

## VI, PE, PI

### Value Iteration

Optimal policy:

$$\pi^*(s) = \operatorname{argmax}_{\pi} V^{\pi}(s) = \operatorname{argmax}_{a \in A(s)} \sum_{s'} P(s'|s, a) V(s')$$

Bellman Equation:

$$V(s) = R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s'|s, a) V(s')$$

Bellman Update:

$$V_{i+1}(s) \leftarrow BV_i = R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s'|s, a) V_i(s')$$

Bellman=Contraction:

$$\|V\| = \max_s |V(s)|$$

$$\|BV_i - BV'_i\| \leq \gamma \|V_i - V'_i\|$$

$$\|BV_i - V\| \leq \gamma \|V_i - V\|$$

Error of the estimate  $V_i$ :

$$\|V_i - V\|$$

$$\|V_0 - V\| \leq 2R_{max}/(1 - \gamma)$$

Bound on state values (utilities):

$$V(s) \leq \pm R_{max}/(1 - \gamma)$$

To get  $\|V_i - V\| \leq \epsilon$ :

$$\gamma^N 2R_{max}/(1 - \gamma) \leq \epsilon$$

$$N = \left\lceil \frac{\log(2R_{max}/(\epsilon(1 - \gamma)))}{\log(1/\gamma)} \right\rceil$$

If  $\|V_{i+1} - V_i\| \leq \epsilon(1 - \gamma)/\gamma$  then  $\|V_{i+1} - V\| < \epsilon$

Policy loss is  $\|V^{\pi_i} - V\|$  and is connected to  $V_i$ :

$$\text{if } \|V_i - V\| < \epsilon \text{ then } \|V^{\pi_i} - V\| < 2\epsilon\gamma/(1 - \gamma)$$

**function** VALUE-ITERATION( $mdp, \epsilon$ ) **returns** a utility function

**inputs:**  $mdp$ , an MDP with states  $S$ , actions  $A(s)$ , transition model  $P(s' | s, a)$ , rewards  $R(s)$ , discount  $\gamma$   
 $\epsilon$ , the maximum error allowed in the utility of any state

**local variables:**  $U, U'$ , vectors of utilities for states in  $S$ , initially zero  
 $\delta$ , the maximum change in the utility of any state in an iteration

```
repeat
   $U \leftarrow U'; \delta \leftarrow 0$ 
  for each state  $s$  in  $S$  do
     $U'[s] \leftarrow R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s' | s, a) U[s']$ 
    if  $|U'[s] - U[s]| > \delta$  then  $\delta \leftarrow |U'[s] - U[s]|$ 
until  $\delta < \epsilon(1 - \gamma)/\gamma$ 
return  $U$ 
```

### Policy Iteration

Policy Evaluation: Given a policy  $\pi_i$ , calculate  $V_i = V^{\pi_i}$ , the utility of each state if  $\pi_i$  were to be executed. Do this by running Value Iteration with (or directly solving in  $O(n^3)$  time the set of linear equations defined by):

$$V_i(s) = R(s) + \gamma \sum_{s'} P(s'|s, \pi_i(s)) V_i(s')$$

Policy Improvement: Calculate a new MEU policy  $\pi_{i+1}$ , using one-step look-ahead based on  $V_i$ .

Terminate when policy improvement yields no change in the utilities.

**function** POLICY-ITERATION( $mdp$ ) **returns** a policy

**inputs:**  $mdp$ , an MDP with states  $S$ , actions  $A(s)$ , transition model  $P(s' | s, a)$   
**local variables:**  $U$ , a vector of utilities for states in  $S$ , initially zero  
 $\pi$ , a policy vector indexed by state, initially random

```
repeat
   $U \leftarrow \text{POLICY-EVALUATION}(\pi, U, mdp)$ 
   $unchanged? \leftarrow \text{true}$ 
  for each state  $s$  in  $S$  do
    if  $\max_{a \in A(s)} \sum_{s'} P(s' | s, a) U[s'] > \sum_{s'} P(s' | s, \pi[s]) U[s']$  then do
       $\pi[s] \leftarrow \operatorname{argmax}_{a \in A(s)} \sum_{s'} P(s' | s, a) U[s']$ 
       $unchanged? \leftarrow \text{false}$ 
until  $unchanged?$ 
return  $\pi$ 
```

### TD-Learning

Off-Policy (passive) TD-Learning Update:

$$V^{\pi}(s) \leftarrow V^{\pi}(s) + \alpha(R(s) + \gamma V^{\pi}(s') - V^{\pi}(s))$$

#### Semi-gradient TD(0) for estimating $\hat{v} \approx v_{\pi}$

Input: the policy  $\pi$  to be evaluated  
Input: a differentiable function  $\hat{v} : S^+ \times \mathbb{R}^n \rightarrow \mathbb{R}$  such that  $\hat{v}(\text{terminal}, \cdot) = 0$

Initialize value-function weights  $\theta$  arbitrarily (e.g.,  $\theta = 0$ )

Repeat (for each episode):

  Initialize  $S$

  Repeat (for each step of episode):

    Choose  $A \sim \pi(\cdot | S)$

    Take action  $A$ , observe  $R, S'$

$$\theta \leftarrow \theta + \alpha[R + \gamma \hat{v}(S', \theta) - \hat{v}(S, \theta)] \nabla \hat{v}(S, \theta)$$

$S \leftarrow S'$

  until  $S'$  is terminal

### Monte-Carlo

#### Gradient Monte Carlo Algorithm for Approximating $\hat{v} \approx v_{\pi}$

Input: the policy  $\pi$  to be evaluated

Input: a differentiable function  $\hat{v} : S \times \mathbb{R}^n \rightarrow \mathbb{R}$

Initialize value-function weights  $\theta$  as appropriate (e.g.,  $\theta = 0$ )

Repeat forever:

  Generate an episode  $S_0, A_0, R_1, S_1, A_1, \dots, R_T, S_T$  using  $\pi$

  For  $t = 0, 1, \dots, T - 1$ :

$$\theta \leftarrow \theta + \alpha[G_t - \hat{v}(S_t, \theta)] \nabla \hat{v}(S_t, \theta)$$

### SARSA

#### Episodic Semi-gradient Sarsa for Estimating $\hat{q} \approx q_*$

Input: a differentiable function  $\hat{q} : S \times \mathcal{A} \times \mathbb{R}^n \rightarrow \mathbb{R}$

Initialize value-function weights  $\theta \in \mathbb{R}^n$  arbitrarily (e.g.,  $\theta = 0$ )

Repeat (for each episode):

$S, A \leftarrow$  initial state and action of episode (e.g.,  $\epsilon$ -greedy)

  Repeat (for each step of episode):

    Take action  $A$ , observe  $R, S'$

    If  $S'$  is terminal:

$$\theta \leftarrow \theta + \alpha[R - \hat{q}(S, A, \theta)] \nabla \hat{q}(S, A, \theta)$$

    Go to next episode

    Choose  $A'$  as a function of  $\hat{q}(S', \cdot, \theta)$  (e.g.,  $\epsilon$ -greedy)

$$\theta \leftarrow \theta + \alpha[R + \gamma \hat{q}(S', A', \theta) - \hat{q}(S, A, \theta)] \nabla \hat{q}(S, A, \theta)$$

$S \leftarrow S'$

$A \leftarrow A'$