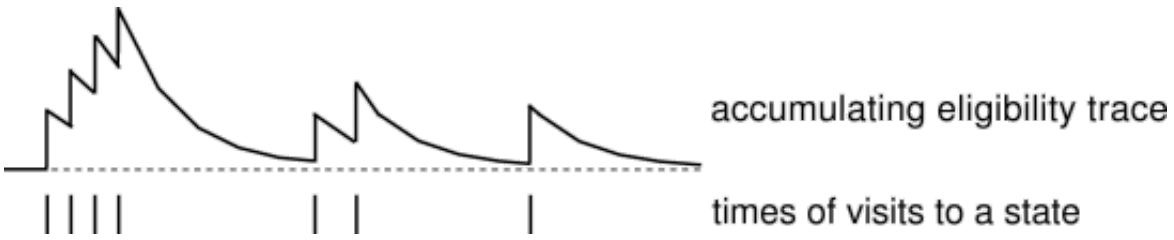
7.3 The Backward View of TD(λ)

In the previous section we presented the forward or theoretical view of the tabular TD(λ) algorithm as a way of mixing backups that parametrically shifts from a TD method to a Monte Carlo method. In this section we instead define TD(λ) mechanistically, and in the next section we show that this mechanism correctly implements the forward view. The mechanistic, or *backward*, view of TD(λ) is useful because it is simple conceptually and computationally. In particular, the forward view itself is not directly implementable because it is *acausal*, using at each step knowledge of what will happen many steps later. The backward view provides a causal, incremental mechanism for approximating the forward view and, in the off-line case, for achieving it exactly.

In the backward view of TD(λ), there is an additional memory variable associated with each state, its *eligibility trace*. The eligibility trace for state s at time t is denoted $e_t(s) \in \Re^+$. On each step, the eligibility traces for all states decay by $\gamma\lambda$, and the eligibility trace for the one state visited on the step is incremented by 1:

$$e_t(s) = \begin{cases} \gamma\lambda e_{t-1}(s) & \text{if } s \neq s_t; \\ \gamma\lambda e_{t-1}(s) + 1 & \text{if } s = s_t, \end{cases} \qquad (7.5)$$

for all nonterminal states s, where γ is the discount rate and λ is the parameter introduced in the previous section. Henceforth we refer to λ as the *trace-decay parameter*. This kind of eligibility trace is called an *accumulating* trace because it accumulates each time the state is visited, then fades away gradually when the state is not visited, as illustrated below:



At any time, the traces record which states have recently been visited, where "recently" is defined in terms of $\gamma\lambda$. The traces are said to indicate the degree to which each state is *eligible* for undergoing learning changes should a reinforcing event occur. The reinforcing events we are concerned with are the moment-by-moment one-step TD errors. For example, the TD error for state-value prediction is

$$\delta_t = r_{t+1} + \gamma V_t(s_{t+1}) - V_t(s_t). \qquad (7.6)$$

In the backward view of TD(λ), the global TD error signal triggers proportional updates to all recently visited states, as signaled by their nonzero traces:

$$\Delta V_t(s) = \alpha \delta_t e_t(s), \qquad \text{for all } s \in \mathcal{S}. \qquad (7.7)$$

As always, these increments could be done on each step to form an on-line algorithm, or saved until the end of the episode to produce an off-line algorithm. In either case, equations ((7.5)-(7.7)) provide the mechanistic definition of the TD(λ) algorithm. A complete algorithm for on-line TD(λ) is given in Figure 7.7.

Initialize $V(s)$ arbitrarily and $e(s) = 0$, for all $s \in \mathcal{S}$
Repeat (for each episode):
  Initialize s
  Repeat (for each step of episode):
    $a \leftarrow$ action given by π for s
    Take action a, observe reward, r, and next state, s'
    $\delta \leftarrow r + \gamma V(s') - V(s)$
    $e(s) \leftarrow e(s) + 1$
    For all s:
      $V(s) \leftarrow V(s) + \alpha \delta e(s)$
      $e(s) \leftarrow \gamma \lambda e(s)$
    $s \leftarrow s'$
  until s is terminal

Figure 7.7: On-line tabular TD(λ).

The backward view of TD(λ) is oriented backward in time. At each moment we look at the current TD error and assign it backward to each prior state according to the state's eligibility trace at that time. We might imagine ourselves riding along the stream of states, computing TD errors, and shouting them back to the previously visited states, as suggested by Figure 7.8. Where the TD error and traces come together, we get the update given by (7.7).

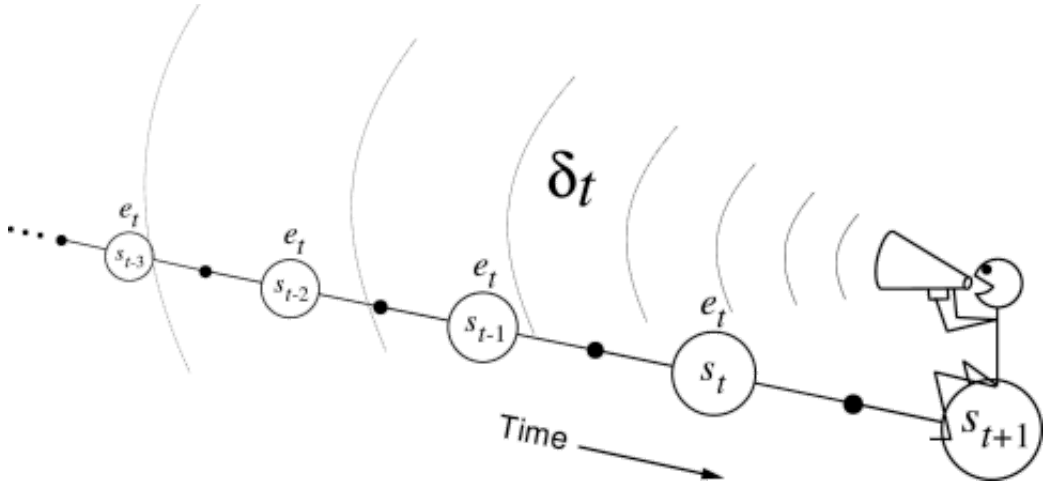


Figure 7.8: The backward or mechanistic view. Each update depends on the current TD error combined with traces of past events.

To better understand the backward view, consider what happens at various values of λ. If $\lambda = 0$, then by (7.5) all traces are zero at t except for the trace corresponding to s_t. Thus the TD(λ) update (7.7) reduces to the simple TD rule (6.2), which we henceforth call TD(0). In terms of Figure 7.8, TD(0) is the case in which only the one state preceding the current one is changed by the TD error. For larger values of λ, but still $\lambda < 1$, more of the preceding states are changed, but each more temporally distant state is changed less because its eligibility trace is smaller, as suggested in the figure. We say that the earlier states are given less *credit* for the TD error.

If $\lambda = 1$, then the credit given to earlier states falls only by γ per step. This turns out to be just the right thing to do to achieve Monte Carlo behavior. For example, remember that the TD error, δ_t, includes an undiscounted term of r_{t+1}. In passing this back k steps it needs to be discounted, like any reward in a return, by γ^k, which is just what the falling eligibility trace achieves. If $\lambda = 1$ and $\gamma = 1$, then the eligibility traces do not decay at all with time. In this case the method behaves like a Monte Carlo method for an undiscounted, episodic task. If $\lambda = 1$, the algorithm is also known as TD(1).

TD(1) is a way of implementing Monte Carlo algorithms that is more general than those presented earlier and that significantly increases their range of applicability. Whereas the earlier Monte Carlo methods were limited to episodic tasks, TD(1) can be applied to discounted continuing tasks as well. Moreover, TD(1) can be performed incrementally and on-line. One disadvantage of Monte Carlo methods is that they learn nothing from an episode until it is over. For example, if a Monte Carlo control method does something that produces a very poor reward but does not end the episode, then the agent's tendency to do that will be undiminished during the episode. On-line TD(1), on the other hand, learns in an n-step TD way from the incomplete ongoing episode, where the n steps are all the way up to the current step. If something unusually good or bad happens during an episode, control methods based on TD(1) can learn immediately and alter their behavior on that same episode.
