dreyfus, 1962).

How can we solve

$$U(i) = R(i) + \max_a \sum_j M^a_{ij} U(j)$$

2.2 Value iteration algorithm

Idea

- start with arbitrary utility values
- update to make them <u>locally consistent</u> with Bellman eqn.
- repeat until "no change"

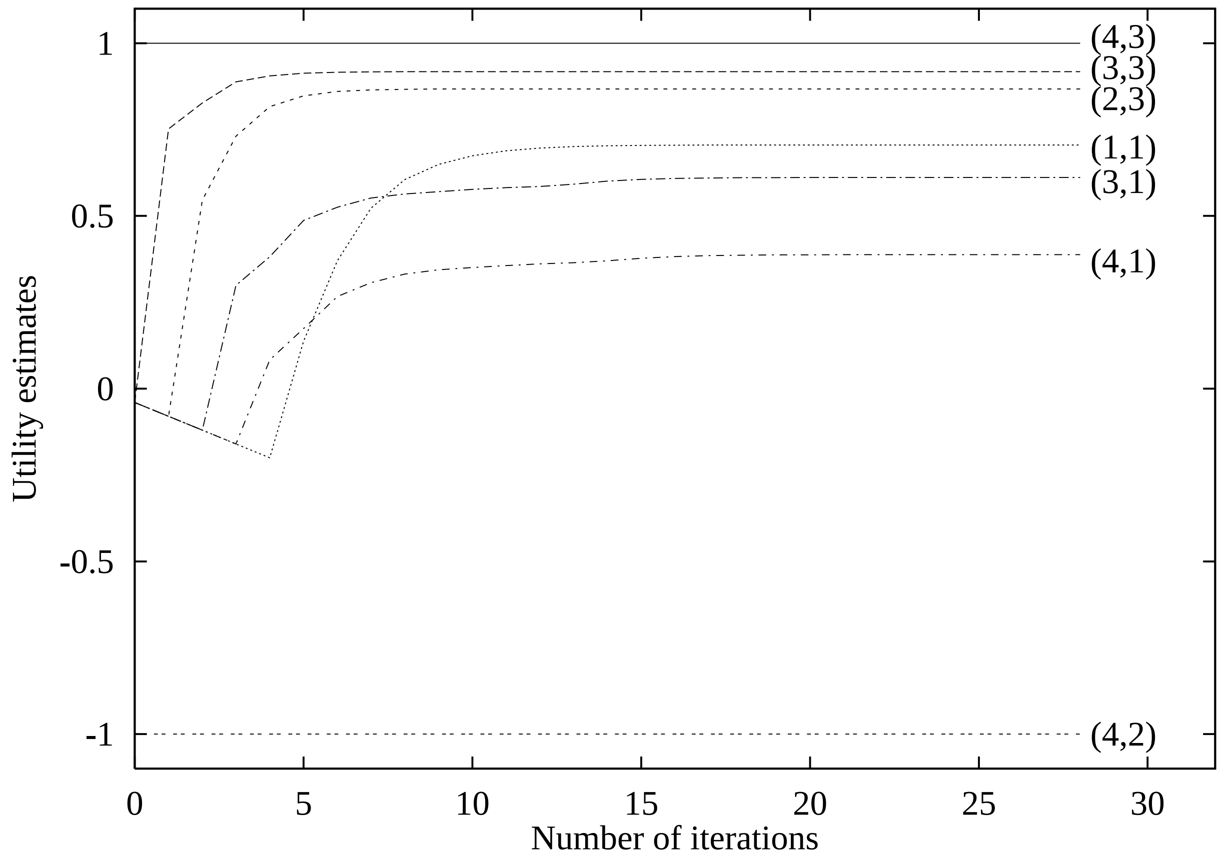
Everywhere locally consistent $\Rightarrow$ global optimality

```
function VALUE-ITERATION($M, R$) returns a utility function
    inputs: $M$, a transition model
            $R$, a reward function on states
    local variables: $U$, utility function, initially identical to $R$
                     $U'$, utility function, initially identical to $R$

    repeat
        $U \leftarrow U'$
        for each state $i$ do
            $U'[i] \leftarrow R[i] + \max_a \Sigma_j M^a_{ij} U[j]$
        end
    until CLOSE-ENOUGH($U, U'$)
    return $U$
```

Applying to our example$\cdots \rightsquigarrow$

2.2 Value iteration algorithm



3	**0.812**	**0.868**	**0.912**	$\boxed{+1}$
2	**0.762**	▨	**0.660**	$\boxed{-1}$
1	**0.705**	**0.655**	**0.611**	**0.388**
	1	**2**	**3**	**4**

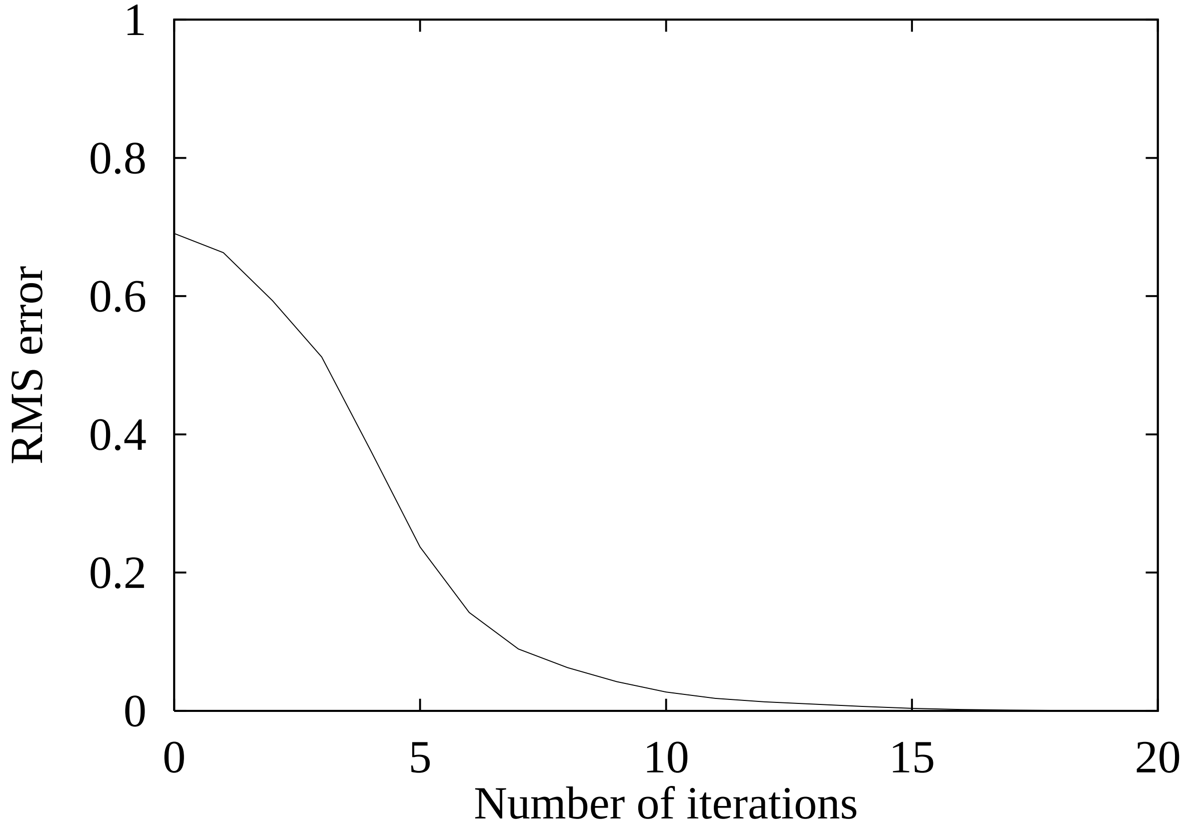
2.3 Assessing performance

Under certain conditions utility values are guaranteed to converge.

Do we require convergence?

Two measures of progress:

1. RMS (root mean square) Error of Utitily Values

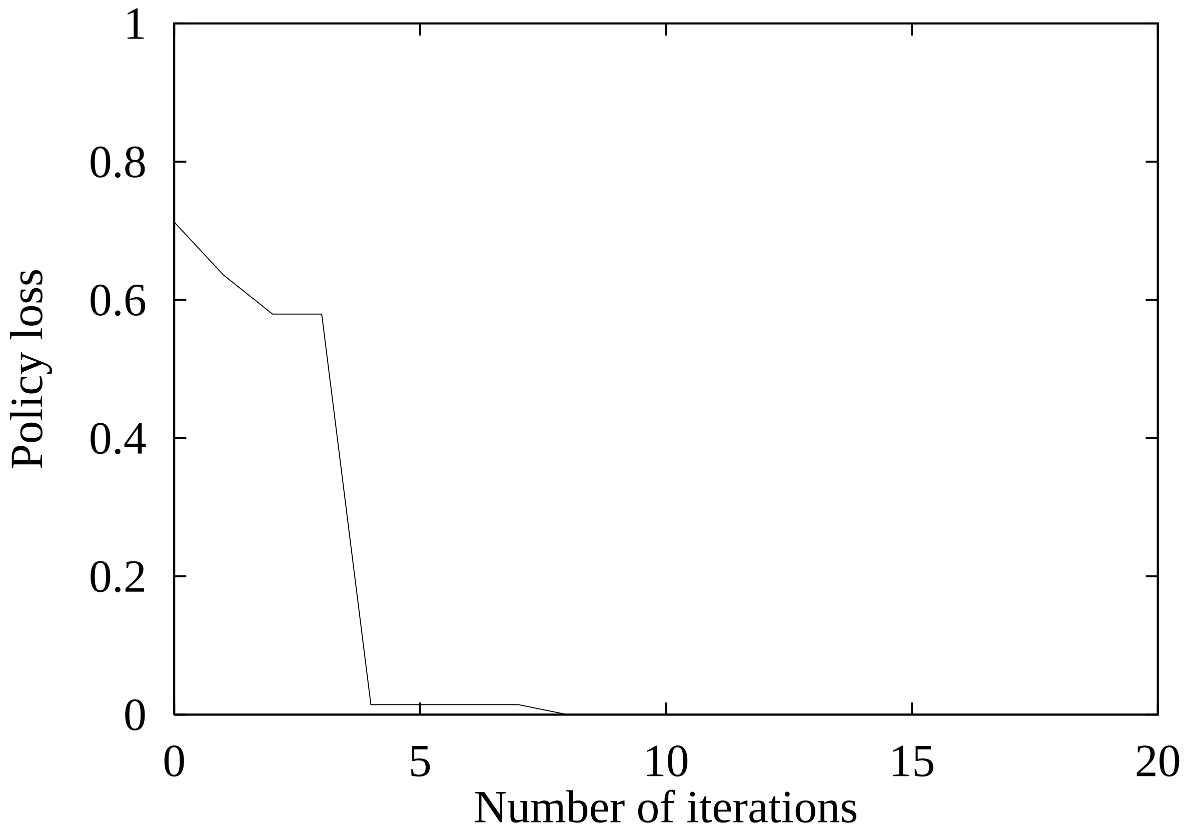


2.3 Assessing performance

2. Policy Loss

Actual utility values less important than the policy they imply

$\Rightarrow$ measure difference between expected utility obtained from policy and expected utility from optimal policy



Note: policy is optimal before RMS error converges.

3. Policy iteration

- policies may not be highly sensitive to *exact* utility values
  $\Rightarrow$ may be less work to iterate through policies than utilities!

Policy Iteration Algorithm

```
$\pi \leftarrow$ an arbitrary initial policy
repeat until no change in $\pi$
    compute utilities given $\pi$ (*value determination*)
    update $\pi$ as if utilities were correct (i.e., local MEU)
```

```
**function** POLICY-ITERATION($M, R$) **returns** a policy
    **inputs**: $M$, a transition model
                $R$, a reward function on states
    **local variables**: $U$, a utility function, initially identical to $R$
                         $P$, a policy, initially optimal with respect to $U$
    **repeat**
        $U \leftarrow$ VALUE-DETERMINATION($P, U, M, R$)
        *unchanged?* $\leftarrow$ true
        **for each** state $i$ **do**
            **if** $\max_a \sum_j M^a_{ij} U[j] > \sum_j M^{P[i]}_{ij} U[j]$ **then**
                $P[i] \leftarrow \arg\max_a \sum_j M^a_{ij} U[j]$
                *unchanged?* $\leftarrow$ false
    **until** *unchanged?*
    **return** $P$
```

3.1 Value determination

$\Rightarrow$ simpler than value iteration since action is fixed

Two possibilities:

1. Simplification of value iteration algorithm.

$$U'(i) \leftarrow R(i) + \sum_j M_{ij}^{\pi(i)} U(j)$$

May take a long time to converge.

2. Direct solution.

$$U(i) = R(i) + \sum_j M_{ij}^{\pi(i)} U(j) \qquad \text{for all } i$$

i.e., n simultaneous *linear* equations in n unknowns, solve in $O(n^3)$ (eg. Gaussian elimination)

Can be most efficient method for small state spaces.

4. What if I live forever?

Agent continues to exist — using the additive definition of utilities

- $U(i)$s are infinite!
- value iteration fails to terminate

How should we compare two infinite lifetimes?

How can we decide what to do?

One method: *discounting*

Future rewards are discounted at rate $\gamma \leq 1$

$$U([s_0, \ldots s_\infty]) = \Sigma_{t=0}^{\infty} \gamma^t R(s_t)$$

Intuitive justification:

1. purely pragmatic
   - smoothed version of limited horizons in game playing
   - smaller γ, shorter horizon
2. model of animal and human preference behaviour
   - *a bird in the hand is worth two in the bush!*
   - eg. widely used in economics to value investments

The End