RL-Week-6

4 N-Step Predic^n & TD-lambda

TD(0)

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

only next step reward,

$$G_t^{(1)} = R_{t+1} + \gamma V_t(S_{t+1})$$

1 step truncated corrected returns

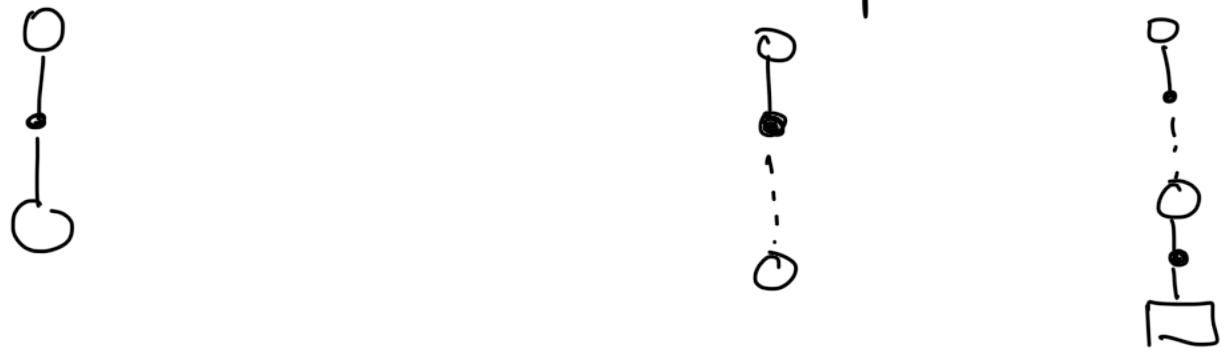
Obv. n step

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

similarly $G_t^{(n)}$ → consider rewards till the end of the episode (MC)

beginning the value func^n & returns could be wrong, it gets fine later by discounting under eventually.

TD(1 step) — ... — n step — ... — MC



- Instead of n-step, let's consider the average of n-step returns.

$$G_t^{avg.} = \frac{1}{2} G_t^{(2)} + \frac{1}{2} G_t^{(4)} \quad \text{eg.}$$

or 0.8 " + 0.2 "

This is TD(λ). The average contains all the n-step backups each weighted α to λ^{n-1}, $0 \leq \lambda \leq 1$

λ - return ⇒ $G_t^{\lambda} = (1-\lambda) \sum_{n=1}^{\infty} \lambda^{n-1} G_t^{(n)}$

↓ it combines 1-step to MC.

↓ $1-\lambda$

→ λ^{T-t-1} ← length of episode

→ λ - returns

$$G_t^{(\lambda)} \propto \sum_{n=1}^{\infty} \lambda^{n-1} G_t^{(n)}$$

$$G_t^{(\lambda)} = G_t \lambda^{T-t-1}$$

so far we saw the forward view of TD(λ)

- Eligibility Traces (Backward view)
  ↳ helps to implement TD(λ)

These are vars. related to each state given by $e_t(s)$. It is indicated the degree to which state is eligible for

learning changes.

$$e_t(s) = \begin{cases} \gamma\lambda e_{t-1}(s) & s \neq s_t \\ \gamma\lambda e_{t-1}(s) + 1 & s = s_t \end{cases}$$

$\forall s \in S$, γ is discount if eligible rate.

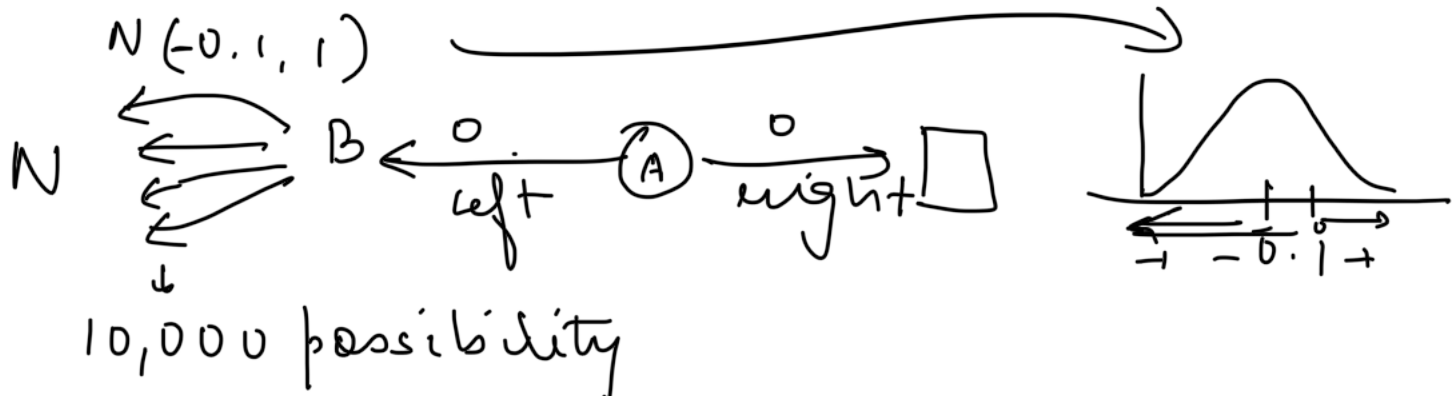
we accumulate the eligible traces where $s = s_t$.

Initially, $e(s) = 0 \rightsquigarrow$ eligibility is initialized.

- both views accumulate the same updates. They are equivalent, bit a hard show.
- some eg.

L2 Double Q-learning

- we are actually estimating q func^n. The target uses the current max. So, therefore, it will propagate errors.
  ↳ maximisa^n bias.



where to move from A

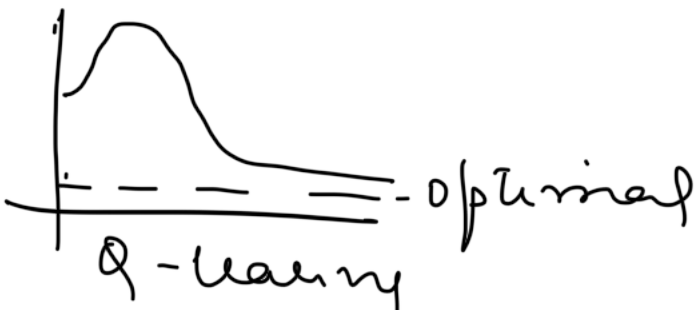
$E[G_t | s_0 = A, a_0 = \text{right}] = 0$

$E[G_t | s_0 = A, a_0 = \text{left}] = 0 + (-0.1) = -0.1$

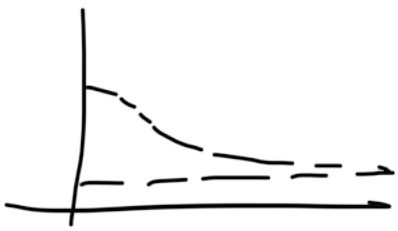
when we use Q-learning.

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s, a) \right]$$

few acns have + value, then it cause to go for left because of max, so it always overestimates. leads to +ve bias, called maximizan bias. All approaches with max provides this.



but we want



- the problem is using the same samples both to determine the max. acn & to estimate the value.

↳ Solun → use diff. estimates for maximizing the acn & estimate the value.

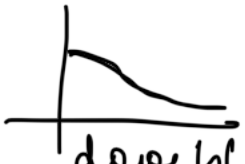
with 1/2 probab.

$$Q_1(s,A) \leftarrow Q_1(s,A) + \alpha\left(R + \gamma Q_2\left(s', \operatorname{argmax}_a Q_1(s',a)\right) - Q_1(s,a)\right)$$

similarly $Q_2(s,A) \propto \gamma Q_1$

terminate $\Rightarrow Q_1 \approx Q_2 \approx q_*$

it reduces maximization bias and not removes it.



double q-learn.