N-Step Bootstrapping

We want to get less variance by waiting for reward from n-steps forward.

For just 1-Step bootstrapping:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)] \tag{1}$$

where we bootstrap for current state's return with estimate of state value from next immediate step onwards:

$$R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}). \tag{2}$$

For 2-step bootstrapping, we bootstrap for current state with estimate of state value from 2 steps onwards:

$$G_{t:t+2} = R_{t+1} + \gamma R_{t+2} + \gamma^2 Q(S_{t+2}, A_{t+2}). \tag{3}$$

So, n-step boostrapping means, we only estimate the state value from nth step onwards:

$$G_{t:t+n} = R_{t+1} + \gamma R_{t+2} + ... + \gamma^{n-1} R^{n-1} + \gamma^n Q(S_{t+n}, A_{t+n}). \tag{4}$$

Thus, the update rule can be reexpressed by the new term for n-step bootstrapped returns:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[G_{t:t+n} - Q(S_t, A_t)] \tag{5}$$

where $G_{t:t+n} - Q(S_t, A_t)$ is the n-step Temporal-Difference Error.

Monte Carlo is basically the extreme version of n-step SARSA where you track rewards all the way till the end and don't estimate/bootstrap at all:

$$Q(S_t, A_t) \leftarrow G_{t:t+T}. \tag{6}$$