Eligibility Traces

- basics of RL
- $TD(\lambda)$, λ is an eligibility trace
+ easily combinable with any TD (SARSA, Q learning, etc)
- λ unifies + generalize TD & MC methods
+ $\lambda = 1$, MC, $\lambda = 0$, TD
+ inbetween, method often better than either end
- how diff from n-step methods?
+ use eligibility trace $z_t \in \mathbb{R}^d$

x when a weight in w participates, its component in z bumps up, then fades away according to λ, called trace-decay param.
* computationally more efficient
* learning is uniform & in a single step rather than time delayed
- easier to implement
+ most algo requires forward views, what isn't always available
+ can do almost, if not, exactly, the same, using backward views
- look at state values, prediction, extend action-value, then control

The λ-return

- $G_{t:t+n} = R_{t+1} + \gamma R_{t+2} + \dots + \gamma^{n-1} R_{t+n} + \gamma^n \hat{v}(S_{t+n-1}, w_{t+n-1})$

- valid update target is not just any n-step but also any avg n-step returns for diff-n's
+ e.g. $\frac{1}{2} G_{t:t+2} + \frac{1}{2} G_{t:t+4}$

+ produce compacted errors & guaranteed convergence
x e.g. avg 1-step & ∞-step returns exp updates with DP updates
- avg update of simple = compound update
+ update can only occur when longest component is done
- $TD(\lambda)$ avg's n-step updates

$$G_t^\lambda \doteq (1-\lambda) \sum_{n=1}^{\infty} \lambda^{n-1} G_{t:t+n}$$

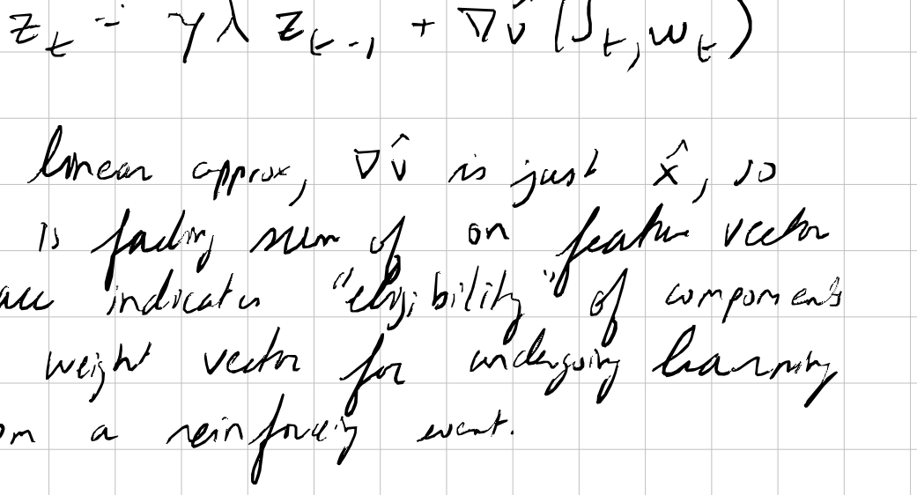
+ $(1-\lambda)$ normalizes to ensure sum = 1
x each return fades with λ
x can rewrite post termination state

$$G_t^\lambda = (1-\lambda) \sum_{n=1}^{T-t-1} \lambda^{n-1} G_{t:t+n} + \lambda^{T-t-1} G_t$$

- off-line λ-return algorithm

$$w_{t+1} \doteq w_t + \alpha \left[G_t^\lambda - \hat{v}(S_t, w_t) \right] \nabla \hat{v}(S_t, w_t)$$

- performance of λ similar to choice of n
- forward-view model



$TD(\lambda)$

- computationally congenial backwards view
- 3 improvements
  1) update at every step, not end. Better, sooner estimates
  2) computation distributed across time, not bulk at end
  3) continuing & episodic application
- z_t initialized with 0 values

$$z_{-1} \doteq 0$$
$$z_t \doteq \gamma \lambda z_{t-1} + \nabla \hat{v}(S_t, w_t)$$

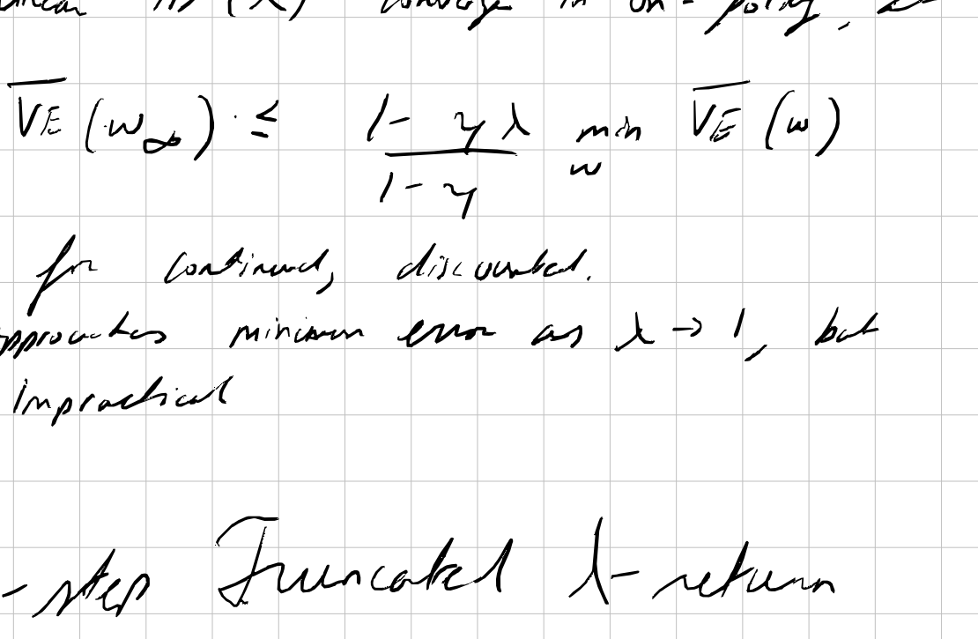
+ in linear approx, $\nabla \hat{v}$ is just $\hat{x}$, so z_t is fading sum of on feature vector
- trace indicates "eligibility" of components in weight vector for undergoing learning from a reinforcing event.

- $\delta_t \doteq R_{t+1} + \gamma \hat{v}(S_{t+1}, w_t) - \hat{v}(S_t, w_t)$
- $w_{t+1} \doteq w_t + \alpha \delta_t z_t$

Semi-gradient $TD(\lambda)$ for $\hat{v} \approx v_\pi$

```
inputs:
  policy $\pi$ for eval
  differentiable func $\hat{v}: S^+ \times \mathbb{R}^d \to \mathbb{R}$, $\hat{v}(term) = 0$
algo params:
  $\alpha > 0$
  $\lambda \in [0, 1]$
  $w \in \mathbb{R}^d$ arbitrarily

for each ep
  init S
  $z = 0$
  for each step in ep:
    $A = \pi(\cdot | S)$
    $S', R$ = env.step(A)
    $z = \gamma \lambda z + \nabla \hat{v}(S, w)$
    $\delta = R + \gamma \hat{v}(S', w) - \hat{v}(S, w)$
    $w = w + \alpha \delta z$
    $S = S'$
```



- backwards, or mechanistic, view
- smaller, or more distant states are given less credit for contributing to current error

- for undiscounted episodic MC method, $\lambda = 1$ $\gamma = 1$
+ if $\lambda = 1$, called $TD(1)$
+ $TD(1)$ is more general, easier to apply
x can do discounting, can do online, can do incremental
* $TD(1)$ can stay away from bad actions mid learn, MC cannot.

- linear $TD(\lambda)$ converge in on-policy, at

$$\overline{VE}(w_\infty) \leq \frac{1 - \gamma \lambda}{1 - \gamma} \min_w \overline{VE}(w)$$

for continued, discounted.
+ approaches minimum error as $\lambda \to 1$, but impractical

n-step Truncated λ-return Methods

- problem: λ-return still unknown till end of ep
+ in continuing case, never known
+ however, dependence of future rewards falls off by $\gamma\lambda$, so some approximation would be good
- truncated λ return

$$G_{t:h}^\lambda \doteq (1-\lambda) \sum_{n=1}^{h-t-1} \lambda^{n-1} G_{t:t+n} + \lambda^{h-t-1} G_{t:h}$$

+ simple return up to horizon h
- family of n-step λ-returns when all $1 \leq k \leq n$ returns are used instead of just nth return
+ for state values, called Truncated $TD(\lambda)$ or $TTD(\lambda)$

x $w_{t+n} \doteq w_{t+n-1} + \alpha \left[G_{t:t+n}^\lambda - \hat{v}(S_t, w_{t+n-1}) \right] \cdot \nabla \hat{v}(S_t, w_{t+n-1})$

- efficient impl can be done using k-return:

$$G_{t:t+k}^\lambda = \hat{v}(S_t, w_{t-1}) + \sum_{i=t}^{t+k-1} (\gamma\lambda)^{i-t} \delta_t'$$

$$\delta_t' = R_{t+1} + \gamma \hat{v}(S_{t+1}, w_t) - \hat{v}(S_t, w_{t-1})$$

Redoing Updates: Online λ-return Algorithm

- how to find the right n?
+ updates are done every step, with better info
+ $t = 1$, we know $G_{0:1}^\lambda$
  $t = 2$, we know $G_{0:2}^\lambda$, $G_{1:2}^\lambda$
x weight vector changes at each horizon, w_t^h

$h = 1$, $w_1^1 = w_0^1 + \alpha \left[G_{0:1}^\lambda - \hat{v}(S_0, w_0^1) \right] \nabla \hat{v}(S_0, w_0^1)$

$h = 2$, $w_1^2 = w_0^2 + \alpha \left[G_{0:2}^\lambda - \hat{v}(S_0, w_0^2) \right] \nabla \hat{v}(S_0, w_0^2)$

$w_2^2 = w_1^2 + \alpha \left[G_{1:2}^\lambda - \hat{v}(S_0, w_0^2) \right] \nabla \hat{v}(S_0, w_1^2)$

. . . -

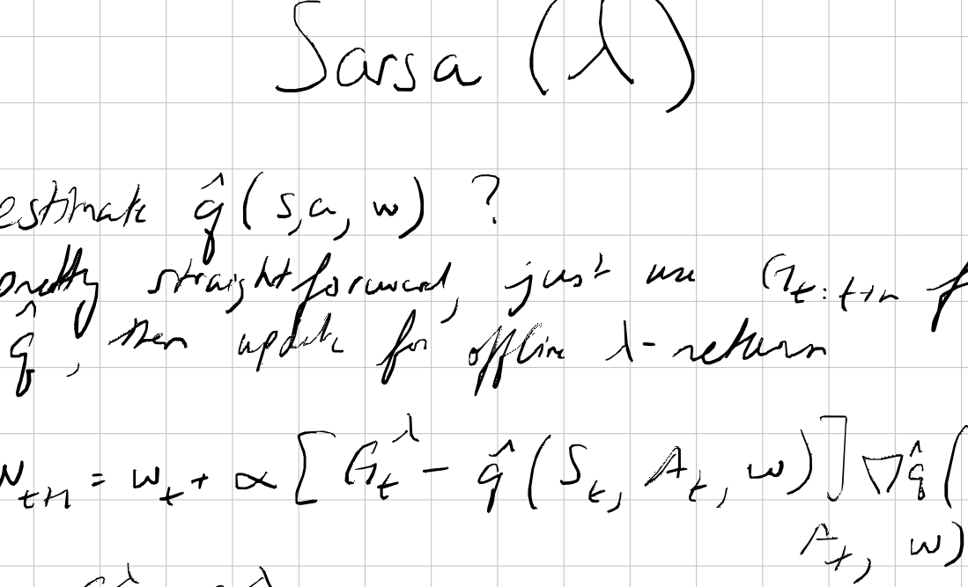
$$w_{t+1}^h \doteq w_t^h + \alpha \left[G_{t:h}^\lambda - \hat{v}(S_t, w_t^h) \right] \nabla \hat{v}(S_t, w_t^h)$$

- thus, $w_t \doteq w_t^t$, online λ-return algo
- strictly more computationally complex because it maintains the weight vector between every timestep
adv: better judgment at all times, even end of episode

True Online $TD(\lambda)$

- use eligibility trace with backward view & linear approx for "true" match to ideal online λ-alg

- this is how the weight vectors look



- In linear, $\hat{v}(s, w) = w^T x(s)$, then:

$$w_{t+1} \doteq w_t + \alpha \delta_t z_t + \alpha (w_t^T x_t - w_{t-1}^T x_t)(z_t - x_t)$$

$$x_t = x(S_t)$$
$$z_t = \gamma \lambda z_{t-1} + (1 - \alpha \gamma \lambda z_{t-1}^T x_t) x_t$$

- technically called a dutch trace
+ previous trace called accumulating trace

- Skipping Dutch Traces in MC (history lesson)

Sarsa (λ)

- estimate $\hat{q}(s, a, w)$?
- pretty straightforward, just use $G_{t:t+n}$ for $\hat{q}$, then update for offline λ-return

$$w_{t+1} = w_t + \alpha \left[G_t^\lambda - \hat{q}(S_t, A_t, w) \right] \nabla \hat{q}(S_t, A_t, w)$$

$$G_t^\lambda = G_{t:\infty}^\lambda$$

- for Sarsa, same update,

$$w_{t+1} = w_t + \alpha \delta_t z_t$$

with $z_{-1} = 0$

$$z_t = \gamma \lambda z_{t-1} + \nabla \hat{q}(S_t, A_t, w_t)$$

- true online $TD(\lambda)$ exists, therefore, so does true online Sarsa (λ)

- also, truncated version, called forward Sarsa (λ) for model free control in multilayer ANNs

Variable λ & γ

- TD algos, final form!
- λ, γ aren't scalars, they are functions of state and action!
+ λ_t, γ_t
+ $\lambda: S \times A \to [0, 1]$ $\quad \lambda_t = \lambda(S_t, A_t)$
  $\gamma: S \to [0, 1]$ $\quad \gamma_t = \gamma(S_t)$
x γ_t is now the termination function
x now return is

$$G_t \doteq R_{t+1} + \gamma_{t+1} G_{t+1}$$
$$= R_{t+1} + \gamma_{t+1} R_{t+2} + \gamma_{t+1} \gamma_{t+2} R_{t+3} + \dots$$
$$= \sum_{k=t}^{\infty} \left(\prod_{i=t+1}^{k} \gamma_i \right) R_{k+1}$$

$\prod_{k=t}^{\infty} \gamma_k = 0$ to assume finite sums

- convenience of definition: episodic setting presented in a single stream of experience
+ no special terminal states, start distributions, or termination times

- only affects solution strategy

$$G_t^{\lambda s} \doteq R_{t+1} + \gamma_{t+1} \left((1 - \lambda_{t+1}) \hat{v}(S_{t+1}, w_t) + \lambda_{t+1} G_{t+1}^{\lambda s} \right)$$

+ bootstrapping from state values (s's)
+ add a's for action values

+ in English

"the return at time t is the immediate reward, undiscounted, unaffected by bootstrapping, plus a possible second term to the extent we are not discounting at the next state (i.e. γ_{t+1} isn't 0, e.g. for term). If we aren't, we use some ratio of bootstrapping or λ-return of t+1"

- Expected Sarsa:

$$G_t^{\lambda a} \doteq R_{t+1} + \gamma_{t+1} \left((1 - \lambda_{t+1}) \bar{V}_t(S_{t+1}) + \lambda_{t+1} G_{t+1}^{\lambda a} \right)$$

$$\bar{V}_t(s) = \sum_a \pi(a|s) \hat{q}(s, a, w)$$

Off Policy Traces With Control Variates

- incorporate importance sampling
+ only applicable for bootstrapped

$$G_t^{\lambda s} \doteq \rho_t \left(R_{t+1} + \gamma_{t+1} \left((1 - \lambda_{t+1}) \hat{v}(S_{t+1}, w_t) + \lambda_{t+1} G_{t+1}^{\lambda} \right) \right) + (1 - \rho_t) \hat{v}(S_t, w_t)$$

- can still be approximated by

$$\delta_t^s \doteq R_{t+1} + \gamma_{t+1} \hat{v}(S_{t+1}, w_t) - \hat{v}(S_t, w_t)$$

$$G_t^{\lambda s} \approx \hat{v}(S_t, w_t) + \rho_t \sum_{k=t}^{\infty} \delta_k^s \prod_{i=t+1}^{k} \gamma_i \lambda_i \rho_i$$

- can easily be used to a forward view update

$$w_{t+1} = w_t + \alpha \left(G_t^{\lambda s} - \hat{v}(S_t, w_t) \right) \nabla \hat{v}(S_t, w_t)$$

$$\approx w_t + \alpha \rho_t \left(\sum_{k=t}^{\infty} \delta_k^s \prod_{i=t+1}^{k} \gamma_i \lambda_i \rho_i \right) \nabla \hat{v}(S_t, w_t)$$

- eligibility trace with semi-gradient update creates a general on / off policy $TD(\lambda)$
+ off-policy is just not guaranteed stability

- for action value, eligibility trace with Sarsa (λ) is state of art
- $\lambda = 1$ still weaker than MC because updates are still made
+ cancel out in expected value
x there does exist a method for exact equivalence call $PTD(\lambda)$

Watkins $Q(\lambda)$ to Tree-Backup (λ)

- how to apply eligibility trace to Q-learning
- Watkins $Q(\lambda)$ decays elig trace normally, then cuts trace to zero after non-greedy selection

$$G_t^{\lambda a} \approx \hat{q}(S_t, A_t, w_t) + \sum_{k=t}^{\infty} \delta_k^a \prod_{i=t+1}^{k} \gamma_i \lambda_i \pi(A_i | S_i)$$

- Final form TB(λ)

$$z_t \doteq \gamma_t \lambda_t \pi(A_t | S_t) z_{t-1} + \nabla \hat{q}(S_t, A_t, w_t)$$

Stable Off-policy Methods with Traces

- all use Gradient TD or Emphatic TD with linear func approx.
- $GTD(\lambda)$ like TDC
+ learns $w_t \Rightarrow \hat{v}(s, w) = w_t^T x(s) \approx v_\pi(s)$ even if data is from off policy b
+ update

$$w_{t+1} \doteq w_t + \alpha \delta_t^s z_t - \alpha \gamma_{t+1} (1 - \lambda_{t+1}) (z_t^T v_t) x_{t+1}$$

$$v_{t+1} \doteq v_t + \beta \delta_t^s z_t - \beta (v_t^T x_t) x_t$$

- $GQ(\lambda)$ is $GTD(\lambda)$ for action values
- $HTD(\lambda)$ is hybrid of $GTD(\lambda)$ & $TD(\lambda)$
- Emphatic $TD(\lambda)$ extends one step Emphatic TD

Implementation Issues

- most of z is zero, few nonzero
- generally not a problem

Conclusion

- eligibility trace + TD error = efficient, incremental way of going from one step TD to full MC
- online is the biggest win

- can go from how t relates to t+1 (forward view) to how t+1 relates to t (backwards view)

- computationally expensive but good for faster learning and delayed rewards on scarce data