CAP6671 Intelligent Systems

Lecture 12:
Reinforcement Learning

Instructor: Dr. Gita Sukthankar
Email: gitars@eecs.ucf.edu

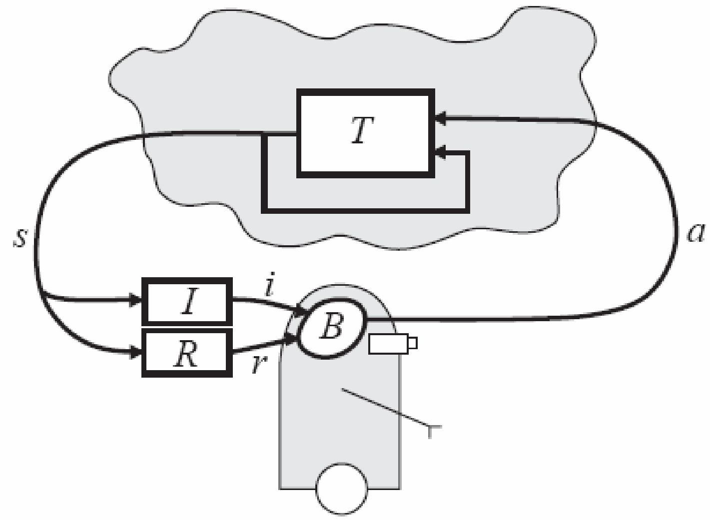
Schedule: T & Th 9:00-10:15am

Location: HEC 302
Office Hours (in HEC 232):
T & Th 10:30am-12

Topics

- Reward models
- Exploration vs. Exploitation
- TD-learning
- Q-learning
- Use of function approximators
- MDP framework
- Value iteration/policy iteration
- CE learning
- Application domains

Problem



Environment:	You are in state 65. You have 4 possible actions.
Agent:	I'll take action 2.
Environment:	You received a reinforcement of 7 units. You are now in state 15. You have 2 possible actions.
Agent:	I'll take action 1.
Environment:	You received a reinforcement of -4 units. You are now in state 65. You have 4 possible actions.
Agent:	I'll take action 2.
Environment:	You received a reinforcement of 5 units. You are now in state 44. You have 5 possible actions.
:	:

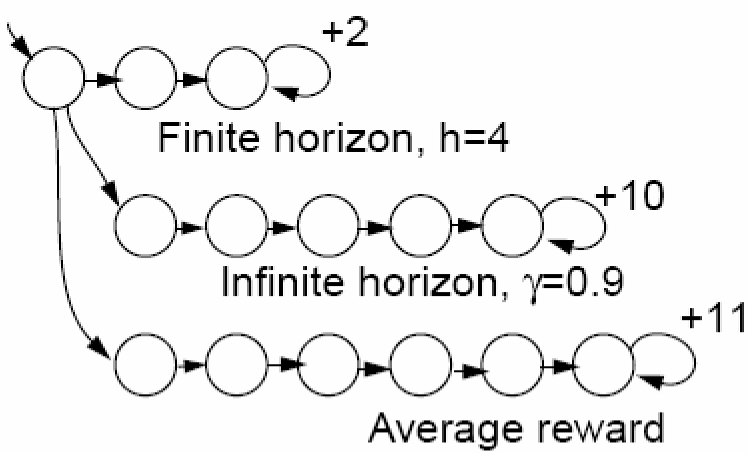
Model

- What assumption do we make about the environment?

- How does this differ from supervised learning?

Reward Horizon

- Different reward models
  - Finite-horizon $E(\sum^{h} r_t)$
  - Infinite-horizon $E(\sum_{t=0}^{\infty} \gamma^t r_t)$
  - Average reward $\lim_{h \to \infty} E(\frac{1}{h} \sum_{t=0}^{h} r_t)$



Case where optimal policy depends on reward model

Determining Action

- If we have an incomplete or imperfect model of the world how does the agent choose an action?
- Ad-hoc strategies
  - Greedy
    - Max observed reward
    - Optimism in tace of uncertainty (optimistic prior is put on action payoffs)
    - Exploration bonuses
  - Randomized
    - Boltzmann exploration

$$P(a) = \frac{e^{ER(a)/T}}{\sum_{a' \in A} e^{ER(a')/T}}$$

(high temperature encourages exploration)

Markov Decision Processes

- Agents actions can affect the state of the world
- Most popular model is the Markov Decision Process
- An MDP has four components, S, A, R, Pr:
  - (finite) state set S   (|S| = n)
  - (finite) action set A   (|A| = m)
  - transition function Pr(s,a,t)
    - each Pr(s,a,-) is a distribution over S
    - represented by set of n x n stochastic matrices
  - bounded, real-valued reward function R(s)
    - represented by an n-vector
    - can be generalized to include action costs: R(s,a)
    - can be stochastic (but replacable by expectation)
- Model easily generalizable to countable or continuous state and action spaces

Assumptions

- Markovian dynamics (history independence)
  - $Pr(S^{t+1}|A^t,S^t,A^{t-1},S^{t-1},\ldots, S^0) = Pr(S^{t+1}|A^t,S^t)$
- Markovian reward process
  - $Pr(R^t|A^t,S^t,A^{t-1},S^{t-1},\ldots, S^0) = Pr(R^t|A^t,S^t)$
- Stationary dynamics and reward
  - $Pr(S^{t+1}|A^t,S^t) = Pr(S^{t'+1}|A^{t'},S^{t'})$ for all t, t'
- **Full observability**
  - though we can't predict what state we will reach when we execute an action, once it is realized, we know what it is

Value Iteration (Bellman 1957)

- Markov property allows exploitation of DP principle for optimal policy construction
  - no need to enumerate $|A|^{Tn}$ possible policies
- Value Iteration

Bellman backup

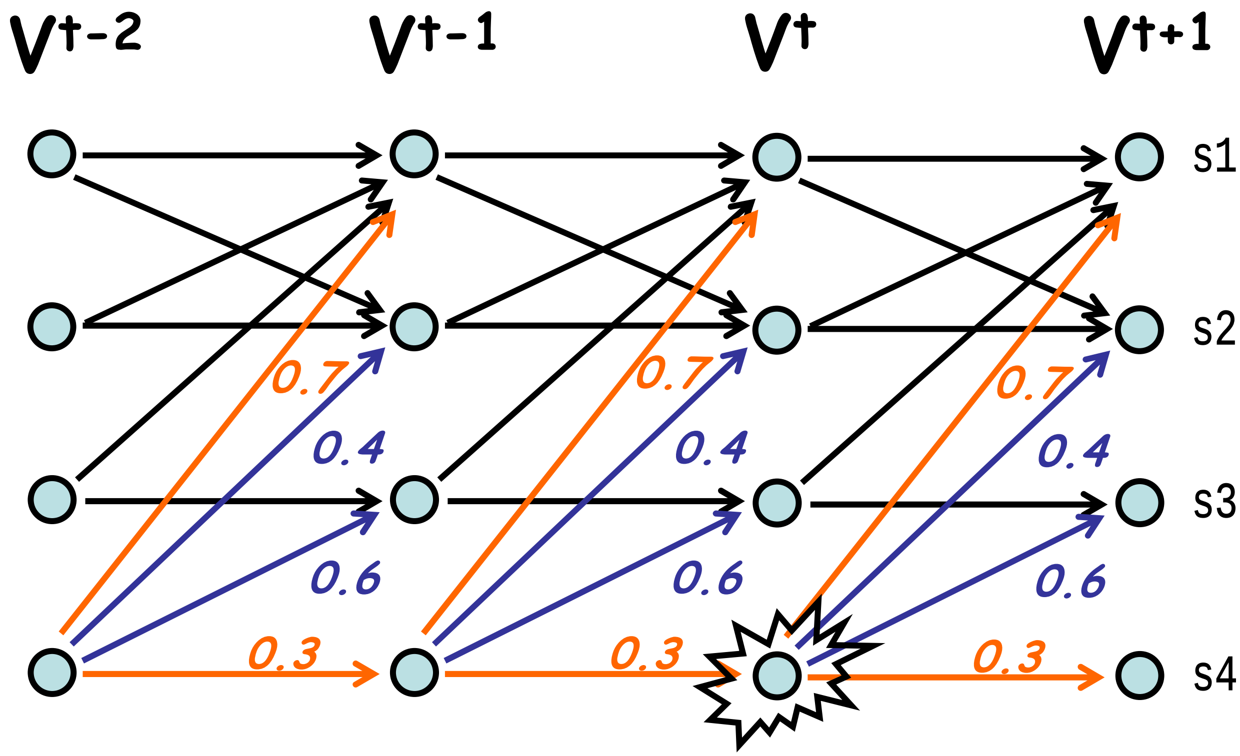
$$V^0(s) = R(s), \quad \forall s$$

$$V^k(s) = R(s) + \max_a \sum_{s'} \Pr(s,a,s') \cdot V^{k-1}(s')$$

$$\pi^*(s,k) = \arg\max_a \sum_{s'} \Pr(s,a,s') \cdot V^{k-1}(s')$$

V^k is optimal k-stage-to-go value function

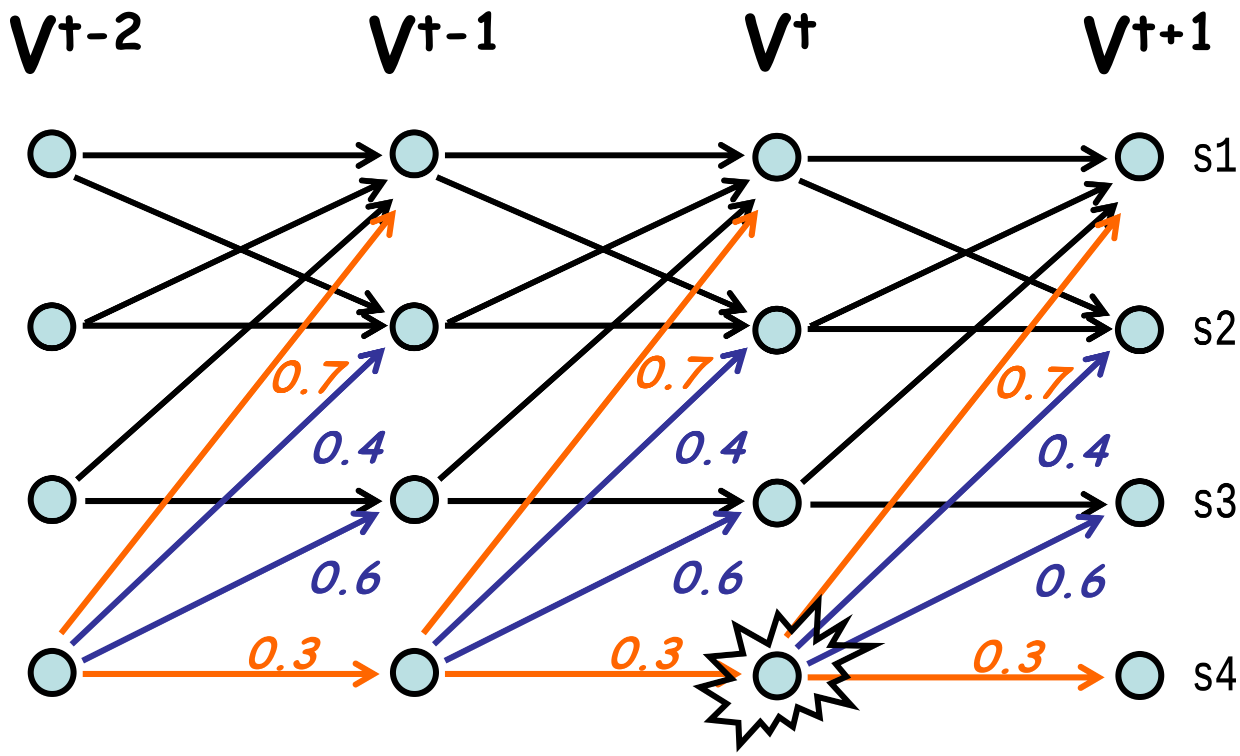
Value Iteration



$V^t(s4) = R(s4) + \max \{ 0.7\, V^{t+1}(s1) + 0.3\, V^{t+1}(s4)$ (orange)

$0.4\, V^{t+1}(s2) + 0.6\, V^{t+1}(s3) \}$ (blue)

Value Iteration



$\Pi^t(s4)$ = max { ■ ■ }

Value Iteration

$V_1(s) := 0$ for all s
$t := 1$
loop
  $t := t + 1$
  loop for all $s \in \mathcal{S}$ and for all $a \in \mathcal{A}$
    $Q_t^a(s) := R(s,a) + \gamma \sum_{s' \in \mathcal{S}} T(s,a,s') V_{t-1}(s')$
    $V_t(s) := \max_a Q_t^a(s)$
  end loop
until $|V_t(s) - V_{t-1}(s)| < \epsilon$ for all $s \in \mathcal{S}$

Policy Iteration

- Manipulate the policy directly rather than finding it indirectly

```
choose an arbitrary policy π'
loop
    π := π'
    compute the value function of policy π:
        solve the linear equations
            V_π(s) = R(s, π(s)) + γ Σ_{s'∈S} T(s, π(s), s') V_π(s')
    improve the policy at each state:
        π'(s) := argmax_a (R(s, a) + γ Σ_{s'∈S} T(s, a, s') V_π(s'))
until π = π'
```

Learning Policy

- Value iteration and policy iteration assume the existence of a known model of the environment
- How can we learn how to act without knowing the model?
  - Model-free approaches
    - Q-learning (most popular)
    - TD-learning
  - Model-based learning
    - Certainty equivalence
    - Dyna
    - Prioritized sweeping

Q-Learning

- Define Q-function

Alpha=learning rate
Gamma=discount reward

$$Q^*(s,a) = R(s,a) + \gamma \sum_{s' \in \mathcal{S}} T(s,a,s') \max_{a'} Q^*(s',a') \quad .$$

- Q-learning rule

$$Q(s,a) := Q(s,a) + \alpha(r + \gamma \max_{a'} Q(s',a') - Q(s,a))$$

Q-function can be stored as a table or can be replaced by a function-approximator

Certainty Equivalence

- Instead of learning a Q-function attempt to learn the transition and reward model by keeping statistics on results of each action
- Then use policy or value iteration to calculate action

Application Domains

- Game playing
  - TD-gammon: backgammon playing
  - Samuels checkers program
- Robotics/control programs
- Every one of the competition domains in the class

Ongoing Research

- Generalizing over multiple problems
  - Multitask learning
- Multi-agent RL