150AML
Spring 2011

Summary of part I
Problem Formulation and Algorithms from [SB]

MDP Model

- Given by transitions Pr(s'|s,a)
- reward r(s,a)
- Criterion: expected total discounted reward (discount factor gamma)

MDP Model

- Two problems:
- calculate value of policy
- calculate optimal value and policy
- We may or may not have a model and may or may not construct one during calculation

Backup Operators

Bellman Backup

$$[B(V)](s) = \max_a [r(s,a) + \sum_{s'} \Pr(s'|s,a)V(s')]$$

Bellman Backup restricted to policy

$$[B^{\Pi}(V)](s) = r(s,\Pi(s)) + \sum_{s'} \Pr(s'|s,\Pi(s))V(s')$$

Bellman Backup restricted to state

$$[B_{s^*}(V)](s^*) = \max_a [r(s^*,a) + \sum_{s'} \Pr(s'|s^*,a)V(s')]$$

$$[B_{s^*}(V)](s) = V(s) \text{ when } s \neq s^*$$

Policy Evaluation (calculate V^Pi)

- Solve linear equations $V = B^{\Pi}(V)$
- Iterative Alg: Repeat $V \Leftarrow B^{\Pi}(V)$
- Monte Carlo: $V(s_t) \Leftarrow V(s_t) + \alpha[R_t - V(s_t)]$
- TD(0): $V(s_t) \Leftarrow V(s_t) + \alpha[r_t + \gamma V(s_{t+1}) - V(s_t)]$
- TD(lambda): l=0 => TD(0) l=1=>MC

Optimization

- VI: Repeat $V \Leftarrow B(V)$
- PI: Repeat $\Pi = \text{greedy(V)}; V = V^{\Pi}$
- MPI: Repeat
  $[\Pi = \text{greedy(V)}; \text{Repeat m times} : V = B^{\Pi}(V)]$
- Linear Programming
  $\text{minimize} \sum_s V(s) \quad \text{subject to } V \geq B^a(\text{V})$

On Line Optimization (RTDP)

Repeat:
pick start state s
Repeat: [in state s]

$$\forall a,\ Q(s,a) = r(s,a) + \gamma \sum_{s'} \Pr(s' \mid s,a)[\max_{a'} Q(s',a')]$$

P = greedy or epsilon-greedy w.r.t. Q
Choose action a using policy P
Sample s' from Pr(s'|s,a)
s=s'

Note: This performs $B_s(V)$ on the Q() representation

On Line Optimization (SARSA)

Repeat:
[in state s] take action a; observe r,s'
choose next action a' using policy P

$$Q(s,a) \Leftarrow Q(s,a) + \alpha[r + \gamma Q(s',a') - Q(s,a)]$$

P = epsilon-greedy w.r.t. Q

s=s'; a=a'

On Line Optimization (SARSA lambda)

Repeat:
[in state s] take action a; observe r,s'
choose next action a' using policy P

$$\delta = r + \gamma Q(s',a') - Q(s,a)$$
$$e(s,a) = e(s,a) + 1$$
$$\text{for all } (s^*,a^*), Q(s^*,a^*) = Q(s^*,a^*) + \alpha\delta\, e(s^*,a^*)$$
$$e(s^*,a^*) = \gamma\lambda\, e(s^*,a^*)$$

P = epsilon-greedy w.r.t. Q
s=s'; a=a'

On Line Optimization (Q learning)

Repeat:
[in state s] take exploration policy action a;
observe r,s'

$$Q(s,a) \Leftarrow Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)]$$

State Aggregation

- Partition states into disjoint sets
  $p_1, p_2, \ldots, p_n$
- State s mapped to p_i=p(s)

$$[B_{s^*}(V)](p(s^*)) = \max_a [r(s^*,a) + \sum_{s'} \Pr(s' \mid s^*,a) V(p(s'))]$$
$$[B_{s^*}(V)](p(s)) = V(p(s)) \text{ when } p(s) \neq p(s^*)$$

- Parametric aggregation more general

On Line Optimization (SARSA)

Repeat:
[in state s] take action a; observe r,s'
choose next action a' using policy P

$$Q(s,a) \Leftarrow Q(s,a) + \alpha[r + \gamma Q(s',a') - Q(s,a)]$$

P = epsilon-greedy w.r.t. Q

s=s'; a=a'

On Line Optimization (SARSA)

(with linear function approximation)

Repeat:

[in state s] take action a; observe r,s'

choose next action a' using policy P

$$Q(s,a) \Leftarrow Q(s,a) + \alpha[r + \gamma Q(s',a') - Q(s,a)]$$

$$\vec{w} \Leftarrow \vec{w} + \alpha[r + \gamma Q_{\vec{w}}(s',a') - Q_{\vec{w}}(s,a)]\vec{\phi}(s,a)$$

P = epsilon-greedy w.r.t. Q

(P represented implicitly)

s=s'; a=a'

On Line Optimization (SARSA)

(with linear function approximation)

Repeat:

[in state s] take action a; observe r,s'

choose next action a' using policy P

$$\vec{w} \Leftarrow \vec{w} + \alpha[r + \gamma Q_{\vec{w}}(s',a') - Q_{\vec{w}}(s,a)]\vec{\phi}(s,a)$$

P = epsilon-greedy w.r.t. Q

(P represented implicitly)

s=s'; a=a'

Dyna-SARSA-RTDP

Repeat:

[in state s] take action a; observe r,s'

choose next action a' using policy P

$$Q(s,a) \Leftarrow Q(s,a) + \alpha[r + \gamma Q(s',a') - Q(s,a)]$$

$$r(s,a) = \ldots \quad \mathrm{p}(s^* \mid s,a) = \ldots$$

Update Q: RTDP with r(), p() estimates

P = epsilon-greedy w.r.t. Q

s=s'; a=a'