CS 6700 Reinforcement Learning

Weekly written Assignment 5

Submitted by : CS17S008 Nitesh Methani

TD(λ) learning using linear Function Approximator

1. We are interested in learning the value-function v of an MDP, which maps each state $s \in S$ to the expected value of the return

$$v(s) = E[G_t|S_t = s]$$

When the state space becomes huge or for continuous state spaces it is not possible to represent the value function exactly. Therefore we approximate it. Here we are using Linear Approximator to represent the state-value $v(s)$.
$\hat{v}(s;\theta)$ approximates the value of s given the weight vector θ using linear approximation as follows:

$$\hat{v}(s) = \theta^T \phi(s) = \sum_i \theta_{i,t}\phi_i(s_t)$$

where $\phi_i(s_t)$ is the value of feature i of the state $s_t$and $\theta_{i,t}$ is the weight of that feature at time step t.
Now, we can do the incremental updates to our weight parameter θ like SGD update equations:

$$\theta_{t+1} = \theta_t + \alpha[U_t - \hat{v}_t(s_t)]\nabla_t \hat{v}_t(S_t)$$

where U_t is the target update which is given as:

$$U_t = R_{t+1} + \gamma \hat{v}_t(s_t + 1)$$

Note that this an offline version of update target. Online versions exist too.
The original linear TD(λ) used accumulating traces, in which e (eligibility trace) is initialized to the zero vector, and then gets updated according to the following equation:

$$e_t = \gamma\lambda e_{t-1} + \alpha\phi_t$$

where $\lambda \in [0, 1]$.
Note that for $\lambda = 0$, the TD(λ) update boils down to TD(0) update.
Given the above description of the TD(λ) with linear approximation and accumulating traces, we are now ready to write the pseudo-code.
The pseudo-code for TD(λ) using a linear function approximation(FA), is as follows :

Algorithm 1 Linear TD(λ) with accumulating traces

```
1: initialize $\theta$ arbitrarily
2: for each episode do
3:     initialize $e = 0$
4:     initialize $S$
5:     while steps in the episode and S is not terminal do
6:         generate reward $R$ and next state $S'$ for $S$
7:         $\delta \leftarrow \mathcal{R} + \gamma\theta^T\phi(S') - \theta^T\phi(S)$
8:         $e \leftarrow \gamma\lambda e + \alpha\phi(S)$
9:         $\theta \leftarrow \theta + \delta e$
10:        $S \leftarrow S'$
```

Note that in step 8 we are maintaining an eligibility trace for each parameter in the FA.

2. **Replacing Traces**
There is a variant of accumulating traces called as *replacing traces* in which the traces are reset (generally to 1) on re-visiting the state.
The update equation for replacing trace with slight modification can be written as follows:

$$e_t = \begin{cases} \gamma\lambda e_{i,t-1} & if \phi_{i,t} = 0 \\ \alpha\phi_{i,t} & if \phi_{i,t} \neq 0 \end{cases}$$

This update allow the features to take on any values, and include α in the trace also. In the pseudo-code above only 8th step will be changed, rest of the things remains same.
Replacing traces are suitable when:

- state space is discrete
- linear function approximation is done using binary features

So which one is better, accumulating or replacing trace, depends on the the state space (discrete or continuous) of the problem you are dealing with and the features of your states (whether they are binary or non-binary). So if the state space is huge and continuous and features are not binary, accumulating traces seems to be the best choice.

References

[1] TD(λ)
http://proceedings.mlr.press/v32/seijen14.pdf