Chapter 6 - Temporal Difference Learning

Atul Balaji - EE16B002

February 26, 2019

- Comparison between TD and other methods
  - DP - Full backup; System dynamics required; Bootstrapping
  - MC - Sample backup; System dynamics not required; No bootstrapping
  - TD - Sample backup and Bootstrapping; System dynamics not required

1 TD Prediction

Unlike Monte Carlo methods, Temporal Difference (TD) methods don't wait till the end of the episode for update, but only until the next time step. The simplest method is TD(0), where the update is made using the observed reward R_{t+1} and value estimate $V(S_{t+1}$ since only one time step is used:

$$V(S_t) \leftarrow V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)] \tag{1}$$

The term $\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$ is called the TD error.

2 Advantages of TD

- It is superior compared to DP, because a model of the environment is not required.
- An advantage of TD over MC is that it can be implemented in an online and incremental manner, instead of waiting until the end of episode to perform updates.
- While estimating values using a fixed policy π, the estimates of TD learning converge to v_π provided that the stochastic approximation conditions hold on the step-size α.

3 Optimality of TD(0)

- Given a finite set of episodes, the increments for a particular state s are accumulated throughout the set and finally, the overall increment is applied to $V(s)$. This is referred as the batch update and this will converge as long as α is sufficiently small. Similarly, batch-update can be done for MC methods as well.
- Using batch-update, TD(0) shows better convergence results than MC methods, but the solutions obtained are different.
- MC tries to minimize RMS error on the training data whereas TD(0) finds estimates that would be a maximum-likelihood estimate for an inherent Markov chain of the system. TD(0), using this process converges to the certainty-equivalence estimate.
- Batch TD(0) converges faster than batch MC, due to the assumption of Markov nature. On the other hand, for a non-Markov system, the learning may be more complex and hence converging to such an estimate needs a longer time.

4 SARSA: On-policy TD Control

- Generalized Policy Iteration is used for control, with TD methods used in prediction and evaluation steps.

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)] \tag{2}$$

- The above update is done for every non-terminal state S_t. Action-values for all terminal states are set to zero. This update rule involves the set $(S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1})$, lending it the name SARSA.
- The SARSA algorithm converges to the optimal policy and optimal action-value function provided all (s,a) pairs are visited infinitely often. The policy converges to the greedy policy in the limit (eg. Use an ϵ-greedy policy with a decaying ϵ).
- It is an on-policy algorithm as the update involves A_{t+1} which depends on the policy followed and hence the estimates correspond to the policy which was used to generate the episodes.

5 Q-learning: Off-policy TD Control

- In this method, the optimal action-value function (Q) is learned independent of the behaviour policy. Therefore, it is an off-policy method.
$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma max Q(S_{t+1}, \alpha) - Q(S_t, A_t)] \tag{3}$$
- From the above equation, we can observe that after a large number of updates it will becomes similar to Bellman optimality equation as the average of samples become the expectations.
- Q-learning converges to the optimal policy and the optimal action-value function under the assumption of stochastic approximation conditions on the step-size α.
- If the policy used in SARSA approaches the greedy policy in the limit, both Q-learning and SARSA converge to the same optimal policy.
- For an exploratory policy in SARSA, the obtained policies are different. SARSA results in safer optimal policies, since future exploratory actions are considered.

6 Expected SARSA

- Expected SARSA is a variant of SARSA. where the bootstrap term is replaced with an expectation.
$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[R_{t+1} + \gamma E_\pi \left[Q(S_{t+1}, A_{t+1}) | S_{t+1}\right] - Q(S_t, A_t)\right] \tag{4}$$
- It has lesser variance than SARSA (due to random selection of A_{t+1} in SARSA).
- It also minimizes the dependence on step-size α since expectation is used directly in the update rule.

7 Maximization Bias

- The control algorithms - SARSA and Q-learning both involve a maximization while constructing target policies. A max over estimated values is taken, which results in a positive bias for the estimates $Q(s, a)$, which is not present in the true values $q(s, a)$.
- This bias can be avoided by using **double learning**, where two independent estimates are maintained for each action. One estimate finds the max action and the other is used for value estimation.

8 Afterstates

- In cases where we know the initial part of the dynamics of the system, but not the full dynamics, the concept of afterstates is useful. For example, in games like chess and tic-tac-toe, the immediate effect of our moves is not known since it depends on the opponent.
- Since a state cannot be completely known in such cases, it is best to use afterstates instead.