Going from the forward view to the backward view for TD(λ):

Idea: Store an additional memory variable for each state, called the eligibility trace, \( e_t(s) \), also called the e-trace.

- Quantifies how much \( s \) should be updated if there is a TD error in the current step

\[
e_t(s) = \begin{cases} \gamma \lambda e_{t-1}(s) & \text{if } s \neq s_t \\ \gamma \lambda e_{t-1}(s) + 1 & \text{otherwise} \end{cases}
\]

(Alternative is "Replacing traces".)

\( e_t(s) = 0 \) for all \( s \)
TD(\(\lambda\)) Algorithm (Backward View)

\[ e_t(s) = 0 \text{ for all } s, \text{ for each episode} \]
\[ \delta = R_t + \nu(S_{t-1}) - \nu(S_t) \]
\[ \forall s: e_t(s) \leftarrow \gamma \lambda e_t(s) \]
\[ e_t(s_t) \leftarrow e_t(s_t) + 1 \]
\[ \forall s: \nu_t(s) \leftarrow \nu_t(s) + \alpha \delta e_t(s) \]

**Proof of equiv. of forward & backward views:**

[No hard math, just long!]

**Notation:**

\[ S_t = R_t + \gamma \nu(S_{t+1}) - \nu(S_t) \]
\[ I_{ss_t} = 1 \text{ if } s = s_t \text{ and } 0 \text{ otherwise} \]
\[ e_t(s) = \sum_{k=0}^{\lambda-1} (\delta \lambda)^t I_{ss_k} \]
\[ \Delta V^F_t(s) \equiv \text{update at time } t \text{ of } \nu_t(s) \]
\[ \text{according to the forward view:} \alpha (G_t - \nu_t(s)) \]
\[ \Delta V^B_t(s) \equiv \text{likewise for the backward view:} \alpha \delta e_t(s) \]

\(\lambda = 0 \rightarrow \text{TD} \)
\(\lambda = 1 \rightarrow \text{MC} \)
\(0 < \lambda < 1 \rightarrow \text{TD(}\lambda\text{)}\)

**Easy to apply to more general case (weight updates):**

**Want to show: assumes same actions, rewards, transitions.**

\[ \forall s \in S : \frac{\lambda-1}{\lambda} \Delta V^B_t(s) = \sum_{t=0}^{\lambda-1} \Delta V^F_t(s) I_{ss_t} \]

where \(\lambda\) is the length of the episode.

(only \(s_t\) is updated)
Now, RHS for a single update:

\[ \Delta V_t^F(s) = \alpha (G_t^\lambda - V_t(s_t)) \]

\[ \frac{1}{\alpha} \Delta V_t^F(s) = -V_t(s_t) + G_t^\lambda \]

\[ = -V_t(s_t) + (1-\lambda) \lambda^0 (R_t + \gamma V_t(s_{t+1})) \]

\[ + (1-\lambda) \lambda^1 (R_t + \gamma R_{t+1} + \gamma^2 V_t(s_{t+2})) \]

\[ + \cdots \]

\[ = -V_t(s_t) + (1-\lambda) \sum_{i=0}^{\infty} \lambda^i R_t \]

\[ + (1-\lambda) \sum_{i=0}^{\infty} \lambda^i R_{t+i} \]

\[ + \left[ V_t \text{ terms} \right] \]

\[ = -V_t(s_t) + \sum_{k=0}^{\infty} (\lambda^k R_t + R_{t+k} + \sum_{k=0}^{\infty} (1-\lambda) \lambda^k \gamma^{k+1} V_t(s_{t+k+1})) \]

\[ = -V_t(s_t) + \sum_{k=0}^{\infty} (\lambda^k (R_{t+k} + \gamma V_t(s_{t+k+1}) - \gamma V_t(s_{t+k+1}))) \]

\[ = \sum_{k=0}^{\infty} (\lambda^k (R_{t+k} + \delta V_t(s_{t+k+1}) - V_t(s_{t+k}))) \]

\[ = \sum_{k=0}^{\infty} (\delta \lambda)^k \delta s_k \]
The proof works if $S_t$ always uses $V_t$. But real updates don't do that in the Backward view. Using the actual $V$, the two views are equal up to a difference of $O(\lambda^2)$.

$$\Delta V_t(\lambda)_{S_{t+1}} = \frac{1}{\lambda} \sum_{k=t}^{t+1} (\lambda_k)^{t-k} S_k$$

Coming up: In some cases can adjust Backward View a little and get the Forward + Backward equivalent. Will also apply TD($\lambda$) idea to Sarsa ($\lambda$) and $q$-learning ($q(\lambda)$).