VI, PE, PI

Value Iteration

Optimal policy:

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

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:

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

$$\|BV_i - BV_i'\| \le \gamma\|V_i - V_i'\|$$

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

Error of the estimate V_i:

$$\|V_i - V\|$$

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

Bound on state values (utilities):

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

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

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

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

If $\|V_{i+1} - V_i\| \le \epsilon(1-\gamma)/\gamma$ then $\|V_{i+1} - V\| < \epsilon$
Policy loss is $\|V^{\pi_i} - V\|$ and is connected to V_i:

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

function VALUE-ITERATION(mdp, ϵ) **returns** a utility function
  inputs: mdp, an MDP with states S, actions $A(s)$, transition model $P(s' \mid s, a)$,
    rewards $R(s)$, discount γ
    ϵ, the maximum error allowed in the utility of any state
  local variables: U, U', vectors of utilities for states in S, initially zero
    δ, 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**
      $U'[s] \leftarrow R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s' \mid s, a)\, U[s']$
      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 π_i, calculate $V_i = V^{\pi_i}$, the utility of each state if π_i were t 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 π_{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 $P(s' \mid s, a)$
  local variables: U, a vector of utilities for states in S, initially zero
    π, a policy vector indexed by state, initially random

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

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

Initialize value-function weights $\boldsymbol{\theta}$ arbitrarily (e.g., $\boldsymbol{\theta} = \mathbf{0}$)
Repeat (for each episode):
  Initialize S
  Repeat (for each step of episode):
    Choose $A \sim \pi(\cdot|S)$
    Take action A, observe R, S'
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha\big[R + \gamma\hat{v}(S',\boldsymbol{\theta}) - \hat{v}(S,\boldsymbol{\theta})\big]\nabla\hat{v}(S,\boldsymbol{\theta})$
    $S \leftarrow S'$
  until S' is terminal

Monte-Carlo

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

Input: the policy π to be evaluated
Input: a differentiable function $\hat{v} : \mathcal{S} \times \mathbb{R}^n \to \mathbb{R}$

Initialize value-function weights $\boldsymbol{\theta}$ as appropriate (e.g., $\boldsymbol{\theta} = \mathbf{0}$)
Repeat forever:
  Generate an episode $S_0, A_0, R_1, S_1, A_1, \ldots, R_T, S_T$ using π
  For $t = 0, 1, \ldots, T-1$:
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha\big[G_t - \hat{v}(S_t,\boldsymbol{\theta})\big]\nabla\hat{v}(S_t,\boldsymbol{\theta})$

SARSA

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

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

Initialize value-function weights $\boldsymbol{\theta} \in \mathbb{R}^n$ arbitrarily (e.g., $\boldsymbol{\theta} = \mathbf{0}$)
Repeat (for each episode):
  $S, A \leftarrow$ initial state and action of episode (e.g., ε-greedy)
  Repeat (for each step of episode):
    Take action A, observe R, S'
    If S' is terminal:
      $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha\big[R - \hat{q}(S, A, \boldsymbol{\theta})\big]\nabla\hat{q}(S, A, \boldsymbol{\theta})$
      Go to next episode
    Choose A' as a function of $\hat{q}(S', \cdot, \boldsymbol{\theta})$ (e.g., ε-greedy)
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha\big[R + \gamma\hat{q}(S', A', \boldsymbol{\theta}) - \hat{q}(S, A, \boldsymbol{\theta})\big]\nabla\hat{q}(S, A, \boldsymbol{\theta})$
    $S \leftarrow S'$
    $A \leftarrow A'$