Temporal Difference Learning Part 1 Notes

Zhihan Yang, Oct 26, 2019

TD learning is a combination of Monte Carlo ideas and dynamic programming (DP) ideas. Like Monte Carlo methods, TD methods can learn directly from raw experience without a model of the environment's dynamics. Like DP, TD methods update estimates based in part on other learned estimates, without waiting for a final outcome (they bootstrap).

Both TD and Monte Carlo methods use experience to solve the prediction problem. Given some experience following a policy π, both methods update their estimate V of v_π for the nonterminal states S_t occurring in that experience.

A simple every-visit Monte Carlo method suitable for nonstationary environments is $V(S_t) \leftarrow V(S_t) + \alpha[G_t - V(S_t)]$

- Where:
  - $V(S_t)$ on the right is the new value estimate for S_t
  - $V(S_t)$ on the left is the old value estimate for S_t
  - α is the stepsize $(0, 1]$
  - $[\cdot]$ is called the error term; notice how this update rule moves the old value estimate in the direction of the error
  - G_t is the cumulative reward starting from S_t and onwards
- This is because $V(S_t) = \frac{1}{N(S_t)} \sum_{n=1}^{N(S_t)} G_t^n$ can be written as the update rule $V(S_t) \leftarrow V(S_t) + \frac{1}{N}(G_t^{N(S_t)} - V(S_t))$. If we consider nonstationary environments (values of states are changing) are we want to give more emphasis to recent G_t's, setting a stepsize α greater than $\frac{1}{N}$ is the way to go.
- Also called constant-alpha MC
- Wait until the end of the episode to determine the increment to $V(S_t)$; this is the definition of G_t

---

Let's think about the motivation behind this algorithm.

- This algorithm serves to estimate $v_\pi(s)$ by sampling G_t because
  - by definition, $v_\pi(s) = \mathbb{E}_\pi[G_t|S_t = s]$ (equation 1)
  - and we simply replace the expectation with an empirical mean; for a state s , $v_\pi(s) \approx \frac{1}{N(s)} \sum_{n=1}^{N(s)} G_n(s)$
- However, equation 1 can actually be framed in the following way (remember to add the discounting term)
  - $v_\pi(s) = \mathbb{E}_\pi[G_t|S_t = s] = \mathbb{E}_\pi[r_{t+1} + G_{t+1}|S_t = s]$
  - $= \mathbb{E}_\pi[r_{t+1}|S_t = t] + \mathbb{E}_\pi[G_{t+1}|S_t = s]$
  - $= \mathbb{E}_\pi[r_{t+1}|S_t = t] + \sum_{s'} p(s'|s)\mathbb{E}_\pi[G_{t+1}|S_t = s']$

- $= \mathbb{E}_\pi[r_{t+1}|S_t = t] + \sum_{s'} p(s'|s)v_\pi(s')$
- $= \mathbb{E}_\pi[r_{t+1}|S_t = t] + \mathbb{E}_\pi[v_\pi(S_{t+1})|S_t = s]$
- $= \mathbb{E}_\pi[r_{t+1} + v_\pi(S_{t+1})|S_t = t]$

- What if, for a state $S_t = t$, instead of sampling G_t's, we sample $[r_{t+1} + V(S_{t+1})]$'s, the immediate reward plus the value estimate of the next state?

The simplest TD method makes this update

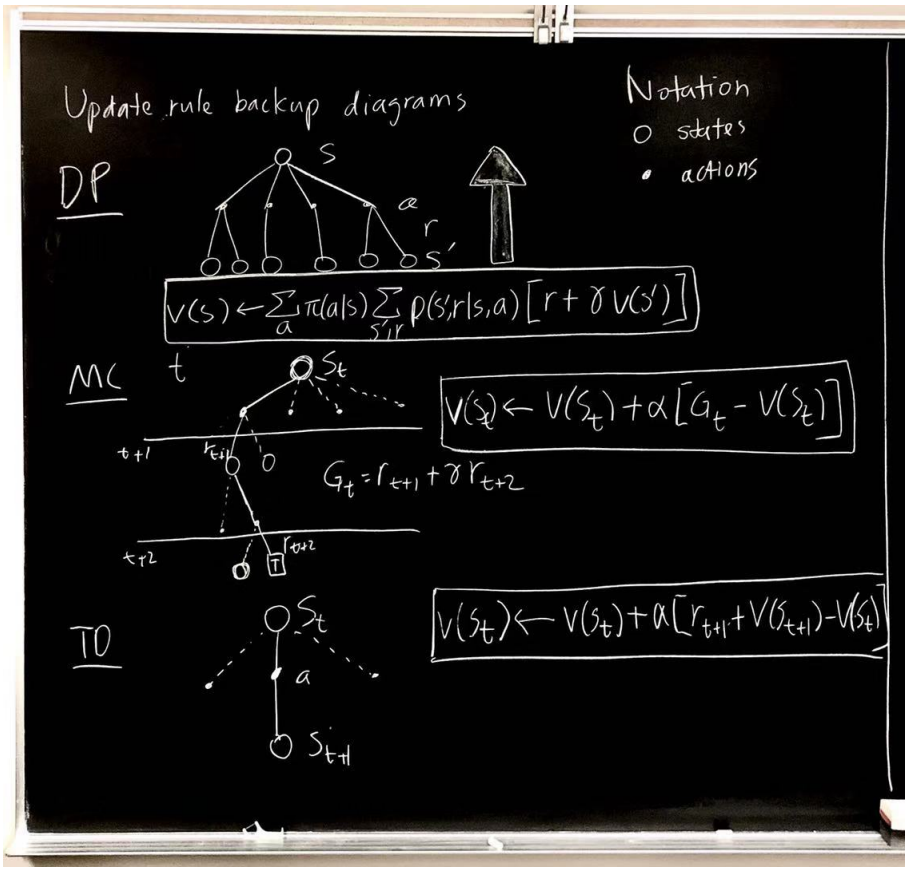
$$V(S_t) \leftarrow V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)] \qquad (1)$$

immediately on transition to S_{t+1} and receiving R_t.

Notice how G_t is replaced by something else.

This method involves bootstrapping because the estimate $V(S_t)$ is updated by sampling a quantity that involves another estimate $V(S_{t+1})$. So a natural question is, why does this move $V(S_t)$ closer to $v_\pi(S_t)$? A simple answer is that each step of the update rule includes R_{t+1} as something from real dynamic of the underlying MDP (markov decision process).

Compare the backup diagrams of Dynamic Programming, Monte Carlo Method and Temporal Difference Method



An example of TD

Read highlighted parts off textbook

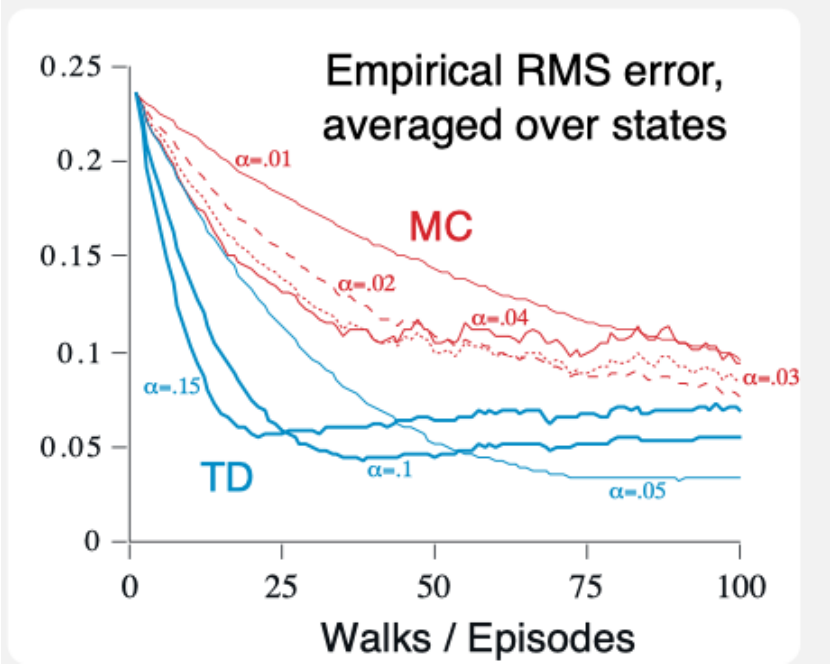
Advantages of TD Prediction Methods

- Essentially, what is the advantage from boostrapping, that is, learn a guess from a guess?

- Advantage over DP methods
  - they do not require a model of the environment, of its reward and next-state probability distributions.
- advantage of TD methods over Monte Carlo methods
  - naturally implemented in an online, fully incremental fashion
    - With Monte Carlo methods one must wait until the end of an episode, because only then is the return known, whereas with TD methods one need wait only one time step.
    - Some applications have very long episodes, so that delaying all learning until the end of the episode is too slow. Other applications are continuing tasks and have no episodes at all.
- Are TD methods sound? Can we guarantee convergence to the correct answer?
  - Yes.
  - For any fixed policy, TD(0) has been proved to converge to v pi , in the mean for a constant step-size parameter if it is su!ciently small, and with probability 1 if the step-size parameter decreases according to the usual stochastic approximation conditions (2.7), for example $1/n$.
- If both TD and Monte Carlo methods converge asymptotically to the correct predictions, then a natural next question is "Which gets there first?"
  - In fact, it is not even clear what is the most appropriate formal way to phrase this question!
  - In practice, however, TD methods have usually been found to converge faster than constant-alpha MC methods on stochastic tasks, as illustrated in the following example.

In this example we empirically compare the prediction abilities of TD(0) and constant-α MC when applied to the following Markov reward process:

- Markov reward process, consider P(S', r|S), not including actions.
- All episodes start in the center state, C, then proceed either left or right by one state on each step, with equal probability.
- Recall that the value of a state is its expected reward into the future, until terminating states, which the agent gets stuck adn receives 0 reward onwards.
- The probability of reaching the rightmost terminating block is 1/6 from A; 2/6 from B, 3/6 from C, 4/6 from D, and 5/6 from E.
- Since reaching the rightmost terminating block gives a reward of 1, the corresponding true values are 1/6, ..., 5/6.
- After 100 iterations


Empirical RMS error,
averaged over states
0.25
0.2
0.15
0.1
0.05
0
α=.01
MC
α=.02
α=.04
α=.03
α=.15
TD
α=.1
α=.05
0
25
50
75
100
Walks / Episodes