

# Value Functions and Reinforcement Learning

CS 603 - Robotics  
April 2, 2009

# Reinforcement Learning

An Introduction



Richard S. Sutton and Andrew G. Barto

# The Markov Property

- the future is independent of the past, given the present
- $P(s_{t+1}|s_t, a_t, s_{t-1}, a_{t-1}, \dots, s_0, a_0) = P(s_{t+1}|s_t, a_t)$ 
  - “one-step dynamics”

$$M = \langle S, A, \Psi, P, R \rangle$$

$S$  = set of states

$A$  = set of actions

$\Psi$  = action availability at each state ( $|S| \times |A|$ )

$P(s, a, s')$  = probability of transitioning to state  $s'$  if action  $a$  is taken in state  $s$

$R(s, a, s')$  = reward associated with transition from state  $s$  to  $s'$  by action  $a$

# Value Functions

- a *policy*  $\pi(s, a)$  expresses the probability of executing action  $a$  in state  $s$
- *value function*:

$$V^\pi(s) = E_\pi \left\{ \underbrace{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1}}_{\text{return}} \mid s_t = s \right\}$$

- $r_t$  is the reward received at time  $t$  (according to  $R(s, a, s')$ )
- $0.0 < \gamma \leq 1.0$  is *discounting factor*
- $E_\pi$  is an *expectation* under policy  $\pi$  (and  $P, R$ )

# Estimating $V^\pi$ from experience

- start from random initial states
  - (according to *start state distribution*)
- follow  $\pi$
- accumulate average return achieved from each state
- ... those averages will converge to  $V^\pi$
- these are called *Monte Carlo* estimates

# Deriving The Bellman Equation

$$\begin{aligned}V^\pi(s) &= E_\pi \left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s \right\} \\&= E_\pi \left\{ r_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} \mid s_t = s \right\} \\&= \sum_a \pi(s, a) \sum_{s'} P(s, a, s') \left[ R(s, a, s') + \right. \\&\quad \left. \gamma E_\pi \left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} \mid s_{t+1} = s' \right\} \right] \\&= \sum_a \pi(s, a) \sum_{s'} P(s, a, s') [R(s, a, s') + \gamma V^\pi(s')]\end{aligned}$$

# The Bellman Equation for $V^\pi$

$$V^\pi(s) = \sum_a \pi(s, a) \sum_{s'} P(s, a, s') [R(s, a, s') + \gamma V^\pi(s')]$$

- expresses a recursive relationship between the value of a state and the value of its successor states
- $V^\pi$  is the unique solution to its Bellman equation

# The Optimal Value Function

- Value functions provide a partial ordering over policies:
  - if the expected return for  $\pi'$  is greater than that for  $\pi$ , then  $V^{\pi'}(s) \geq V^{\pi}(s) \quad \forall s$
- there is an optimal policy  $\pi^*$  which produces a value function  $V^*$  that dominates all other possible value functions
- $V^*(s) = \max_{\pi} V^{\pi}(s)$
- $V^*(s) = \max_a \sum_{s'} P(s, a, s') [R(s, a, s') + \gamma V^*(s')]$ 
  - this is the *Bellman Optimality Equation*
- any policy that is greedy with respect to  $V^*$  is optimal
- Bellman optimality equation is actually a system of  $|S|$  equations in  $|S|$  unknowns
- but there are more practical approaches than linear programming. . .

- to estimate  $V^\pi$ , we can turn the Bellman equation into an update rule:

$$V_{k+1}(s) = \sum_a \pi(s, a) \sum_{s'} P(s, a, s') [R(s, a, s') + \gamma V_k(s')]$$

- can then apply numerical relaxation techniques such as those discussed last time (Jacobi, Gauss-Seidel)

# Policy Improvement

- goal is generally to improve the policy. How do we know if a policy is better?
- can evaluate the utility of choosing action  $a$ , then following the existing policy  $\pi$  thereafter:

$$Q^\pi(s, a) = \sum_{s'} P(s, a, s') [R(s, a, s') + \gamma V^\pi(s')]$$

- the *policy improvement theorem* states that:  
if  $Q^\pi(s, a) > V^\pi(s)$ ,  
then it would be better to change  $\pi$  so that it always chooses  $a$  in state  $s$ .
  - we can do this for all states to get a better policy

# Generalized Policy Iteration

- We can interleave policy evaluation and policy improvement until we get the optimal policy.
- (How will we know it's optimal?)
- This is called *policy iteration*.
- We don't have to do policy evaluation all the way to convergence each time.
  - If we do just one sweep, that's called *value iteration*. It converges, too.

- usually we want to focus on the most relevant parts of the state space,
- and usually we don't know  $P(s, a, s')$  and  $R(s, a, s')$

$$Q_{t+1}(s, a) \leftarrow (1 - \alpha)Q_t(s, a) + \alpha \left[ r_{t+1} + \gamma \max_a Q_t(s', a) \right]$$

- $\alpha$  is a learning rate
- example of a *Temporal Difference* method
- *off-policy*