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

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

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

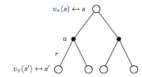
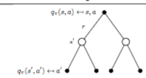
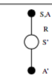
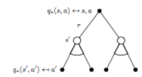
If $\|V_{i+1} - V_i\| \le \epsilon(1-\gamma)/\gamma$ then $\|V_{i+1} - V\| < \epsilon$
Policy loss is $\|V^{\pi_i} - V\|$ and is connected to V_i:

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

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.

Problem	Bellman Equation	Algorithm
Prediction	Bellman Expectation Equation	Iterative Policy Evaluation
Control	Bellman Expectation Equation + Greedy Policy Improvement	Policy Iteration
Control	Bellman Optimality Equation	Value Iteration

- Algorithms are based on state-value function $v_\pi(s)$ or $v_*(s)$
- Complexity $O(mn^2)$ per iteration, for m actions and n states
- Could also apply to action-value function $q_\pi(s,a)$ or $q_*(s,a)$
- Complexity $O(m^2n^2)$ per iteration

	Full Backup (DP)	Sample Backup (TD)
Bellman Expectation Equation for $v_\pi(s)$	 Iterative Policy Evaluation	 TD Learning
Bellman Expectation Equation for $q_\pi(s,a)$	 Q-Policy Iteration	 Sarsa
Bellman Optimality Equation for $q_*(s,a)$	 Q-Value Iteration	 Q-Learning

Full Backup (DP)	*Sample Backup (TD)*
Iterative Policy Evaluation $V(s) \leftarrow \mathbb{E}[R + \gamma V(S') \mid s]$	TD Learning $V(S) \xleftarrow{\alpha} R + \gamma V(S')$
Q-Policy Iteration $Q(s,a) \leftarrow \mathbb{E}[R + \gamma Q(S',A') \mid s,a]$	Sarsa $Q(S,A) \xleftarrow{\alpha} R + \gamma Q(S',A')$
Q-Value Iteration $Q(s,a) \leftarrow \mathbb{E}\left[R + \gamma \max_{a'\in\mathcal{A}} Q(S',a') \mid s,a\right]$	Q-Learning $Q(S,A) \xleftarrow{\alpha} R + \gamma \max_{a'\in\mathcal{A}} Q(S',a')$

where $x \xleftarrow{\alpha} y \equiv x \leftarrow x + \alpha(y - x)$

MC converges to solution with minimum mean-squared error

- Best fit to the observed returns

$$\sum_{k=1}^{K}\sum_{t=1}^{T_k}\left(G_t^k - V(s_t^k)\right)^2$$

- In the AB example, $V(A) = 0$

TD(0) converges to solution of max likelihood Markov model

- Solution to the MDP $\langle \mathcal{S}, \mathcal{A}, \hat{\mathcal{P}}, \hat{\mathcal{R}}, \gamma \rangle$ that best fits the data

$$\hat{\mathcal{P}}_{s,s'}^a = \frac{1}{N(s,a)}\sum_{k=1}^{K}\sum_{t=1}^{T_k}\mathbf{1}(s_t^k, a_t^k, s_{t+1}^k = s, a, s')$$

$$\hat{\mathcal{R}}_s^a = \frac{1}{N(s,a)}\sum_{k=1}^{K}\sum_{t=1}^{T_k}\mathbf{1}(s_t^k, a_t^k = s, a)r_t^k$$

- In the AB example, $V(A) = 0.75$

On/Off-Policy	Algorithm	Table Lookup	Linear	Non-Linear
On-Policy	MC	✓	✓	✓
	LSMC	✓	✓	-
	TD	✓	✓	✗
	LSTD	✓	✓	-
Off-Policy	MC	✓	✓	✓
	LSMC	✓	✓	-
	TD	✓	✗	✗
	LSTD	✓	✓	-

Sarsa converges to the optimal action-value function, $Q(s,a) \rightarrow q_*(s,a)$, under the following conditions:

- *GLIE sequence of policies $\pi_t(a|s)$*
- *Robbins-Monro sequence of step-sizes α_t*

$$\sum_{t=1}^{\infty} \alpha_t = \infty$$
$$\sum_{t=1}^{\infty} \alpha_t^2 < \infty$$

Algorithm	Table Lookup	Linear	Non-Linear
Monte-Carlo Control	✓	(✓)	✗
Sarsa	✓	(✓)	✗
Q-learning	✓	✗	✗
LSPI	✓	(✓)	-

(✓) = chatters around near-optimal value function

PAC

w.p. at least $1-\delta$, on all but N_ϵ steps, select a for state s that is ϵ-close to V^*.

$$|Q(s,a) - V^*(s)| < \epsilon$$

i.e.

$$P(N_\epsilon \le F_{\text{poly}}(|S|, |A|, \delta, \epsilon, \gamma)) \ge 1 - \delta$$

where N_ϵ is the number of episodes not ϵ close to optimal.

Comparisons/Overview

Model Free: Derive optimal policy without learning the model (transitions and rewards, leads to weaker theoretical results; ideal for large state space; dont learn a model learn value function or policy directly
Model Based: Learn transition and reward model, use it to get optimal policy, leads to stronger theoretical results (e.g. R_{max} satisfies PAC); ideal for small state space; more exploration (more of the model is learned)
Monte Carlo: resets (assume restart from state several times), wasteful (same trajectory traversed many times), does not assume Markov (works better for non-Markov), high variance, zero bias, not sensitive to initial value, wait till end of episode for reward, converges to minimum least square error between values and observed returns
TD: learns at every step, learns from incomplete sequences, works in non-terminating environments, more sensitive to initial value, more efficient, converges to solution of max-likelihood model
Value Iteration: Active, Model-Based, Neither
Policy Iteration: Active, Model-Based, Neither
Policy Evaluation: Passive, Model-Based, Neither
Monte Carlo: (Pred=Passive, Ctrl=Active), Model-Free, Either
TD-Learning: (Pred=Passive, Ctrl=Active, Model-Free, Either
Q-Learning: Active, Model-Free, Off-Policy
SARSA: Active, Model-Free, On-Policy
Passive: Agent executes a fixed policy and evaluates it **Active**: Agents updates policy as it learns