Temporal Difference Methods

CS60077: Reinforcement Learning

Abir Das

IIT Kharagpur

Sept 24, 30, Oct 01, 07, 2021

Agenda

- Understand incremental computation of Monte Carlo methods
- From incremental Monte Carlo methods, the journey will take us to different Temporal Difference (TD) based methods.

Resources

- § Reinforcement Learning by Udacity [Link]
- § Reinforcement Learning by Balaraman Ravindran [Link]
- § Reinforcement Learning by David Silver [Link]
- § SB: Chapter 6

MRP Evaluation - Model Based

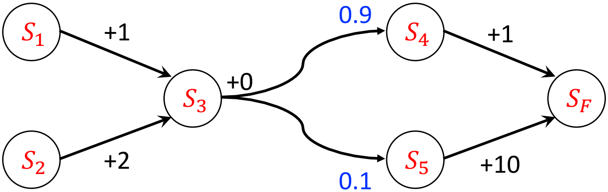
- § Like the previous approaches, here also we are going to first look at the evaluation problems using TD methods and then later, we will do TD control.

MRP Evaluation - Model Based

- Like the previous approaches, here also we are going to first look at the evaluation problems using TD methods and then later, we will do TD control.
- Let us take a MRP. Why MRP?

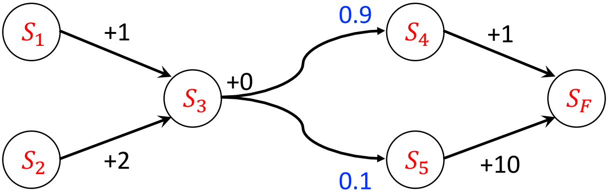
MRP Evaluation - Model Based

- Like the previous approaches, here also we are going to first look at the evaluation problems using TD methods and then later, we will do TD control.
- Let us take a MRP. Why MRP?



MRP Evaluation - Model Based

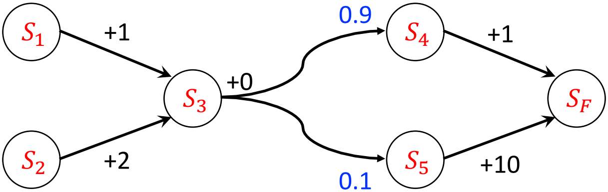
- Like the previous approaches, here also we are going to first look at the evaluation problems using TD methods and then later, we will do TD control.
- Let us take a MRP. Why MRP?



- Find $V(S_3)$, given $\gamma = 1$

MRP Evaluation - Model Based

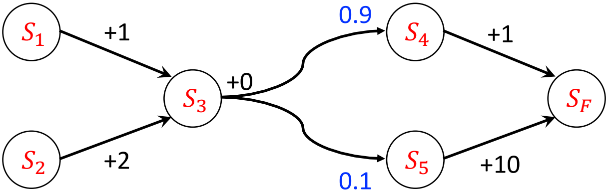
- Like the previous approaches, here also we are going to first look at the evaluation problems using TD methods and then later, we will do TD control.
- Let us take a MRP. Why MRP?



- Find $V(S_3)$, given $\gamma = 1$
- $V(S_F) = 0$

MRP Evaluation - Model Based

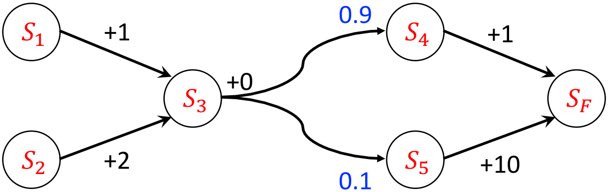
- Like the previous approaches, here also we are going to first look at the evaluation problems using TD methods and then later, we will do TD control.
- Let us take a MRP. Why MRP?



- Find $V(S_3)$, given $\gamma = 1$
- $V(S_F) = 0$
- Then $V(S_4) = 1 + 1 \times 0 = 1, V(S_5) = 10 + 1 \times 0 = 10$

MRP Evaluation - Model Based

- Like the previous approaches, here also we are going to first look at the evaluation problems using TD methods and then later, we will do TD control.
- Let us take a MRP. Why MRP?



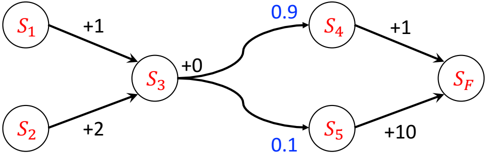
- Find $V(S_3)$, given $\gamma = 1$
- $V(S_F) = 0$
- Then $V(S_4) = 1 + 1 \times 0 = 1, V(S_5) = 10 + 1 \times 0 = 10$
- Then $V(S_3) = 0 + 1 \times (0.9 \times 1 + 0.1 \times 10) = 1.9$

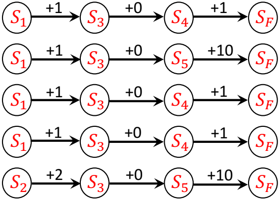
MRP Evaluation - Monte Carlo

§ Now let us think about how to get the values from 'experience' without knowing the model.

MRP Evaluation - Monte Carlo

- Now let us think about how to get the values from ‘experience’ without knowing the model.
- Let’s say we have the following samples/episodes.

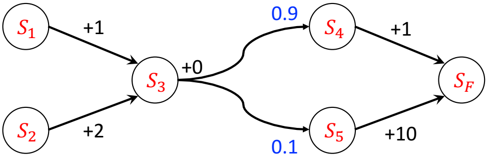


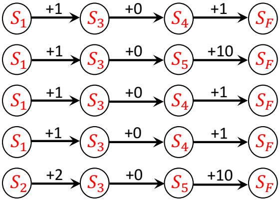


MRP Evaluation - Monte Carlo

§ Now let us think about how to get the values from ‘experience’ without knowing the model.

§ Let’s say we have the following samples/episodes.

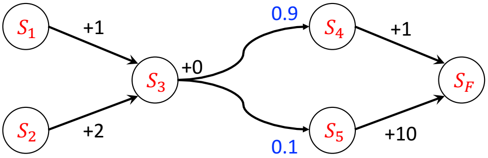


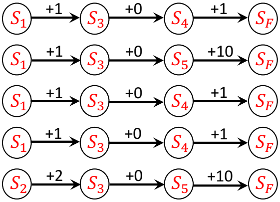


§ What is the estimated value of $V(S_1)$ - after 3 epiodes? after 4 episodes?

MRP Evaluation - Monte Carlo

- Now let us think about how to get the values from 'experience' without knowing the model.
- Let's say we have the following samples/episodes.





- What is the estimated value of $V(S_1)$ - after 3 epiodes? after 4 episodes?
- After 3 episodes: $\frac{(1+0+1)+(1+0+10)+(1+0+1)}{3} = 5.0$
- After 4 episodes: $\frac{(1+0+1)+(1+0+10)+(1+0+1)+(1+0+1)}{4} = 4.25$

Incremental Monte Carlo

§ Next we are going to see how we can 'incrementally' compute an estimate for the value of a state given the previous estimate, *i.e.*, given the estimate after 3 episodes, how do we get that after 4 episodes and so on.

Incremental Monte Carlo

- Next we are going to see how we can 'incrementally' compute an estimate for the value of a state given the previous estimate, *i.e.*, given the estimate after 3 episodes, how do we get that after 4 episodes and so on.
- Let $V_{T-1}(S_1)$ is the estimate of the value function at state S_1 after $(T-1)^{th}$ episode.
- Let the return (or total discounted reward) of the T^{th} episode be $G_T(S_1)$
- Then,

$$\begin{aligned} V_T(S_1) &= \frac{V_{T-1}(S_1) * (T-1) + G_T(S_1)}{T} \\ &= \frac{T-1}{T} V_{T-1}(S_1) + \frac{1}{T} G_T(S_1) \\ &= V_{T-1}(S_1) + \alpha_T \left(G_T(S_1) - V_{T-1}(S_1) \right), \quad \alpha_T = \frac{1}{T} \end{aligned}$$

Incremental Monte Carlo

$$V_T(S_1) = V_{T-1}(S_1) + \alpha_T \left(G_T(S_1) - V_{T-1}(S_1)\right), \quad \alpha_T = \frac{1}{T}$$

- Think of T as time *i.e.*, you are drawing sampling trajectories and getting the $(T-1)^{th}$ episode at time $(T-1)$, T^{th} episode at time T and so on.
- Then we are looking at a 'Temporal difference'. The 'update' to the value of S_1 is going to be equal to the difference between the return $(G_T(S_1))$ at step T and the estimate $(V_{T-1}(S_1))$ at the previous time step $T-1$
- As we get more and more episodes, the learning rate α_T, gets smaller and smaller. So we make smaller and smaller changes.

Properties of Learning Rate

§ This learning falls under a general learning rule where the value at time T = the value at time $T-1$ + some learning rate*(difference between what you get and what you expected it to be)

$$V_T(S_1) = V_{T-1}(S_1) + \alpha_T\left(G_T(S_1) - V_{T-1}(S_1)\right)$$

Properties of Learning Rate

- This learning falls under a general learning rule where the value at time T = the value at time $T-1$ + some learning rate*(difference between what you get and what you expected it to be)

$$V_T(S_1) = V_{T-1}(S_1) + \alpha_T \left(G_T(S_1) - V_{T-1}(S_1)\right)$$

- In limit, the estimate is going to converge to the true value, *i.e.*, $\lim_{T\rightarrow\infty}(S) = V(S)$, given two conditions that the learning rate sequence has to obey.
  I. $\sum_T \alpha_T = \infty$
  II. $\sum_T \alpha_T^2 < \infty$

Properties of Learning Rate

§ Let us see what $\sum_{T=1}^{\infty} \frac{1}{T}$ is.

Properties of Learning Rate

§ Let us see what $\sum_{T=1}^{\infty} \frac{1}{T}$ is.

§ It is $1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \cdots$ What is it known as?

Properties of Learning Rate

§ Let us see what $\sum_{T=1}^{\infty} \frac{1}{T}$ is.

§ It is $1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \cdots$ What is it known as? Harmonic series.

Properties of Learning Rate

- § Let us see what $\sum_{T=1}^{\infty} \frac{1}{T}$ is.
- § It is $1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \cdots$ What is it known as? Harmonic series.
- § Does it converge?

Properties of Learning Rate

§ Let us see what $\sum_{T=1}^{\infty} \frac{1}{T}$ is.

§ It is $1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \cdots$ What is it known as? Harmonic series.

§ Does it converge? No.

$$
\begin{aligned}
&1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5} + \frac{1}{6} + \frac{1}{7} + \frac{1}{8} + \frac{1}{9} + \cdots \\
>&1 + \frac{1}{2} + \underbrace{\frac{1}{4} + \frac{1}{4}}_{\frac{1}{2}} + \underbrace{\frac{1}{8} + \frac{1}{8} + \frac{1}{8} + \frac{1}{8}}_{\frac{1}{2}} + \frac{1}{16} + \cdots \\
=&1 + \frac{1}{2} + \frac{1}{2} + \frac{1}{2} + \cdots = \infty
\end{aligned}
$$

Properties of Learning Rate

- A generalization of the harmonic series is the p-series (or hyperharmonic series), defined as $\sum_{n=1}^{\infty} \frac{1}{n^p}$, for any +ve real number p.
- p-series converges for all $p > 1$ (in which case, it is called the over-harmonic series) and diverges for all $p \leq 1$.
- So, according to these rules, lets see if the following α_T's result in a converging algorithm.

α_T	$\sum \alpha_T$	$\sum \alpha_T^2$	Algo Converges
$\frac{1}{T^2}$			
$\frac{1}{T}$			
$\frac{1}{T^{\frac{2}{3}}}$			
$\frac{1}{T^{\frac{1}{2}}}$			

Properties of Learning Rate

- A generalization of the harmonic series is the p-series (or hyperharmonic series), defined as $\sum_{n=1}^{\infty}\frac{1}{n^p}$, for any +ve real number p.
- p-series converges for all $p > 1$ (in which case, it is called the over-harmonic series) and diverges for all $p \leq 1$.
- So, according to these rules, lets see if the following α_T's result in a converging algorithm.

α_T	$\sum \alpha_T$	$\sum \alpha_T^2$	Algo Converges
$\frac{1}{T^2}$	$< \infty$	$< \infty$	No
$\frac{1}{T}$			
$\frac{1}{T^{\frac{2}{3}}}$			
$\frac{1}{T^{\frac{1}{2}}}$			

Properties of Learning Rate

- A generalization of the harmonic series is the p-series (or hyperharmonic series), defined as $\sum_{n=1}^{\infty} \frac{1}{n^p}$, for any +ve real number p.
- p-series converges for all $p > 1$ (in which case, it is called the over-harmonic series) and diverges for all $p \leq 1$.
- So, according to these rules, lets see if the following α_T's result in a converging algorithm.

α_T	$\sum \alpha_T$	$\sum \alpha_T^2$	Algo Converges
$\frac{1}{T^2}$	$< \infty$	$< \infty$	No
$\frac{1}{T}$	∞	$< \infty$	Yes
$\frac{1}{T^{\frac{2}{3}}}$			
$\frac{1}{T^{\frac{1}{2}}}$			

Properties of Learning Rate

- A generalization of the harmonic series is the p-series (or hyperharmonic series), defined as $\sum_{n=1}^{\infty} \frac{1}{n^p}$, for any +ve real number p.
- p-series converges for all $p > 1$ (in which case, it is called the over-harmonic series) and diverges for all $p \leq 1$.
- So, according to these rules, lets see if the following α_T's result in a converging algorithm.

α_T	$\sum \alpha_T$	$\sum \alpha_T^2$	Algo Converges
$\frac{1}{T^2}$	$< \infty$	$< \infty$	No
$\frac{1}{T}$	∞	$< \infty$	Yes
$\frac{1}{T^{\frac{2}{3}}}$	∞	$< \infty$	Yes
$\frac{1}{T^{\frac{1}{2}}}$			

Properties of Learning Rate

- A generalization of the harmonic series is the p-series (or hyperharmonic series), defined as $\sum_{n=1}^{\infty} \frac{1}{n^p}$, for any +ve real number p.
- p-series converges for all $p > 1$ (in which case, it is called the over-harmonic series) and diverges for all $p \leq 1$.
- So, according to these rules, lets see if the following α_T's result in a converging algorithm.

α_T	$\sum \alpha_T$	$\sum \alpha_T^2$	Algo Converges
$\frac{1}{T^2}$	$< \infty$	$< \infty$	No
$\frac{1}{T}$	∞	$< \infty$	Yes
$\frac{1}{T^{\frac{2}{3}}}$	∞	$< \infty$	Yes
$\frac{1}{T^{\frac{1}{2}}}$	∞	∞	No

TD(1)

Algorithm 1: TD(1)

initialization: Episode No. $T \leftarrow 1$;

repeat

  foreach $s \in \mathcal{S}$ **do**

    initialize $e(s) = 0$ // $e(s)$ is called 'eligibility' of state s.

    $V_T(s) = V_{(T-1)}(s)$// same as the previous episode.

  $t \leftarrow 1$;

  repeat

    After state transition, $s_{t-1} \xrightarrow{R_t} s_t$

      $e(s_{t-1}) = e(s_{t-1}) + 1$// updating state eligibility.

    foreach $s \in \mathcal{S}$ **do**

      $V_T(s) \leftarrow V_{T-1}(s) + \alpha_T \left(R_t + \gamma V_{T-1}(s_t) - V_{T-1}(s_{t-1})\right) e(s)$;

      $e(s) = \gamma e(s)$

    $t \leftarrow t + 1$

  until *this episode terminates*;

  $T \leftarrow T + 1$

until *all episodes are done*;

TD(1) Example

§ Let us try to walk through the pseudocode with the help of a very little example.

$$S_0 \xrightarrow{R_1} S_1 \xrightarrow{R_2} S_2 \xrightarrow{R_3} S_F$$

e	0	0	0

TD(1) Example

§ Let us try to walk through the pseudocode with the help of a very little example.

$$S_0 \xrightarrow{R_1} S_1 \xrightarrow{R_2} S_2 \xrightarrow{R_3} S_F$$

e	0	0	0

§ Now as a result of transition from s_0 to s_1 the eligibilities change as,

$$S_0 \xrightarrow{R_1} S_1 \xrightarrow{R_2} S_2 \xrightarrow{R_3} S_F$$

e	1	0	0

TD(1) Example

§ Let us try to walk through the pseudocode with the help of a very little example.

$$S_0 \xrightarrow{R_1} S_1 \xrightarrow{R_2} S_2 \xrightarrow{R_3} S_F$$

e	0	0	0

§ Now as a result of transition from s_0 to s_1 the eligibilities change as,

$$S_0 \xrightarrow{R_1} S_1 \xrightarrow{R_2} S_2 \xrightarrow{R_3} S_F$$

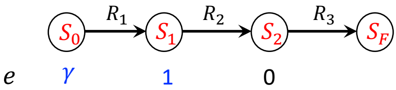
e	1	0	0

§ Now, we are going to loop through all the states and apply the TD update $\left[R_1 + \gamma V_{(T-1)}(s_1) - V_{(T-1)}(s_0)\right]$ proportional to the eligibility and the learning rate of all the states.

- $V_T(s_0) = \alpha_T \left(R_1 + \gamma V_{(T-1)}(s_1) - V_{(T-1)}(s_0)\right)$
- $V_T(s_1) = 0$
- $V_T(s_2) = 0$

TD(1) Example

§ Now transition from s_1 to s_2 happens and the eligibilities become



§ The temporal difference is $\left[R_2 + \gamma V_{(T-1)}(s_2) - V_{(T-1)}(s_1)\right]$

- $V_T(s_0) = \alpha_T \left(R_1 + \cancel{\gamma V_{(T-1)}(s_1)} - V_{(T-1)}(s_0)\right) + \gamma\alpha_T \left(R_2 + \gamma V_{(T-1)}(s_2) - \cancel{V_{(T-1)}(s_1)}\right) = \alpha_T \left(R_1 + \gamma R_2 + \gamma^2 V_{(T-1)}(s_2) - V_{(T-1)}(s_0)\right)$
- $V_T(s_1) = \alpha_T \left(R_2 + \gamma V_{(T-1)}(s_2) - V_{(T-1)}(s_1)\right)$
- $V_T(s_2) = 0$

TD(1) Example

§ Now transition from s_2 to s_F happens and the eligibilities become

$$S_0 \xrightarrow{R_1} S_1 \xrightarrow{R_2} S_2 \xrightarrow{R_3} S_F$$

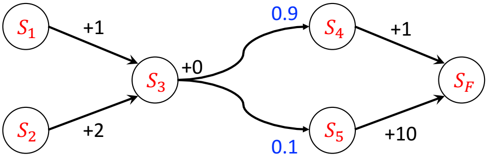
$$e \quad \gamma^2 \quad \gamma \quad 1$$

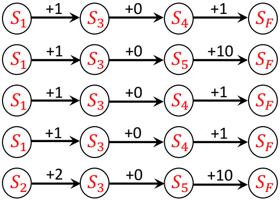
§ The temporal difference is $\left[R_3 + \gamma V_{(T-1)}(s_F) - V_{(T-1)}(s_2)\right]$

- $V_T(s_0) = \alpha_T\Big(R_1 + \gamma R_2 + \cancel{\gamma^2 V_{(T-1)}(s_2)} - V_{(T-1)}(s_0)\Big) + \alpha_T\gamma^2\left(R_3 + \gamma V_{(T-1)}(s_F) - \cancel{V_{(T-1)}(s_2)}\right) = \alpha_T\left(R_1 + \gamma R_2 + \gamma^2 R_3 + \gamma^3 V_{(T-1)}(s_F) - V_{(T-1)}(s_0)\right)$
- $V_T(s_1) = \alpha_T\left(R_2 + \cancel{\gamma V_{(T-1)}(s_2)} - V_{(T-1)}(s_1)\right) + \alpha_T\gamma\left(R_3 + \gamma V_{(T-1)}(s_F) - \cancel{V_{(T-1)}(s_2)}\right) = \alpha_T\left(R_2 + \gamma R_3 + \gamma^2 V_{(T-1)}(s_F) - V_{(T-1)}(s_1)\right)$
- $V_T(s_2) = \alpha_T\left(R_3 + \gamma V_{(T-1)}(s_F) - V_{(T-1)}(s_2)\right)$
- So, some pattern is emerging!!

TD(1) Example

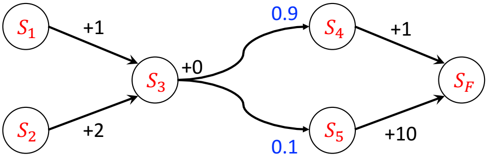
§ Let us try to apply TD(1) to our starting MRP.

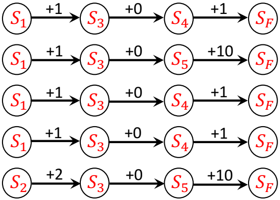




TD(1) Example

§ Let us try to apply TD(1) to our starting MRP.



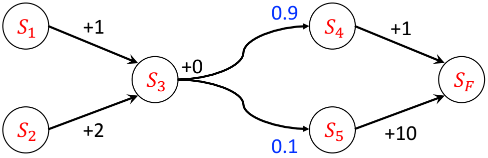


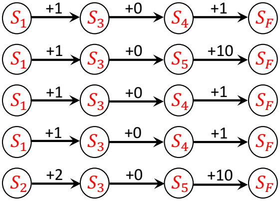
§ s_2 is seen only once. So, $V(s_2)$ will be computed for this episode only. $V(s_2) = \alpha_t \left(2 + \gamma * 0 + \gamma^2 * 10 + \gamma^3 * \cancelto{0}{V(s_F)} - \cancelto{0}{V(s_2)}\right) = 1 * 12 = 12$

§ γ is taken to be 1 for easy computation.

TD(1) Example

- What is the maximum likelihood estimate?

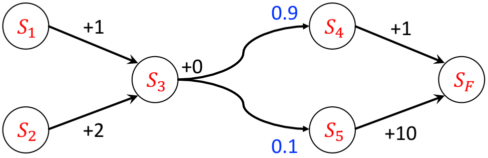


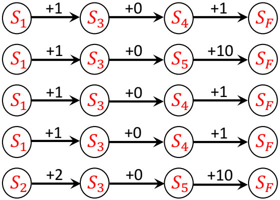


- Estimated state transition probabilities:
  - $s_3 \rightarrow s_4 : \frac{3}{5} = 0.6$
  - $s_3 \rightarrow s_5 : \frac{2}{5} = 0.4$

TD(1) Example

- What is the maximum likelihood estimate?

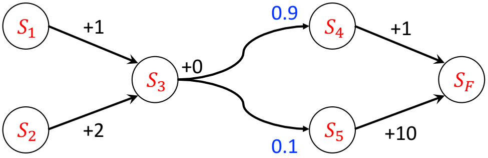


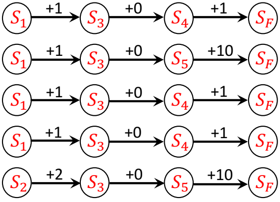


- Estimated state transition probabilities:
  - $s_3 \rightarrow s_4 : \frac{3}{5} = 0.6$
  - $s_3 \rightarrow s_5 : \frac{2}{5} = 0.4$
- So,
  - $V(S_F) = 0$
  - Then $V(S_4) = 1 + 1 \times 0 = 1, V(S_5) = 10 + 1 \times 0 = 10$
  - Then $V(S_3) = 0 + 1 \times (0.6 \times 1 + 0.4 \times 10) = 4.6$
  - and $V(S_2) = 2 + 1 \times 4.6 = 6.6$

TD(1) Example

§ What is the maximum likelihood estimate?





§ Estimated state transition probabilities:

- $s_3 \to s_4 : \frac{3}{5} = 0.6$
- $s_3 \to s_5 : \frac{2}{5} = 0.4$

§ So,

- $V(S_F) = 0$
- Then $V(S_4) = 1 + 1 \times 0 = 1, V(S_5) = 10 + 1 \times 0 = 10$
- Then $V(S_3) = 0 + 1 \times (0.6 \times 1 + 0.4 \times 10) = 4.6$
- and $V(S_2) = 2 + 1 \times 4.6 = 6.6$

§ The true value of state s_2, we found when the true transition probabilities are known, is 3.9

TD(1) Analysis

- One reason why TD(1) estimate is far off is because - we only used one of the five trajectories to propagate information. But, the maximum likelihood estimate used information from all 5 trajectories.
- So, TD(1) suffers when a rare event occurs in a run $(s_3 \rightarrow s_5 \rightarrow s_F)$. Then the estimate can be far off.
- We will try to shore up some of these issues next

TD(0)

§ Let us look at the TD(1) update rule more carefully.

$$V_T(s) \leftarrow V_{T-1}(s) + \alpha_T \left(R_t + \gamma V_{T-1}(s_t) - V_{T-1}(s_{t-1})\right) e(s)$$

§ Let us change only a few terms in the above rule.

$$V_T(s_{t-1}) \leftarrow V_{T-1}(s_{t-1}) + \alpha_T \left(R_t + \gamma V_{T-1}(s_t) - V_{T-1}(s_{t-1})\right)$$

§ What would we expect this outcome to be on average?

TD(0)

- Let us look at the TD(1) update rule more carefully.

$$V_T(s) \leftarrow V_{T-1}(s) + \alpha_T \left(R_t + \gamma V_{T-1}(s_t) - V_{T-1}(s_{t-1})\right) e(s)$$

- Let us change only a few terms in the above rule.

$$V_T(s_{t-1}) \leftarrow V_{T-1}(s_{t-1}) + \alpha_T \left(R_t + \gamma V_{T-1}(s_t) - V_{T-1}(s_{t-1})\right)$$

- What would we expect this outcome to be on average?
- The random thing here is the state s_t. We are in some state s_{t-1} and we make a transition, we don't really know where we are going to end up. There is some probability involved in that.
- So, ignoring α_T for the time being, the expected value of the above modified rule is $\mathbb{E}_{s_t}\left[R_t + \gamma V_T(s_t)\right]$, which is basically averaging after sampling different possible s_t values.
- This is what maximum likelihood is also doing.

TD(1) and TD(0)

Algorithm 2: TD(1)

initialization: Episode No. $T \leftarrow 1$;
repeat
 foreach $s \in \mathcal{S}$ **do**
  initialize $e(s) = 0$;
  $V_T(s) = V_{(T-1)}(s)$
 $t \leftarrow 1$;
 repeat
  After state transition,
  $s_{t-1} \xrightarrow{R_t} s_t$
   $e(s_{t-1}) = e(s_{t-1}) + 1$
  foreach $s \in \mathcal{S}$ **do**
   $V_T(s) \leftarrow V_{T-1}(s) + \alpha_T(R_t + \gamma V_{T-1}(s_t) - V_{T-1}(s_{t-1}))e(s)$;
   $e(s) = \gamma e(s)$
  $t \leftarrow t + 1$
 until *this episode terminates*;
 $T \leftarrow T + 1$
until *all episodes are done*;

Algorithm 3: TD(0)

initialization: Episode No. $T \leftarrow 1$;
repeat
 foreach $s \in \mathcal{S}$ **do**
  $V_T(s) = V_{(T-1)}(s)$
 $t \leftarrow 1$;
 repeat
  After $s_{t-1} \xrightarrow{R_t} s_t$
  for $s = s_{t-1}$ **do**
   $V_T(s) \leftarrow V_{T-1}(s) + \alpha_T(R_t + \gamma V_{T-1}(s_t) - V_{T-1}(s_{t-1}))$
  $t \leftarrow t + 1$
 until *this episode terminates*;
 $T \leftarrow T + 1$
until *all episodes are done*;

TD(λ)

Algorithm 4: TD(λ)

initialization: Episode No. $T \leftarrow 1$;
repeat
  foreach $s \in \mathcal{S}$ **do**
    initialize $e(s) = 0$;
    $V_T(s) = V_{(T-1)}(s)$
  $t \leftarrow 1$;
  repeat
    After $s_{t-1} \xrightarrow{R_t} s_t$
      $e(s_{t-1}) = e(s_{t-1}) + 1$;
    foreach $s \in \mathcal{S}$ **do**
      $V_T(s) \leftarrow V_{T-1}(s) + \alpha_T(R_t + \gamma V_{T-1}(s_t) - V_{T-1}(s_{t-1}))e(s)$;
      $e(s) = \lambda\gamma e(s)$
    $t \leftarrow t + 1$
  until *this episode terminates*;
  $T \leftarrow T + 1$
until *all episodes are done*;

K-Step Estimators

- For some convenience in later analysis, let us change the time index by adding 1 everywhere. Thus, the TD(0) update rule becomes,

$$V(s_t) \leftarrow V(s_t) + \alpha_T \left(R_{t+1} + \gamma V(s_{t+1}) - V(s_t)\right)$$

- The interpretation remains the same *i.e.*, estimating the value of a state (s_t) that we are just leaving by moving a little bit (α_T) in the direction of the immediate reward (R_{t+1}) plus the discounted estimated value of the state $(V(s_{t+1}))$ that we just landed in and subtract the value of the state $(V(s_t))$ we just left.
- This basically means a one step look ahead or one step estimator. Lets call it E_1.
- Similarly a two-step estimator (E_2) is,

$$V(s_t) \leftarrow V(s_t) + \alpha_T \left(R_{t+1} + \gamma R_{t+2} + \gamma^2 V(s_{t+2}) - V(s_t)\right)$$

K-Step Estimators

§

$$
\begin{aligned}
E_1 &: V(s_t) \leftarrow V(s_t) + \alpha_T \left(R_{t+1} + \gamma V(s_{t+1}) - V(s_t)\right) \\
E_2 &: V(s_t) \leftarrow V(s_t) + \alpha_T \left(R_{t+1} + \gamma R_{t+2} + \gamma^2 V(s_{t+2}) - V(s_t)\right) \\
E_3 &: V(s_t) \leftarrow V(s_t) + \alpha_T \left(R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 V(s_{t+3}) - V(s_t)\right) \\
&\vdots \\
E_k &: V(s_t) \leftarrow V(s_t) + \alpha_T \left(R_{t+1} + \cdots + \gamma^{k-1} R_{t+k} + \gamma^k V(s_{t+k}) - V(s_t)\right) \\
E_\infty &: V(s_t) \leftarrow V(s_t) + \alpha_T \left(R_{t+1} + \cdots + \gamma^{k-1} R_{t+k} + \cdots - V(s_t)\right)
\end{aligned}
$$

§ E_1: is basically TD(0) and E_∞: is TD(1)

§ Next we will relate these estimators to TD(λ) which will be a weighted combination of all these infinite estimators.

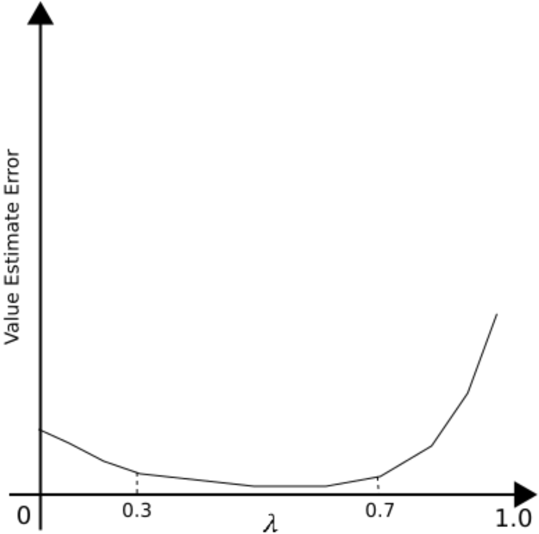
K-Step Estimators and TD(λ)

	λ	$\lambda = 0$	$\lambda = 1$
E_1	$1 - \lambda$	1	0
E_2	$\lambda(1 - \lambda)$	0	0
E_3	$\lambda^2(1 - \lambda)$	0	0
E_k	$\lambda^{k-1}(1 - \lambda)$	0	0
E_∞	λ^∞	0	1

- The idea is when we are updating the value of a state $V(s)$, using any of the TD(λ) methods, all the estimators give their preferences to what the value update should be.
- Checking that the sum of weights is 1.

$$\sum_{k=1}^{\infty} \lambda^{k-1}(1 - \lambda) = (1 - \lambda) \sum_{k=1}^{\infty} \lambda^{k-1}$$
$$= (1 - \lambda) \frac{1}{(1 - \lambda)} = 1$$

Good Value of λ



Unified View: Temporal-Difference Backup

$$V(s_t) \leftarrow V(s_t) + \alpha_T \left(R_{t+1} + \gamma V(s_{t+1}) - V(s_t)\right)$$

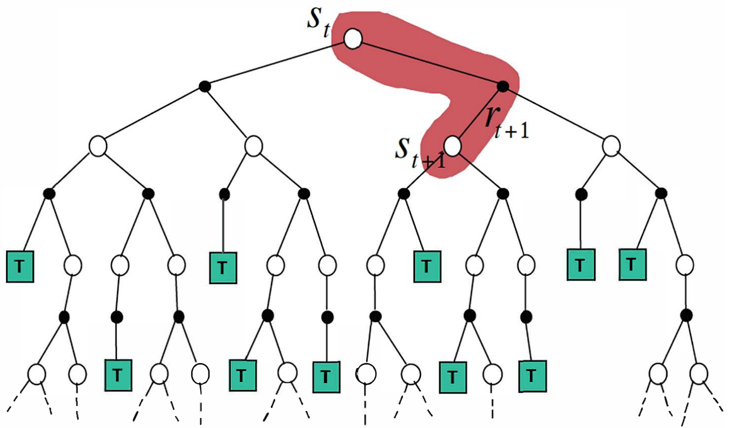


Figure credit: *David Silver, DeepMind*

§ Use of 'sample backups' and 'bootstrapping'.

Unified View: Dynamic Programing Backup

$$v_\pi \doteq v^{(k+1)}(s) \leftarrow \sum_{a \in \mathcal{A}} \pi(a|s) \left\{ r(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s,a) v^{(k)}(s') \right\}$$

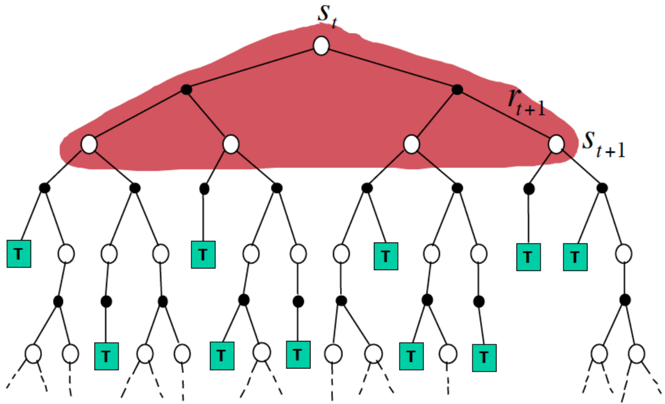


Figure credit: *David Silver, DeepMind*

§ Use of 'full backups' and 'bootstrapping'.

Unified View: Monte-Carlo Backup

$$V(s_t) \leftarrow V(s_t) + \alpha_T \left(G_t - V(s_t)\right)$$

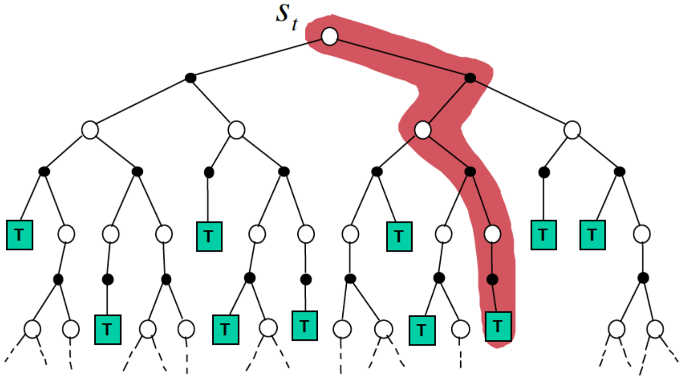
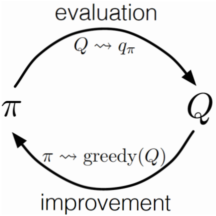


Figure credit: *David Silver, DeepMind*

§ Use of 'sample backups' and no 'bootstrapping'.

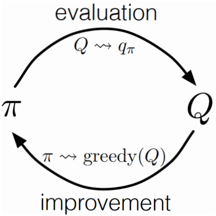
TD Control

- We will now, see how TD estimation can be used in *control*.
- This is mostly like the generalized policy iteration (GPI) where one maintains both an approximate policy and an approximate value function.



TD Control

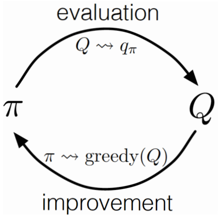
- We will now, see how TD estimation can be used in *control*.
- This is mostly like the generalized policy iteration (GPI) where one maintains both an approximate policy and an approximate value function.



- Policy evaluation is done as TD evaluation
- Then, we can do greedy policy improvement.

TD Control

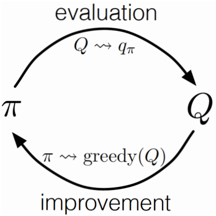
- We will now, see how TD estimation can be used in *control*.
- This is mostly like the generalized policy iteration (GPI) where one maintains both an approximate policy and an approximate value function.



- Policy evaluation is done as TD evaluation
- Then, we can do greedy policy improvement.
- What is the problem!! Remember the MC Lectures!!

TD Control

- We will now, see how TD estimation can be used in *control*.
- This is mostly like the generalized policy iteration (GPI) where one maintains both an approximate policy and an approximate value function.



- Policy evaluation is done as TD evaluation
- Then, we can do greedy policy improvement.
- What is the problem!! Remember the MC Lectures!!
- $\pi'(s) \doteq \arg\max_{a \in \mathcal{A}} \left\{ r(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s,a) v_\pi(s') \right\}$

TD Control

§ Greedy policy improvement over $v(s)$ requires model of MDP

$$\pi'(s) \doteq \arg\max_{a\in\mathcal{A}} \left\{ r(s,a) + \gamma \sum_{s'\in\mathcal{S}} p(s'|s,a) v_\pi(s') \right\}$$

TD Control

§ Greedy policy improvement over $v(s)$ requires model of MDP

$$\pi'(s) \doteq \arg\max_{a \in \mathcal{A}} \left\{ r(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s,a) v_\pi(s') \right\}$$

§ Greedy policy improvement over $Q(s,a)$ is model-free

$$\pi'(s) \doteq \arg\max_{a \in \mathcal{A}} Q(s,a)$$

TD Control

- Greedy policy improvement over $v(s)$ requires model of MDP
  $$\pi'(s) \doteq \arg\max_{a \in \mathcal{A}} \left\{ r(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s,a) v_\pi(s') \right\}$$
- Greedy policy improvement over $Q(s,a)$ is model-free
  $$\pi'(s) \doteq \arg\max_{a \in \mathcal{A}} Q(s,a)$$
- How can we do TD policy evaluation for $Q(s,a)$?

TD Control

- Greedy policy improvement over $v(s)$ requires model of MDP
  $\pi'(s) \doteq \arg\max_{a\in\mathcal{A}} \left\{ r(s,a) + \gamma \sum_{s'\in\mathcal{S}} p(s'|s,a) v_\pi(s') \right\}$
- Greedy policy improvement over $Q(s,a)$ is model-free
  $\pi'(s) \doteq \arg\max_{a\in\mathcal{A}} Q(s,a)$
- How can we do TD policy evaluation for $Q(s,a)$?
- The TD(0) update rule for $V(s)$ is,

$$V_T(s_t) \leftarrow V_{T-1}(s_t) + \alpha_T \left(R_{t+1} + \gamma V_{T-1}(s_{t+1}) - V_{T-1}(s_t) \right)$$

- The TD(0) update rule for $Q(s,a)$ is also similar,

$$Q_T(s_t, a_t) \leftarrow Q_{T-1}(s_t, a_t) + \alpha_T \left(R_{t+1} + \gamma Q_{T-1}(s_{t+1}, a_{t+1}) - Q_{T-1}(s_t, a_t) \right)$$

TD Control

§ Let us spend some time on the update equation.

$$Q_T(s_t, a_t) \leftarrow Q_{T-1}(s_t, a_t) + \alpha_T \left(R_{t+1} + \gamma Q_{T-1}(s_{t+1}, a_{t+1}) - Q_{T-1}(s_t, a_t)\right)$$

§ what we really want in place of the red term is $V_{T-1}(s_{t+1})$.

§ So, why using $Q_{T-1}(s_{t+1}, a_{t+1})$ in place of $V_{T-1}(s_{t+1})$ is fine?

TD Control

- Let us spend some time on the update equation.

$$Q_T(s_t, a_t) \leftarrow Q_{T-1}(s_t, a_t) + \alpha_T \left(R_{t+1} + \gamma Q_{T-1}(s_{t+1}, a_{t+1}) - Q_{T-1}(s_t, a_t)\right)$$

- what we really want in place of the red term is $V_{T-1}(s_{t+1})$.
- So, why using $Q_{T-1}(s_{t+1}, a_{t+1})$ in place of $V_{T-1}(s_{t+1})$ is fine?
- Remember $V(s) = \mathbb{E}_a[Q(s,a)] = \sum_{a \in \mathcal{A}} \pi(a/s)Q(s,a)$.
- So instead of taking the expectation we are replacing it with one sample. So, if we take enough samples, this will eventually converge to $V(s)$.

TD Control

§ Let us spend some time on the update equation.

$$Q_T(s_t, a_t) \leftarrow Q_{T-1}(s_t, a_t) + \alpha_T \left(R_{t+1} + \gamma Q_{T-1}(s_{t+1}, a_{t+1}) - Q_{T-1}(s_t, a_t)\right)$$

§ what we really want in place of the red term is $V_{T-1}(s_{t+1})$.

§ So, why using $Q_{T-1}(s_{t+1}, a_{t+1})$ in place of $V_{T-1}(s_{t+1})$ is fine?

§ Remember $V(s) = \mathbb{E}_a[Q(s,a)] = \sum_{a \in \mathcal{A}} \pi(a/s) Q(s,a)$.

§ So instead of taking the expectation we are replacing it with one sample. So, if we take enough samples, this will eventually converge to $V(s)$.

§ But think carefully again - Could we not have taken the expectation also?

TD Control

§ Like MC Control algorithms, we would use ϵ-soft policies like ϵ-greedy policies for exploration here.

TD Control

§ Like MC Control algorithms, we would use ϵ-soft policies like ϵ-greedy policies for exploration here.

Algorithm 6: On-policy TD Control

```
Parameters: Learning rate α ∈ (0,1], small ε > 0 ;
Initialization: Q(s,a), ∀s ∈ S, a ∈ A arbitrarily except Q(terminal,.) = 0 ;
repeat
    t ← 0, Choose s_t i.e., s_0;
    Pick a_t according to Q(s_t,.) (e.g., ε-greedy);
    repeat
        Apply action a_t from s_t, observe R_{t+1} and s_{t+1};
        Pick a_{t+1} according to Q(s_{t+1},.) (e.g., ε-greedy);
        Q(s_t,a_t) ← Q(s_t,a_t) + α(R_{t+1} + γQ(s_{t+1},a_{t+1}) − Q(s_t,a_t));
        t ← t + 1
    until this episode terminates;
until all episodes are done;
```

TD Control

- Like MC Control algorithms, we would use ϵ-soft policies like ϵ-greedy policies for exploration here.

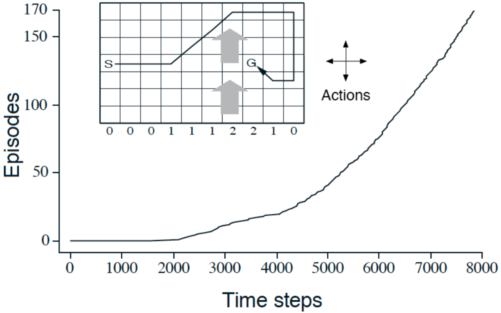
Algorithm 7: On-policy TD Control

Parameters: Learning rate $\alpha \in (0, 1]$, small $\epsilon > 0$;
Initialization: $Q(s, a), \forall s \in \mathcal{S}, a \in \mathcal{A}$ arbitrarily except $Q(terminal, .) = 0$;
repeat
    $t \leftarrow 0$, Choose s_t *i.e.*, s_0;
    Pick a_t according to $Q(s_t, .)$ (*e.g.*, ϵ-greedy);
    repeat
        Apply action a_t from s_t, observe R_{t+1} and s_{t+1};
        Pick a_{t+1} according to $Q(s_{t+1}, .)$ (*e.g.*, ϵ-greedy);
        $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left(R_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)\right)$;
        $t \leftarrow t + 1$
    until *this episode terminates*;
until *all episodes are done*;

- Any guess for the name of this algorithm?

SARSA Example

- The windy-gridworld example is taken from SB [Chapter 6].
- Standard gridworld with start and end states, but upward wind through the middle of the grid. The strength of the wind is given below each column.
- Actions are standard four - `left`, `right`, `up`, `down`. Undiscounted episodic task, with constant rewards of -1 until the goal state is reached.



SARSA Variants

§ Coming back to the question of taking expectation over Q values. This gives what is called an *expected SARSA*.

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\left(R_{t+1} + \gamma \sum_{a \in \mathcal{A}} \pi(a/s_{t+1}) Q(s_{t+1}, a) - Q(s_t, a_t)\right)$$

SARSA Variants

- Coming back to the question of taking expectation over Q values. This gives what is called an *expected SARSA*.

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left(R_{t+1} + \gamma \sum_{a \in \mathcal{A}} \pi(a/s_{t+1}) Q(s_{t+1}, a) - Q(s_t, a_t) \right)$$

- Also can we think of sample backups but no bootstraping? - This will be more like MC control. The TD error term is,

$$R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots + \gamma^{k-1} R_{t+k} + \cdots - Q(s_t, a_t)$$

SARSA Variants

§ Coming back to the question of taking expectation over Q values. This gives what is called an *expected SARSA*.

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left(R_{t+1} + \gamma \sum_{a \in \mathcal{A}} \pi(a/s_{t+1}) Q(s_{t+1}, a) - Q(s_t, a_t) \right)$$

§ Also can we think of sample backups but no bootstraping? - This will be more like MC control. The TD error term is,

$$R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots + \gamma^{k-1} R_{t+k} + \cdots - Q(s_t, a_t)$$

§ Can we also in the same way, think of a spectrum of algorithms like those in between TD(0) and TD(1) a.k.a MC?

k-step SARSA

- Let us define k-step Q-return as,

$$Q_t^{(k)} = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots + \gamma^{k-1} R_{t+k} + \gamma^k Q(s_{t+k}, a_{t+k})$$

- Consider the following k-step returns for $k = 1, 2, \cdots, \infty$

$$k = 1 : Q_t^{(1)} = R_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) (SARSA)$$

$$k = 2 : Q_t^{(2)} = R_{t+1} + \gamma R_{t+2} + \gamma^2 Q(s_{t+2}, a_{t+2})$$

$$k = 3 : Q_t^{(3)} = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 Q(s_{t+3}, a_{t+3})$$

$$\vdots$$

$$k = k : Q_t^{(k)} = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots + \gamma^{k-1} R_{t+k} + \gamma^k Q(s_{t+k}, a_{t+k})$$

$$k = \infty : Q_t^{(\infty)} = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots + \gamma^{k-1} R_{t+k} + \cdots$$

- k-step SARSA updates $Q(s, a)$ towards the k-step Q-return

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left(Q_t^{(k)} - Q(s_t, a_t) \right)$$

SARSA(λ)

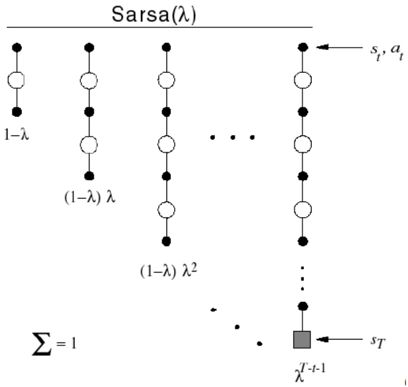


Figure credit: *David Silver, DeepMind*

- The Q^λ return combines all k-step Q-returns $Q_t^{(k)}$.
- Using weight $(1-\lambda)\lambda^{k-1}$

$$Q_t^\lambda = (1-\lambda)\sum_{k=1}^{\infty}\lambda^{k-1}Q_t^{(k)}$$

- The update equation for SARSA(λ) is,
  $Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha\left(Q_t^\lambda - Q(s_t,a_t)\right)$

SARSA(λ)

- Just like TD(λ) evaluation, SARSA(λ) control uses the concept of '*eligibility of states*' in the implementation.
- In TD(λ) evaluation, we had eligibility traces for each state, for SARSA(λ) control we will have eligibility traces for each state-action pair.
- Lets say we get a reward at the end of some step. What eligibility trace says is that the credit for the reward should trickle down in proportion to all the way to the first state. The credit should be more for the state-action pairs which were close to the rewarding step and also for those state-action pairs which were visited frequently along the way.
- $Q(s, a)$ is updated for every state and action in proportion to the TD-error and eligibility of the state-action pair.

SARSA(λ) Algorithm

Initialize $Q(s, a)$ arbitrarily, for all $s \in \mathcal{S}, a \in \mathcal{A}(s)$
Repeat (for each episode):
  $E(s, a) = 0$, for all $s \in \mathcal{S}, a \in \mathcal{A}(s)$
  Initialize S, A
  Repeat (for each step of episode):
    Take action A, observe R, S'
    Choose A' from S' using policy derived from Q (e.g., ε-greedy)
    $\delta \leftarrow R + \gamma Q(S', A') - Q(S, A)$
    $E(S, A) \leftarrow E(S, A) + 1$
    For all $s \in \mathcal{S}, a \in \mathcal{A}(s)$:
      $Q(s, a) \leftarrow Q(s, a) + \alpha \delta E(s, a)$
      $E(s, a) \leftarrow \gamma \lambda E(s, a)$
    $S \leftarrow S'$; $A \leftarrow A'$
  until S is terminal

Figure credit: *David Silver, DeepMind*

SARSA(λ) Gridworld Example

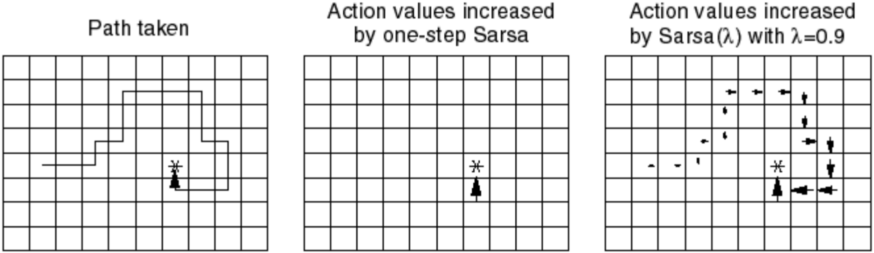


Figure credit: *David Silver, DeepMind*

TD Control

§ The SARSA update rule is

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left(\underbrace{R_{t+1} + \gamma Q(s_{t+1}, a_{t+1})}_{\text{TD Target}} - Q(s_t, a_t) \right)$$

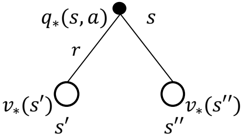
§ The TD target gives a one-step estimate of Q function. Optimal Q function gives the long-term expected reward for taking action a_t at state s_t and then behaving optimally thereafter.

TD Control

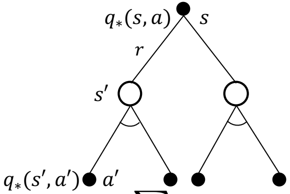
- The SARSA update rule is

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left(\underbrace{R_{t+1} + \gamma Q(s_{t+1}, a_{t+1})}_{\text{TD Target}} - Q(s_t, a_t) \right)$$

- The TD target gives a one-step estimate of Q function. Optimal Q function gives the long-term expected reward for taking action a_t at state s_t and then behaving optimally thereafter.
- Going back to the MDP slides



$$q_*(s,a) = r(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s,a) v_*(s')$$



$$q_*(s,a) = r(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s,a) \max_{a' \in \mathcal{A}} q_*(s', a')$$

Revisiting Bellman equations

§ SARSA:

$$q_\pi(s,a) = r(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s,a) \left\{ \sum_{a' \in \mathcal{A}} \pi(a'|s') q_\pi(s',a') \right\}$$

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha \left(R_{t+1} + \gamma Q(s_{t+1},a_{t+1}) - Q(s_t,a_t) \right)$$

Revisiting Bellman equations

§ SARSA:

$$q_\pi(s,a) = r(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s,a) \left\{ \sum_{a' \in \mathcal{A}} \pi(a'|s') q_\pi(s',a') \right\}$$

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha \left(R_{t+1} + \gamma Q(s_{t+1},a_{t+1}) - Q(s_t,a_t) \right)$$

§ Q-learning:

$$q_*(s,a) = r(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s,a) \max_{a' \in \mathcal{A}} q_*(s',a')$$

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha \left(R_{t+1} + \gamma \max_{a'} Q(s_{t+1},a') - Q(s_t,a_t) \right)$$

Q-learning

Algorithm 8: Off-policy TD Control

Parameters: Learning rate $\alpha \in (0,1]$, small $\epsilon > 0$;
Initialization: $Q(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}$ arbitrarily except $Q(terminal,.) = 0$;
repeat
    $t \leftarrow 0$, Choose s_t *i.e.*, s_0;
    repeat
        Pick a_t according to $Q(s_t,.)$ (*e.g.*, ϵ-greedy);
        Apply action a_t from s_t, observe R_{t+1} and s_{t+1};
        $Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha\left(R_{t+1} + \gamma \max_{a'} Q(s_{t+1},a') - Q(s_t,a_t)\right)$;
        $t \leftarrow t+1$
    until *this episode terminates*;
until *all episodes are done*;

Q-learning

Algorithm 9: Off-policy TD Control

Parameters: Learning rate $\alpha \in (0, 1]$, small $\epsilon > 0$;
Initialization: $Q(s, a), \forall s \in \mathcal{S}, a \in \mathcal{A}$ arbitrarily except $Q(terminal, .) = 0$;
repeat
    $t \leftarrow 0$, Choose s_t *i.e.*, s_0;
    repeat
        Pick a_t according to $Q(s_t, .)$ (*e.g.*, ϵ-greedy);
        Apply action a_t from s_t, observe R_{t+1} and s_{t+1};
        $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left(R_{t+1} + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t) \right)$;
        $t \leftarrow t + 1$
    until *this episode terminates*;
until *all episodes are done*;

§ Note the differences with SARSA. Why is it off-policy?

Q-learning

Algorithm 10: Off-policy TD Control

Parameters: Learning rate $\alpha \in (0, 1]$, small $\epsilon > 0$;
Initialization: $Q(s, a), \forall s \in \mathcal{S}, a \in \mathcal{A}$ arbitrarily except $Q(terminal, .) = 0$;
repeat
    $t \leftarrow 0$, Choose s_t *i.e.*, s_0;
    repeat
        Pick a_t according to $Q(s_t, .)$ (*e.g.*, ϵ-greedy);
        Apply action a_t from s_t, observe R_{t+1} and s_{t+1};
        $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left(R_{t+1} + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t) \right)$;
        $t \leftarrow t + 1$
    until *this episode terminates*;
until *all episodes are done*;

- Note the differences with SARSA. Why is it off-policy?
- Next action is picked after the update here. In SARSA the next action was picked before the update.

Q-learning

- In essence, SARSA picks actions from old Q's and Q-learning picks actions from new Q's.
- Since Q-learning updates the Q values by maximizing over all possible actions, getting the states from a trajectory is not necessary.
- Advantage??

Q-learning

- In essence, SARSA picks actions from old Q's and Q-learning picks actions from new Q's.
- Since Q-learning updates the Q values by maximizing over all possible actions, getting the states from a trajectory is not necessary.
- Advantage?? – Asynchronous update.
- Disadvantage of arbitrarily choosing states for update??

Q-learning

- In essence, SARSA picks actions from old Q's and Q-learning picks actions from new Q's.
- Since Q-learning updates the Q values by maximizing over all possible actions, getting the states from a trajectory is not necessary.
- Advantage?? – Asynchronous update.
- Disadvantage of arbitrarily choosing states for update?? – Like we saw in RTDP, making updates along trajectory makes sure the state-action pairs that are visited frequently *i.e.*, state-action pairs that are important gets to the optimal values quickly.

Q-learning

- In essence, SARSA picks actions from old Q's and Q-learning picks actions from new Q's.
- Since Q-learning updates the Q values by maximizing over all possible actions, getting the states from a trajectory is not necessary.
- Advantage?? – Asynchronous update.
- Disadvantage of arbitrarily choosing states for update?? – Like we saw in RTDP, making updates along trajectory makes sure the state-action pairs that are visited frequently *i.e.*, state-action pairs that are important gets to the optimal values quickly.
- Q-learning generally learns faster than SARSA. This may be due to the fact that Q-learning updates only when it finds a better move. In contrast, SARSA uses the estimate of the next action value in its target. The value thus, changes everytime an exploratory action is taken.

Q-learning

- In essence, SARSA picks actions from old Q's and Q-learning picks actions from new Q's.
- Since Q-learning updates the Q values by maximizing over all possible actions, getting the states from a trajectory is not necessary.
- Advantage?? – Asynchronous update.
- Disadvantage of arbitrarily choosing states for update?? – Like we saw in RTDP, making updates along trajectory makes sure the state-action pairs that are visited frequently *i.e.*, state-action pairs that are important gets to the optimal values quickly.
- Q-learning generally learns faster than SARSA. This may be due to the fact that Q-learning updates only when it finds a better move. In contrast, SARSA uses the estimate of the next action value in its target. The value thus, changes everytime an exploratory action is taken.
- There are some undesirable situations also for Q-learning.

Q-learning

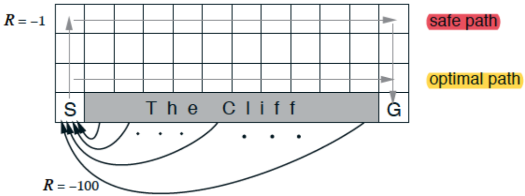


Figure credit: *[SB-Chapter 6]*