Temporal-Difference Learning

- central & novel to RL
- MC combo combo with DP
+ TD can learn from experience w/o dynamics like MC
+ TD can update estimates from other estimates w/o final outcome (bootstrap) like DP
* relationship between all is throughout rest of book
- Once again, focus on evaluation or prediction problem first, then iteration or control problem of finding optimal policy
+ find v_π off π, then new π of v_π

$$\begin{tikzcd} v_\pi \arrow[r, bend left, "1"] & \pi \arrow[l, bend left, "2"] \end{tikzcd}$$ GPI

TD Prediction

- MC basic form

$$V(S_t) \leftarrow V(S_t) + \alpha\left[G_t - V(S_t)\right]$$

+ $\boxed{\text{Constant } \alpha \text{ MC}}$ (step-size param)

- MC waits till end of ep because it must know full full return for G_t
- TD only needs R_{t+1}, simplest form:

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

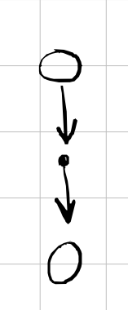
- MC update target G_t | TD update target $R_{t+1} + \gamma V(S_{t+1})$

- Called $\boxed{TD(0)}$ / $\boxed{\text{One-step TD}}$

- bootstraps on estimate
- remember

$$v_\pi(s) \doteq \mathbb{E}_\pi\left[G_t \mid S_t\right] \quad ①$$
$$\doteq \mathbb{E}_\pi\left[R_{t+1} + \gamma G_{t+1} \mid S_t\right] \quad ②$$
$$\doteq \mathbb{E}_\pi\left[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t\right] \quad ③$$

- MC uses estimate of ①, DP uses estimate of ③
+ MC uses sample return, hence estimate
+ DP uses $V(S_{t+1})$, because $v_\pi(S_{t+1})$ is not known.
* "estimate" difference
- TD does both, sample expected return & uses estimate V, not v_π

- 
backup diagram for sample updates

TD(0) - use reward & value of next state (or state-action) to update current state

- sample not expected because we don't look at the whole distribution of possible successors like in DP.
- $\boxed{\text{TD error}}$

$$\delta_t \doteq R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$$

between current est. S_t and better one $R + \gamma V(S_{t+1})$

+ it error at t available at $t+1$
* Suppose V isn't updated in the ep, like in MC methods, then error looks like:

$$G_t - V(S_t) = R_{t+1} + \gamma G_{t+1} - V(S_t) + \gamma V(S_{t+1}) - \gamma V(S_{t+1})$$
$$= \delta_t + \gamma\left(G_{t+1} - V(S_{t+1})\right)$$
$$= \delta_t + \gamma\left(\delta_{t+1} + \gamma\left(G_{t+2} - V(S_{t+2})\right)\right)$$
$$= \sum_{k=t}^{T-1} \gamma^{k-t} \delta_t$$

EXERCISE 6.1 What if V changed by a small amount?

Let $\Delta V_t(S_t) = V_{t+1}(S_t) - V_t(S_t)$ ①

$$G_t - V_t(S_t) = R_{t+1} + \gamma G_{t+1} - V_t(S_t)$$
$$= R_{t+1} + \gamma G_{t+1} - V_t(S_t) + \gamma V_t(S_{t+1}) - \gamma V_t(S_{t+1})$$
$$= \delta_t + \gamma\left(G_{t+1} - V_t(S_{t+1})\right)$$

sub ①

$$= \delta_t + \gamma\left(G_{t+1} - V_{t+1}(S_{t+1}) - \Delta V_t(S_{t+1})\right)$$
$$= \delta_t + \gamma\left(G_{t+1} - V_{t+1}(S_{t+1})\right) - \gamma \Delta V_t(S_{t+1})$$
$$= \delta_t + \gamma\left(\delta_{t+1} + \gamma\left(G_{t+2} - V_{t+2}(S_{t+2})\right) - \gamma \Delta V_{t+1}(S_{t+2})\right) - \gamma \Delta V_t(S_{t+1})$$
$$= \sum_{k=t}^{T-t} \gamma^{k-t} \delta_t + \boxed{\sum_{k=t}^{T-t} \gamma^{k-t+1} \Delta V_k(S_{k+1})}$$

Advantages of TD Predictive Models

- develop intuition
- TD learns online, mid activity
+ some applications are continuous, have no episodes
+ MC must discount wrong experimentation, slowing learning
- TD methods converge
- Open question, which is faster? MC or TD?
+ in practice TD seems faster, but no formal proof
× question itself is questionable

EXERCISE 6.3

$$V(A) = V(A) + \alpha\left[R_T + \gamma V(T) - V(A)\right]$$
$$= 0.5 + 0.1\left[0 + \quad - 0.5\right]$$
$$= 0.5 - 0.05 = \boxed{0.45}$$

Rest of states $V(S) = 0.5$, so updates to $V(S) = 0$ except @ terminal state, where value $= 0$

EXERCISE 6.4

I don't think any larger or different values would help. It would only overshoot the true value so the error term would bring it back down. Most of these algos have to adjust the step size down over time to avoid that problem

Optimality of TD(0)

- $\boxed{\text{batch updating}}$ → compute increments, but don't update V after all increments are computed
+ keep reprocessing same experience data till convergence
- in TD(0) converges to answer regardless of α (iff small)
+ in MC, converges to diff answer

EXAMPLE 6.3: Random Walk
- batch updates of TD(0), MC
- MC reaches a V that is optimal for its experience
- TD(0) outperforms

+ MC is optimal in limited way, TD optimal more relevant to prediction

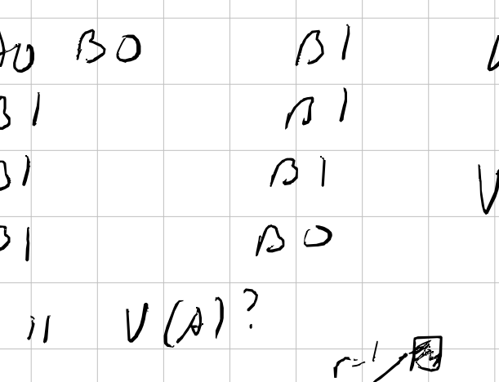
EXAMPLE 6.4: 8 eps

A0 B0   B1
B1   B1
B1   B1
B1   B0

What is $V(B)$?

$V(B) = 3/4$

What is $V(A)$?



Markov process $V(A) = 3/4$

However! MC attempt to reduce RMS on training data will est. $V(A) = 0$! TD(0) will update A on every visit from

- MC minimizes error on seen data
- TD(0) estimates value for maximum likelihood Markov Mode
+ certainty-equivalence estimate because it assumes that the est. of underlying process was known for certain
× TD(0) converges to ↑
* non batch does not converge the way batch TD does.
- almost always infeasible to compute certainty equivalence est due to space and time needs in states.

EXERCISE 6.7 off policy TD(0) with π / b & $\rho_{t:t}$

Incremental Update becomes:

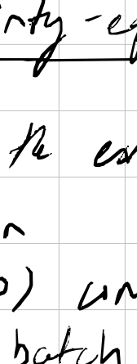
$$V(S_t) = V(S_t) + \alpha \rho_{t:t}\left[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\right]$$

Sarsa: On-Policy TD Control

- explore vs exploit in control, again.
- first, use state-action values:

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

- $Q(S_{t+1}, A_{t+1}) = 0$ if $t+1 = T$ (terminal)
- quintuple in use: $(S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1})$


SARSA

- convergence depends on policy's relationship with Q
+ ε-soft or ε-greedy, with $\varepsilon = 1/t$
× state-action pairs must be visited infinitely
× policy must converge to limit of greedy policy

SARSA (on policy TD control) for $Q \approx q_*$

```
α ∈ (0,1]
ε > 0 (small)
Q(s,a) ∀ s ∈ S⁺, a ∈ A(s) arbitrarily
    except Q(terminal, ·) = 0

for each ep
    init S
    choose A from S based on Q (ε-greedy)
    for each step in ep:
        R, S' ← step(A)
        choose A' from S'
        Q(S,A) = Q(S,A) + α[R + γ Q(S',A') - Q(S,A)]
        S = S'
        A = A'
    until termination
```

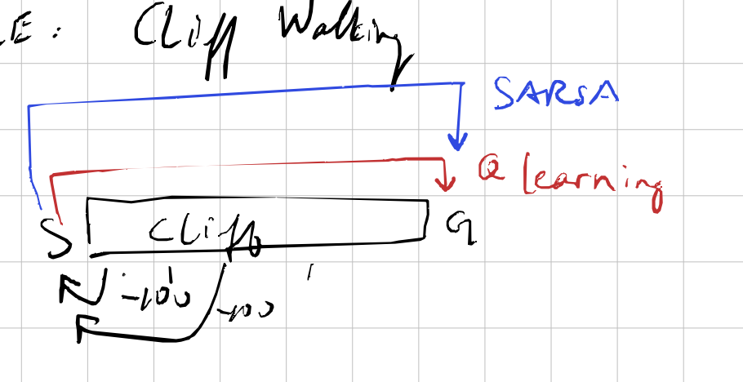
- important to remember that if a policy chosen for MC gets stuck, ep never terms, MC never updates Q
+ SARSA online doesn't suffer, will pick a new policy

Q-learning: Off-Policy TD Control

$$Q(S_t, A_t) = Q(S_t, A_t) + \alpha\left[R_{t+1} + \gamma \max_a Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)\right]$$

- Q directly approximates q_*, regardless of policy
+ makes it "off policy" on update

EXAMPLE: Cliff Walking



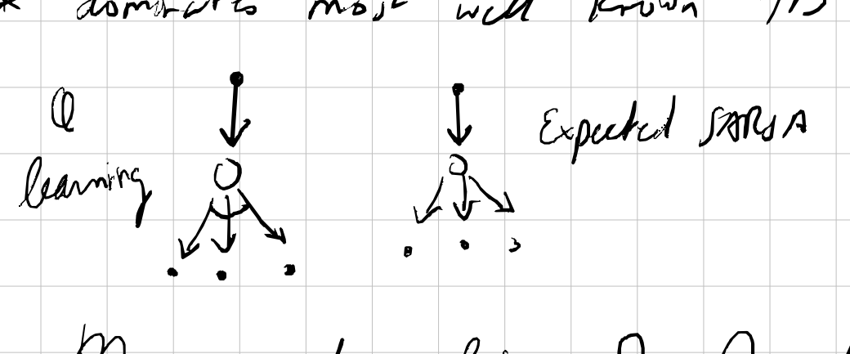
- SARSA takes action into account, so stays away from cliff
- Q-learning will learn optimal faster, but will die a lot because it doesn't care about action taken

EXPECTED SARSA

- what if you liked Q learning, but also considered the likelihood of each action, not just the max value
+ hence, obtain expected value

$$Q(S_t, A_t) = Q(S_t, A_t) + \alpha\left[R_{t+1} + \gamma \mathbb{E}_\pi\left[Q(S_{t+1}, A_{t+1}) \mid S_{t+1}\right] - Q(S_t, A_t)\right]$$
$$= Q(S_t, A_t) + \alpha\left[R_{t+1} + \gamma \sum_a \pi(a \mid S_{t+1}) Q(S_{t+1}, a) - Q(S_t, A_t)\right]$$

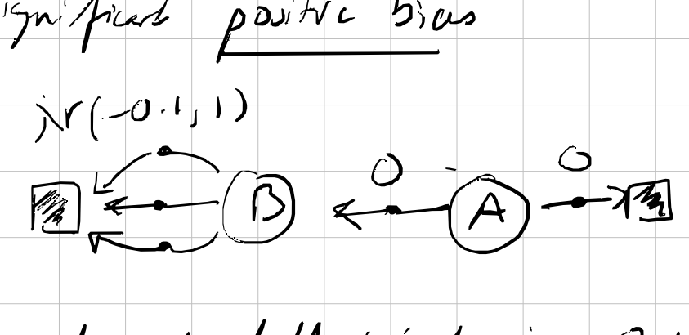
- deterministic movement as SARSA in expectation
+ $\boxed{\text{Expected SARSA}}$
- outperforms SARSA, does better online than Q-learning
+ if done with off-policy strategy, it is the same as Q-learning
× Expected SARSA generalizes Q-learning
* dominates most well known TD methods



Maximization Bias & Double Learning

- current algos use maximum of estimate for true value
- leads to significant positive bias

EXAMPLE



- Expected value of left trajectory is -0.1 right 0
- left always worse
- current control prefers left because of maximization bias
+ positive values over attract

- fix by viewing problem as split between estimating value and picking best action
+ don't use the same samples for both
+ $Q_1, Q_2 \approx q_*$
+ $A^* = \arg\max_a Q_1(a)$ use Q_1 to pick
+ $Q_2(A^*) = Q_2(\arg\max_a Q_1(a))$ now use Q_2 to estimate value
* then flip flop every timestep to evenly distribute bias
* $\boxed{\text{Double Q-learning}}$

$$Q_1(S_t, A_t) \mathrel{+}= \alpha\left[R_{t+1} + \gamma Q_2\left(S_{t+1}, \arg\max_a Q_1(S_{t+1}, a)\right) - Q_1(S_t, A_t)\right]$$

* policy can pick action A using $Q_1 + Q_2$
- double learning of SARSA and Expected SARSA exist too

EXERCISE 6.13 Double Expected SARSA??

$$Q_1(S_t, A_t) \mathrel{+}= \alpha\left[R_{t+1} + \gamma \sum_a \pi(a \mid S_{t+1}) Q_2(S_{t+1}, a) - Q_1(S_t, A_t)\right]$$

Games, Afterstates, & Other Special Cases

- what happens when different state actions lead to the same after state?
+ games often have this ability.
+ same state must have same value
× speeds up learning
- occur in many tasks impossible to speak to all the specializations
+ can still interleave afterstate value function and policy iteration in GPI

Summary

- These TD methods are simple & powerful
- They are one step, tabular, model free
+ Soon, extend to n-step (MC link) & model forms (planning & DP)
× after, function approximation instead of tabular (neural nets)