<u>Chapter 7 - Multi-step Bootstrapping</u> (Aravind S - EE14B013 - RL project RA 4)

n-step methods

- MC methods use an exact sample of return (till episode termination) whereas TD methods use an estimate using a one-step look ahead.
- Both of these are 2 extremes and intermediate methods which estimate return using a look ahead of n-steps might work better.
- Also, in TD: you perform updates for each time step by bootstrapping on the estimate of the next-state's value. But, bootstrapping works better if the estimates differ over a large range temporally i.e in many cases, successive states might be correlated and as a result, there won't be much of a difference between successive state's value functions. As a result the increments will be less in magnitude and learning will be slower.
- On the other hand, n-step returns consider bootstrapping with a state that occurs n-steps down the line. Thus, there would be a good difference between the 2 states (S_t and S_{t+n}). This results in better learning.

n-step TD

- From a state s, the target for a n-step backup is the n-step return $G_t^{(n)}$.

$$G_t^{(n)} \doteq R_{t+1}+\gamma R_{t+2}+\cdots+\gamma^{n-1}R_{t+n}+\gamma^n V_{t+n-1}(S_{t+n}),\ \ n\geq 1, 0\leq t<T-n.$$

$$G_t^{(n)} \doteq G_t \text{ if } t+n\geq T.$$

- Also, since the target depends on the rewards obtained over n-steps no update will happen during the first (n-1) steps in the episode. To balance this, the same number of updates is done once the episode terminates for the last (n-1) states in the trajectory.
- Note that as n tends to ∞, n-step TD becomes MC update. As, we know MC method converges to an estimate which minimises error on the training data. This means n-step TD is closer to MC and hence error on training data for n-step TD is less than one-step TD. This is referred to as the error-reduction property which bounds the max-error in the estimate of the value function.

$$\max_s \left|\mathbb{E}_\pi\left[G_t^{(n)}\middle|S_t=s\right]-v_\pi(s)\right| \ \leq\ \gamma^n \max_s \left|V_{t+n-1}(s)-v_\pi(s)\right|$$

n-step SARSA

- You approximate target using 'n' rewards and bootstrap it with an estimate of $Q(S_{t+n}, A_{t+n})$.

$$G_t^{(n)} \doteq R_{t+1}+\gamma R_{t+2}+\cdots+\gamma^{n-1}R_{t+n}+\gamma^n Q_{t+n-1}(S_{t+n}, A_{t+n}),\ \ n\geq 1, 0\leq t<T-n$$

$$G_t^{(n)} \doteq G_t \text{ if } t+n\geq T.$$

- Update is done for $Q(S_t,A_t)$ at after 'n' steps and Q values of other (s, a) pairs remain unchanged.
- **Expected SARSA:** Instead of using $Q(S_{t+n}, A_{t+n})$ we use an expectation over $Q(S_{t+n}, a)$ for all actions 'a' (Eqn 7.6).

n-step Off-policy learning using Importance sampling

- Say we are using a behaviour policy μ (eg: some ε-greedy policy) and we estimate the target policy π.
- Importance sampling weighs the samples with the relative probability of taking that action according to policies π and μ. Here, since 'n' steps are considered, the weight depends on relative probability of those 'n' actions (Eqn 7.8).
- Using the above idea, we can obtain off-policy versions of n-step SARSA, n-step expected SARSA (Pg 158).

n-step Tree Backup algorithm

- **Explanation:** For a particular (s, a) [the next state - s', the next action - a'], the target consists of a backup using Q(s', x) where x≠a' and uses the sample (s', a') to bootstrap from later part of the episode (Pg 160).
- It is a really good way to bridge between the samples obtained (the actions taken) and the estimates obtained (for the states that were not visited in that episode).
- **Comparison with Importance sampling based methods**
  - Importance sampling based methods have high variance whereas the n-step tree backup estimate has lower variance (as full backups are considered).
  - Note that the behaviour policy μ should be properly chosen. Because: Bootstrapping depends on $\pi(A_i\,|S_i)$. If μ doesn't cover π, then these probabilities go to zero and the algorithm reduces to some k-step TD.
  - Also if the behaviour policy isn't exploratory enough, the estimates used for backup are not reliable and as a result bootstrapping doesn't work well.

n-step Q(σ)

- In all the above methods, the only thing which differs is the way in which we define the targets: whether we choose to use the sample or perform a full-backup using an expectation.
- Since these are n-step methods, there can be lot of other variants proposed as there are 'n' steps where we have a choice - whether to sample or to backup. Such possibilities are generalised in the n-step Q(σ) method.
- $\sigma_t \in [0, 1]$ - denotes the degree of sampling at time step 't'. $\sigma_t = 1$: full sampling and $\sigma_t = 0$: expectation. σ_t can be defined as a function for every (s, a) or can be randomly sampled (toss of a coin; head = 1, tail = 0)
- The n-step return accounts for both expectation and sampling depending on the value of σ_t (Eqn 7.14). It can be extended to off-policy training using importance sampling - only the sampled actions are weighted (Eqn 7.15).