Artificial Intelligence:
Representation and Problem
Solving

15-381

May 1, 2007

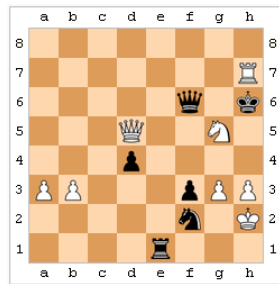
Reinforcement Learning 1

Reinforcement learning

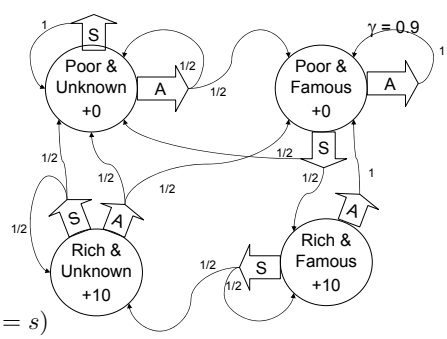
- It would be much easier if robots and agents could just teach themselves.
- only feasible method of learning in complex domains
- fundamental requirement for action exploration
- the exploration / exploitation dilemma
- how do we learn without a teacher providing detailed feedback?
- broad class of optimal control methods based on estimating value functions from experience, simulation, or search
- task of reinforcement learning is to use observed rewards to learn an optimal policy for the environment

Game playing

- What are you doing when you're learning a game?
- Supervised learning:
  - teacher (ie desired output) giving detailed feedback of the best move in each situation
  - infeasible for complex games, e.g. which chess move is best from a given board position?
- Reinforcement learning:
  - no teacher; only feedback is whether you won or lost
  - computer could simulate many games to explore space



Recall Markov Decision Processes



- An MDP is defined by:

states: $\{s_1, \ldots, s_N\}$

initial state: S_0

actions: $\{a_1, \ldots, a_M\}$

rewards: $R(s) = \{r_1, \ldots, r_N\}$

transitions: $T(s, a, s') = P(s_{t+1} = s' | a_t = a, s_t = s)$

discount: γ

- Value iteration:

$$V_{i+1}(s) = R(s) + \gamma \max_a \sum_{s'} T(s, a, s') V_i(s')$$

$$V^*(s) = \lim_{i \to \infty} V_i(s)$$

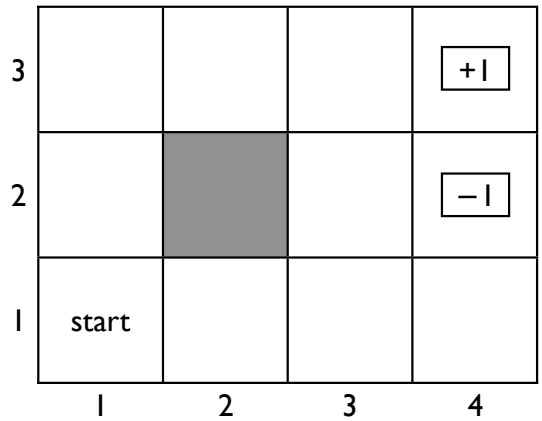
- Optimal policy:

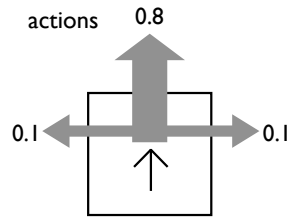
$$\pi^*(s) = \arg\max_a \left[R(s) + \gamma \sum_{s'} T(s, a, s') V^*(s') \right]$$

i	V_i(PU)	V_i(PF)	V_i(RU)	V_i(RF)
1	0	0	10	10
2	0	4.5	14.5	19
3	2.03	6.53	25.08	18.55
4	3.852	12.20	29.63	19.26
5	7.22	15.07	32.00	20.40
6	10.03	17.65	33.58	22.43

The 4x3 grid world from R&N Chap. 17

R(s) = -0.04



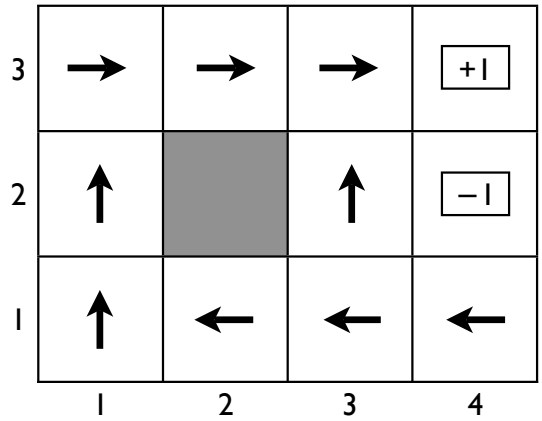


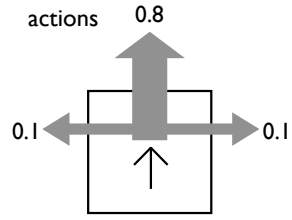
$$T(s,a,s') = P(s_{t+1}|a_t, s_t)$$

- Here, the agent has a complete model of the fully-observable world
- The reward function R(s) and the transition model T(s,a,s') are also known.
- In this example, there is no discounting
- How does the agent maximize reward? Value (or policy) iteration.

The optimal policy

R(s) = -0.04



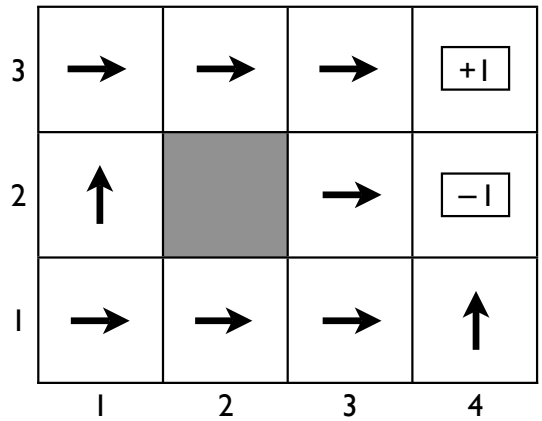


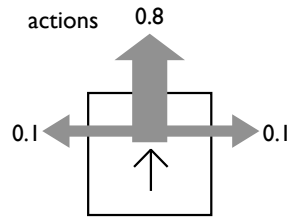
$$T(s,a,s') = P(s_{t+1}|a_t, s_t)$$

- With weak negative reward, best policy is to avoid -1 when possible and seek +1

Another optimal policy

R(s) = < 1.6285



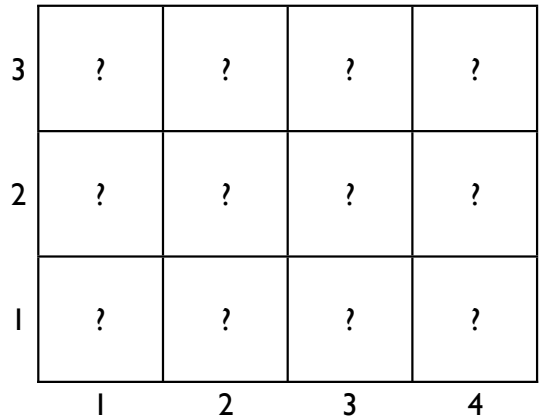


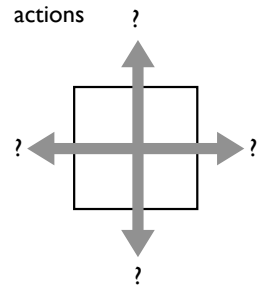
$$T(s, a, s') = P(s_{t+1}|a_t, s_t)$$

- With strong negative reward, best policy is to always seek end states

Reinforcement learning: *learning from experience*

R(s) = ?



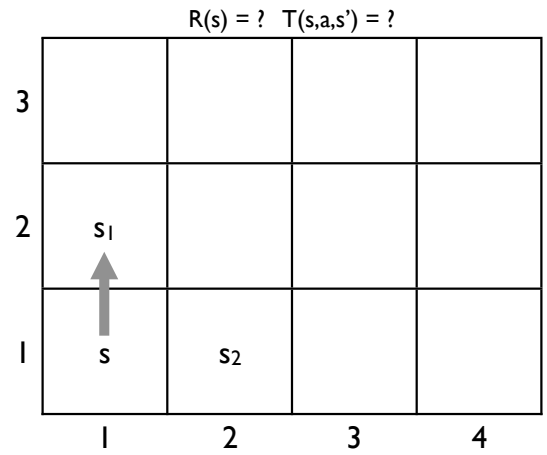


$$T(s, a, s') = ?$$

- Now: don't know environment, transition model, or reward function
- Need to explore the world. At each state the agent:
  1. selects one of the available actions
  2. receives reward
  3. observes resulting state
  4. repeat until a terminal state

In *reinforcement learning*, the objective is to find an optimal policy from these observations.

Model estimation



$$R(s) = \text{reward for state s}$$

$$\hat{T}(s,a,s') \approx \frac{\#\text{ transitions } s \to s' \text{ for action } a}{\#\text{ times } a \text{ selected at state } s}$$

- Observe statistics for actions from each state.
- Eg, for 10 "up" actions from state (1,1) we observe next state is
  - s_1 = (1,2) 8 times & R= -0.04
  - s_2 = (2,1) 2 times & R= -0.04
- ⇒T(s, Up, s_1) = 8/10 = 0.8

  T(s, Up, s_2) = 2/10 = 0.2

  R(s_1) = R(s_2) = 0.04
- Could use Bayesian estimates here
- Continued exploration of the grid world will give increasingly accurate estimates of T(s,a,s') and R(s).

Model estimation

- With estimates of T(s,a,s') and R(s), we can just treat it like an MDP.
- Compute the expected value of each state as:

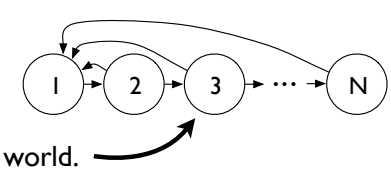
$$V_{i+1}(s) = \hat{R}(s) + \gamma \max_a \sum_{s'} \hat{T}(s,a,s') V_i(s')$$

$$V^*(s) = \lim_{i \to \infty} V_i(s)$$

- And the optimal policy (with value iteration) is:

$$\pi^*(s) = \arg\max_a \left[\hat{R}(s) + \gamma \sum_{s'} \hat{T}(s,a,s') V^*(s') \right]$$

- This is called *Certainty Equivalent* learning.
- Issues and caveats:
  - Could be a very inefficient way to explore the world.
  - When do we stop and estimate the policy?
  - Solving for V*(s) every time could be expensive.



More efficient updating

- How expensive is it to re-evaluate the value equation V(s)?

$$V_{i+1}(s) \quad = \quad \hat{R}(s) + \gamma \max_a \sum_{s'} \hat{T}(s,a,s')V_i(s')$$

  - O(N^3) if solving system of linear equations
  - O(N^2k) for value iteration with k iterations to converge
- But, we don't have to re-solve the entire set of equations.
- At each time step
  - observe the state transition for action a: $s_i(a) \rightarrow s_j$
  - update counts for $\hat{T}(s_i, a, s_j)$
  - do not re-evaluate for V_{i+1} for all of s, only s_i

$$V_{i+1}(s_i) \quad = \quad \hat{R}(s_i) + \gamma \max_a \sum_{s'} \hat{T}(s_i,a,s')V_i(s')$$

"one backup" CE learning

  - this gives an "online" update for V, so now we can use it at anytime
  - involves much less computation at each iteration

Further refinements: prioritized sweeping

- What if values V(s) have already converged for some states?
  - We don't want to keep updating those, each still costs N operations.
  - Which states should we update?
- Prioritized sweeping (Moore & Atkeson):
  - update the ones that have the most potential for change
  - keep a priority queue of which states have biggest potential for changing V(s)
  - Basic algorithm:

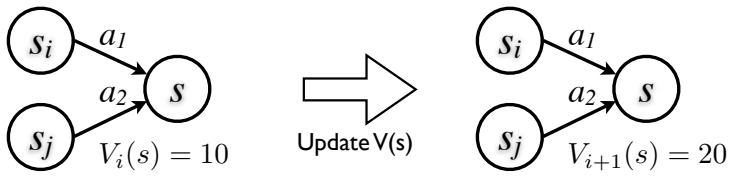
1. set queue to s. To keep track change in V: $V^{prev}(s) = V(s)$
2. update the value equation:

$$V_{i+1}(s_i) \quad = \quad \hat{R}(s_i) + \gamma \max_a \sum_{s'} \hat{T}(s_i,a,s')V_i(s')$$

3. record the change: $\Delta = |V^{\text{prev}}(s) - \hat{V}(s)|$
4. for each state-action (s_p, a_p) that leads to s:
   - add s_p to priority queue with priority: $\Delta\hat{T}(s_p, a_p, s)$

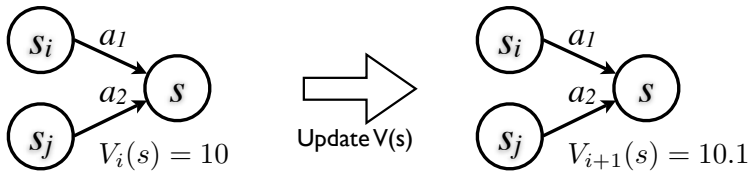
Example: prioritized sweeping

- For a big change in V(s), V(s_i) is likely to change too, so update it.



$$V_{i+1}(s_i) \quad = \quad \hat{R}(s_i) + \gamma \max_a \sum_{s'} \hat{T}(s_i, a, s') V_i(s')$$

- Here, V(s) hardly changes, so updating states is low priority.

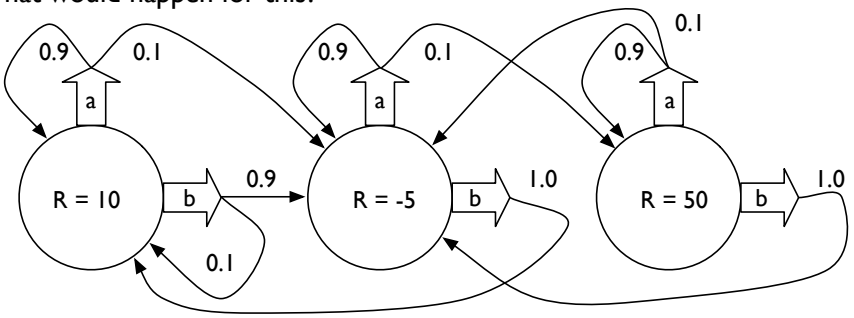


How do we choose the actions?

- How do we ensure that we explore the environment efficiently?
- Should we use the current estimate of the best policy?

$$\pi^*(s) = \arg\max_a \left[\hat{R}(s) + \gamma \sum_{s'} \hat{T}(s, a, s') V^*(s') \right]$$

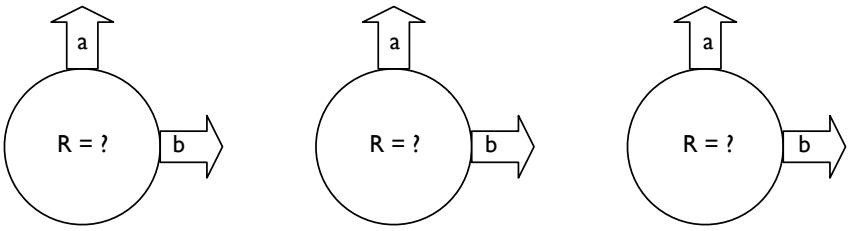
- What would happen for this?



- What about changing environments?

Exploration strategy

- What's the best strategy for exploration?
  - Greedy: always use π*(s)
  - Random
  - Some of both?



The N-arm bandit problem

- Consider a row of N slot machines.
- Each has reward $R(s_i)$ and payoff probability $P(s_i)$
- How do you maximize your payoff when you don't know $R(s_i)$ or $P(s_i)$?

- If you did know the expected reward = $R(s_i)P(s_i)$ then it would be trivial: just pull the lever with highest expected reward

- But what if you don't?
- And what if the payoffs change over time?

- This situation models many real-life decision problems.

The exploitation - exploration dilemma

- Should you exploit the information you've learned

  or explore new options in the hope of greater payoff.

- Possible strategies:
  - ϵ-*greedy* (exploit, but once in a while do something random)
    - choose machine with current best expected reward with probability $1-\epsilon$
    - choose another machine (action) randomly with probability ϵ / (N-1)
  - *Boltzmann exploration* (choose probabilistically relative to expected rewards)
    - choose machine k with probability ~ $\exp(-P(s_k)/T)$
    - decrease T (temperature) over time
      - high T: nearly equiprobable (*exploration*)
      - low T: increasingly biased toward best action (*exploitation*)

Some *real* intelligence

- The exploration/exploitation dilemma is a fundamental aspect of a wide range biological decision making
  - food foraging
  - mate selection
  - choosing nest sites
  - "fight" or "flight" decisions
- Importantly, the environmental statistics are in constant flux.
- There are special neural circuits and neurotransmitters that mediate
  - the memory of short- and long-term reward and
  - the trade-off between exploration and exploitation.
  - we seem to use a Boltzmann-like (softmax) exploration rule (Daw, et al, *Nature* 2006 - using a 4-arm bandit task, no less)

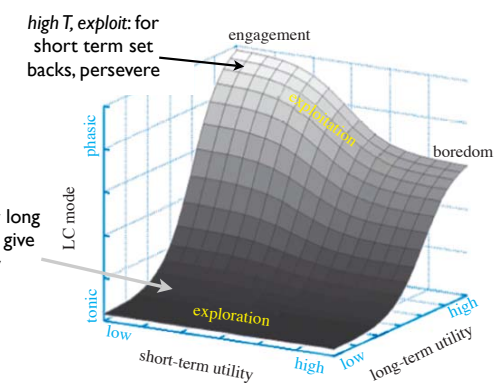
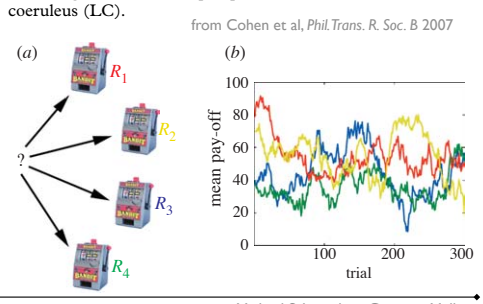


Figure 2. Aston-Jones & Cohen (2005) propose that exploration and exploitation may be mediated by separate short- and long-term measures of utility (cost and reward). Exploration and exploitation, in this model, are mediated by the firing mode of norepinephrine neurons in the locus coeruleus (LC).

from Cohen et al, *Phil.Trans. R. Soc.* B 2007



Temporal difference learning

- Do we have to estimate all the transitions?
- Consider the summation in the Bellman equations: The expected value of each state s equals its own reward plus the (discounted) reward of its successor states:

$$V_{i+1}(s) \quad = \quad \hat{R}(s) + \gamma \max_a \sum_{s'} \hat{T}(s,a,s') V_i(s')$$

- For state s: we receive reward R(s) and take action a = π(s) to transition from s to s'.
- For any successor state s', we can adjust V(s) so it better agrees with the constraint equations, ie closer to the expected future rewards.

New estimate is weighted sum of — current value — expected future reward

$$\begin{aligned} \hat{V}(s) \quad &\leftarrow \quad (1-\alpha)\hat{V}(s) + \alpha(R(s) + \gamma \hat{V}(s')) \\ &= \quad \hat{V}(s) + \alpha(R(s) + \gamma \hat{V}(s') - \hat{V}(s)) \end{aligned}$$

current value — discrepancy between current value and new estimate after transitioning to s'.

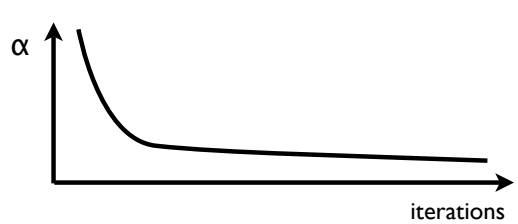
The TD (or temporal difference) equation

The temporal difference equation

$$\hat{V}(s) \quad \leftarrow \quad \hat{V}(s) + \alpha(R(s) + \gamma \hat{V}(s') - \hat{V}(s))$$

- How do we choose α (the learning rate)?
- It's a weighted sum, so 0 < α < 1.
- Too small: V(s) will converge slowly, biased toward current estimate of V
- Too large: V(s) will change quickly, biased toward new estimate

- Decaying learning rate strategy: start large when we're not sure, then reduce as we become more sure of our estimates.



Summary

- types of reinforcement learning, how it's distinct from supervised learning
- relationship to MDPs: MDP but world, reward, and transitions are unknown
- types of updating our estimates
  - certainty equivalent (CE) learning
  - one backup CE learning
  - prioritized sweeping
- TD learning
- Next time: more details and general types of reinforcement learning