**Learning Algorithms for Solving MDPs**

References: Barto, Bradtke and Singh (1995)

"Learning to Act Using Real-Time Dynamic Programming" in *Machine Learning* (also on WWW)

**1. Q-Learning** Given an MDP problem, define the $Q$ function by:

$$Q(s,a) = u(s,a) + \beta E_a V(s) \tag{1}$$

i.e. $Q(s,a)$ is the expected present discounted value of taking action $a$ in state $s$. Let $\tilde{s}'(s,a)$ denote the (random) successor to $(s,a)$, i.e. a random draw from $p(s'|s,a)$. Then if $Q_t(s,a)$ denotes the estimate of the $Q$ function at iteration $t$ of the $Q$-learning algorithm, we have:

$$\begin{aligned} &Q_{t+1}(s,a) = Q_t(s,a) + \\ &\lambda_t(s,a)\left[u(s,a) + \beta \max_{a' \in A(\tilde{s}'(s,a))} Q_t(\tilde{s}'(s,a), a') - Q_t(s,a)\right] \end{aligned} \tag{2}$$

**Comment:** Each of the $|S| \times |A|$ $(s,a)$ pairs is updated at each step of the $Q$-learning algorithm. At any stage $t$ the recommended action in state $s$ is the action $a$ for which the corresponding $Q_t(s,a)$ is the highest, i.e.

$$\alpha_t(s) = \underset{a \in A(s)}{argmax}\, Q_t(s,a) \tag{3}$$

**Comment:** $\{\lambda_t\}$ are a sequence of possibily stochastic *step size* functions satisfying for each $(s,a)$:

$$\begin{aligned} &\lambda_t(s,a) \in [0,1] \\ &\sum_{t=0}^{\infty} \lambda_t(s,a) = \infty \quad \text{w.p.1} \\ &\sum_{t=0}^{\infty} \lambda_t^2(s,a) \leq C \quad \text{w.p.1} \end{aligned} \tag{4}$$

## Relationship to Stochastic Approximation

1. **Standard stochastic approximation:** Find fixed point $F(x^*) = x^*$ where $F : R^N \to R^N$ by
$$x_{t+1} = x_t + \lambda_t \left[F(x_t) - x_t + \epsilon_t\right] \tag{5}$$
where $\{\epsilon_t\}$ is a sequence of stochastic shocks satisfying
$$\begin{aligned} &E\{\epsilon_t | \mathcal{F}_t\} = 0 \\ &\|\text{var}(\epsilon_t)\| \leq A + B\|x_t\| \end{aligned} \tag{6}$$

2. **Asynchronous stochastic approximation:** Only update or "back up" some of the components of $x_t$ at time $t$. Let $T_i$ be an infinite sequence of times at which state $i$ is updated. Then
$$x_{t+1}(i) = \begin{cases} x_t(i) & t \notin T_i \\ x_t(i) + \lambda_t(i)\left[F_i(x_t) - x_t(i) + \epsilon_t(i)\right] & t \in T_i \end{cases} \tag{7}$$
where
$$x_t(i) = \Big(x_1(\tau_1^i(t)), \ldots, x_N(\tau_N^i(t))\Big)$$
where $\tau_j^i(t)$ denotes the last time component $j$ of $x_t$ was updated. Obviously $\tau_j^i(t) \leq t$.

**Theorem (Tsitsiklis, 1993)** *If for all $i$ and $j$ we have $lim_{t\to\infty} \tau_j^i(t) = \infty$ with probability $1$ and if $F$ is a contraction mapping, then $x_t \to x^*$ with probability 1.*

**Corollary:** *If $\beta < 1$ then we have $Q_t(s,a) \to Q(s,a)$ with probability 1 for each $(s,a) \in S \times A$.*

**Comment:** It is not necessary to back up each $(s,a)$ pair at every iteration $t$ of the $Q$-learning algorithm. However it is necessary that every $(s,a)$ pair be backed up *infinitely often.* This corresponds to the assumption that $lim_{t\to\infty} \tau_j^i(t) = \infty$ with probability 1 in Tsitsiklis's theorem.

**Real Time Dynamic Programming** A learning algorithm closely related to *successive approximations* that works in *real time,* i.e. you update value function during the process of controlling system.

0. **Successive Approximations**

$$\begin{aligned} V_{t+1}(s) &= \max_{a \in A(s)} \left[ u(s,a) + \beta E_a V_t(s) \right] \\ &\equiv \max_{a \in A(s)} Q_t(s, a | V_t) \end{aligned} \tag{5}$$

1. **Gauss-Seidel**

$$\begin{aligned} V_{t+1}(s_i) &= \max_{a \in A(s_i)} \left[ u(s,a) + \beta E_a V_{t,i}(s) \right] \\ &= \max_{a \in A(s_i)} Q_t(s, a | V_{i,t}). \end{aligned} \tag{6}$$

where

$$\hat{V}_{t,i}(s_j) = \begin{cases} V_{t+1}(s_j) & \text{if } j < i \\ V_t(s_j) & \text{otherwise} \end{cases} \tag{7}$$

2. **Asynchronous Successive Approximations**

$$V_{t+1}(s) = \begin{cases} \max_{a \in A(s)} Q_t(s, a | V_t) & \text{if} s \in B_t \\ V_t(s) & \text{otherwise.} \end{cases} \tag{8}$$

**Comment:** $B_t$ denotes the set of states $s \in S$ which are *backed up* at iteration $t+1$ of the asynchronous successive approximations algorithm. Successive approximations is a special case when $B_t = S$ (all states backed up) and Gauss-Seidel method is a special case when $B_0 = \{s_1\}$, $B_1 = \{s_2\}, \ldots, B_{N-1} = \{s_N\}$, $B_N = \{s_1\}, \ldots$

3. **Real Time Dynamic Programming** This is a special case of asynchronous successive approximations when $B_t = \{s_t\}$ where $s_t$ is the realized state of the process at time $t$.

**Theorem:** *In discounted finite MDPs, if* $\{V_t\}$ *denotes the sequence of value functions from asynchronous successive approximations, then* $\|V_t - V\| \rightarrow 0$ *with probability 1 provided each state* $s \in S$ *is backed up infinitely often.*