9 Eligibility Traces

Melih Kandemir

Özyeğin University
Computer Science Department
melih.kandemir@ozyegin.edu.tr

Lecture 9

Intro

- Eligibility traces formulate another way to bridge the gap between TD(0) and MC.
- $\lambda = 0$ is one-step TD and $\lambda = 1$ is MC.
- Eligibility traces substantially reduce the computational efficiently.
- An *eligibility trace* is a short-term memory of reinforcing events.

Compound returns

The $n-$step return was formulated earlier as

$$G_{t:t+n} \triangleq R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{n-1} R_{t+n} + \gamma^n \hat{v}(S_{t+n}, \mathbf{w}_{t+n-1}),$$
$$0 \leq t \leq T - n.$$

Define a **compound return** as the average of $2-$step and $4-$step returns

$$G_t \triangleq \frac{1}{2} G_{t:t+2} + \frac{1}{2} G_{t:t+4}$$

The related update is called a **compound update.**

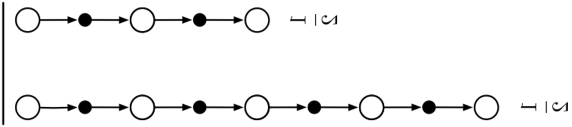

Figure: R. Sutton and A. Barto, MIT Press, 2017

The $\lambda-$step return

Generalized to infinite components with proper weights, we attain the λ-return

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

- if $\lambda = 0$, then we get one-step TD
- if $\lambda = 1$, we get MC
- $(1-\lambda)$ is the normalizer. Note:

$$\lambda^0 + \lambda^1 + \cdots + \lambda^\infty = \frac{1}{1-\lambda}.$$

TD(λ)

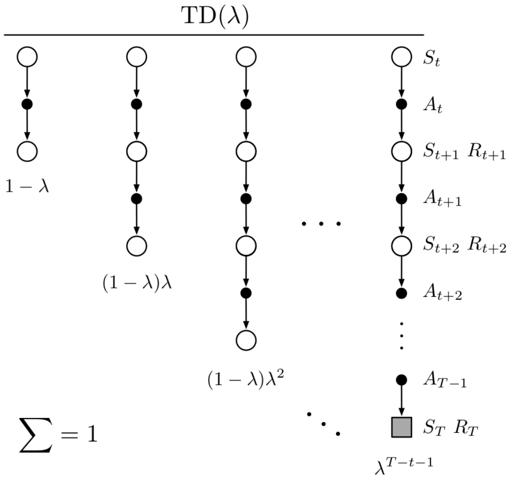

Figure: R. Sutton and A. Barto, MIT Press, 2017

TD(λ) for an episode of length T

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

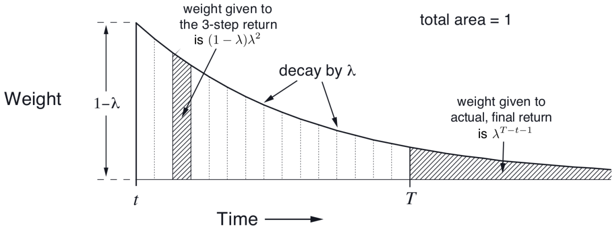

Figure: R. Sutton and A. Barto, MIT Press, 2017

Offline $\lambda-$return algorithm

Only observe until the end of the episode and then replay all the state transitions while updating by

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \alpha\left[G_t^\lambda - \hat{v}(S_t, \mathbf{w}_t)\right]\nabla\hat{v}(S_t, \mathbf{w}_t), \quad t = 0, ..., T-1.$$

19-state random walk results

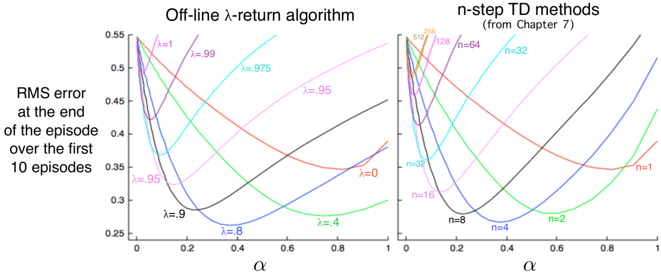

Figure: R. Sutton and A. Barto, MIT Press, 2017

Off-line $\lambda-$return vs TD(λ)

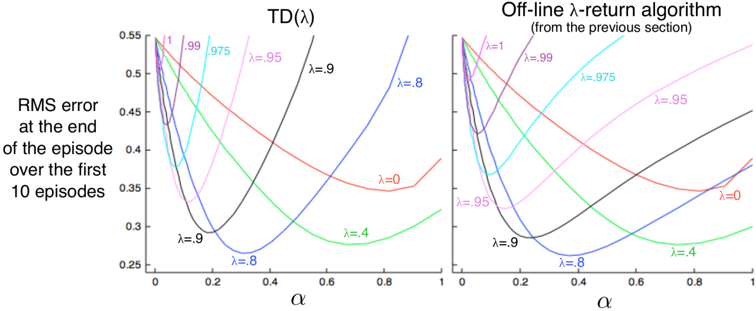

Figure: R. Sutton and A. Barto, MIT Press, 2017

The eligibility trace

Keep track of the components of the weight vector that have contributed to learning in the recent past (being in the last $\gamma\lambda$ time steps). An **eligibility trace** is a vector of the same size as the weight vector, to which the below updates are applied

$$\begin{aligned} \mathbf{e}_{-1} &\leftarrow \mathbf{0}, \\ \mathbf{e}_t &\leftarrow \gamma\lambda \mathbf{e}_{t-1} + \nabla \hat{v}(S_t, \mathbf{w}_t), \quad 0 \le t \le T. \end{aligned}$$

For the conventional TD error

$$\delta_t \triangleq R_{t+1} + \gamma \hat{v}(S_{t+1}, \mathbf{w}_t) - \hat{v}(S_t, \mathbf{w}_t),$$

perform learning using the update

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \alpha \delta_t \mathbf{e}_t.$$

Forward and backward views

The forward view

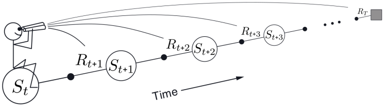

Figure: R. Sutton and A. Barto, MIT Press, 2017

The backward view

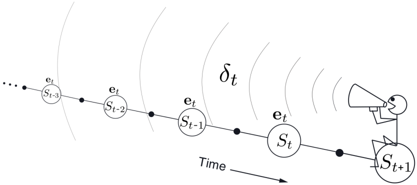

Figure: R. Sutton and A. Barto, MIT Press, 2017

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

Input: the policy π to be evaluated
Input: a differentiable function $\hat{v} : \mathcal{S}^+ \times \mathbb{R}^d \to \mathbb{R}$ such that $\hat{v}(\text{terminal},\cdot) = 0$

Initialize value-function weights $\mathbf{w}$ arbitrarily (e.g., $\mathbf{w} = \mathbf{0}$)
Repeat (for each episode):
  Initialize S
  $\mathbf{e} \leftarrow \mathbf{0}$ (An n-dimensional vector)
  Repeat (for each step of episode):
  . Choose $A \sim \pi(\cdot|S)$
  . Take action A, observe R, S'
  . $\mathbf{e} \leftarrow \gamma\lambda\mathbf{e} + \nabla\hat{v}(S,\mathbf{w})$
  . $\delta \leftarrow R + \gamma\hat{v}(S',\mathbf{w}) - \hat{v}(S,\mathbf{w})$
  . $\mathbf{w} \leftarrow \mathbf{w} + \alpha\delta\mathbf{e}$
  . $S \leftarrow S'$
  until S' is terminal

Figure: R. Sutton and A. Barto, MIT Press, 2017

The truncated $\lambda-$return

- The effect of future terms decays exponentially.
- Truncate after h.

$$G_{t:h}^{\lambda} = (1-\lambda)\sum_{n=1}^{h-t-1} \lambda^{n-1} G_{t:t+n} + \lambda^{h-t-1} G_{t:h}, \quad 0 \leq t < h \leq T$$

The resultant algorithm is called Truncated TD(λ) or TTD(λ).

The truncated $\lambda-$return

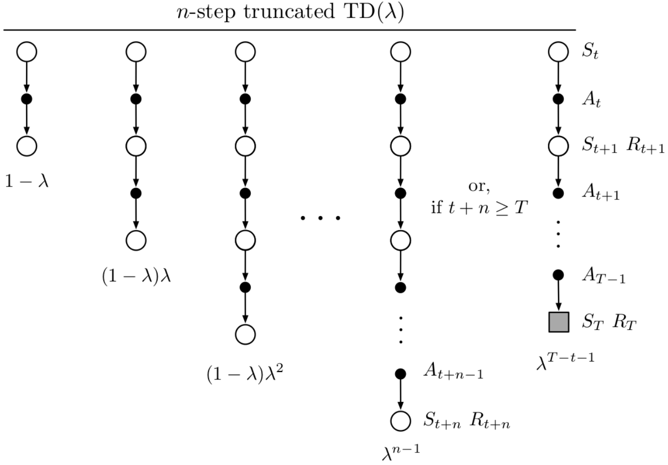

Figure: R. Sutton and A. Barto, MIT Press, 2017

Dutch traces in MC

$$
\begin{aligned}
\mathbf{w}_T &= \mathbf{w}_{T-1} - \alpha\big(G - \mathbf{w}_{T-1}T\mathbf{x}_{T-1}\big)\mathbf{x}_{T-1} \\
&= \mathbf{w}_{T-1} + \alpha\mathbf{x}_{T-1}\big(-\mathbf{x}_{T-1}^T\mathbf{w}_{T-1}\big) + \alpha G\mathbf{x}_{T-1} \\
&= \big(\mathbf{I} - \alpha\mathbf{x}_{T-1}\mathbf{x}_{T-1}^T\big)\mathbf{w}_{T-1} + \alpha G\mathbf{x}_{T-1} \\
&= \mathbf{F}_{T-1}\mathbf{w}_{T-1} + \alpha G\mathbf{x}_{T-1}
\end{aligned}
$$

where $\mathbf{F}_t \triangleq \mathbf{I} - \alpha\mathbf{x}_t\mathbf{x}_t^T$ is a *forgetting*,or *fading*, matrix.

Dutch traces in MC

$$
\begin{aligned}
&= \mathbf{F}_{T-1}(\mathbf{F}_{T-2}\mathbf{w}_{T-2} + \alpha G \mathbf{x}_{T-2}) + \alpha G \mathbf{x}_{T-1} \\
&= \mathbf{F}_{T-1}\mathbf{F}_{T-2}\mathbf{w}_{T-2} + \alpha G(\mathbf{F}_{T-1}\mathbf{x}_{T-2} + \mathbf{x}_{T-1}) \\
&= \mathbf{F}_{T-1}\mathbf{F}_{T-2}(\mathbf{F}_{T-3}\mathbf{w}_{T-3} + \alpha G \mathbf{x}_{T-3}) + \alpha G(\mathbf{F}_{T-1}\mathbf{x}_{T-2} + \mathbf{x}_{T-1}) \\
&= \mathbf{F}_{T-1}\mathbf{F}_{T-2}\mathbf{F}_{T-3}\mathbf{w}_{T-3} \\
&\qquad + \alpha G(\mathbf{F}_{T-1}\mathbf{F}_{T-2}\mathbf{x}_{T-3} + \mathbf{F}_{T-1}\mathbf{x}_{T-2} + \mathbf{x}_{T-1}) \\
&\quad \cdots \\
&= \underbrace{\mathbf{F}_{T-1}\mathbf{F}_{T-2}\ldots\mathbf{F}_0\mathbf{w}_0}_{a_{T-1}} + \alpha G \underbrace{\sum_{k=0}^{T-1} \mathbf{F}_{T-1}\mathbf{F}_{T-2}\cdots\mathbf{F}_{k+1}\mathbf{x}_k}_{\mathbf{e}_{T-1}} \\
&= \mathbf{a}_{T-1} + \alpha G \mathbf{e}_{T-1}
\end{aligned}
$$

where $\mathbf{a}_{T-1}$ and $\mathbf{e}_{T-1}$ are the values at time $T-1$ of two auxiliary memory vectors that can be updated incrementally without knowledge of G.

Dutch traces in MC

$$
\begin{aligned}
\mathbf{e} &\triangleq \sum_{k=0}^{t} \mathbf{F}_t \mathbf{F}_{t-1} \cdots \mathbf{F}_{k+1} \mathbf{x}_k, \qquad 1 \leq t < T \\
&= \sum_{k=0}^{t-1} \mathbf{F}_t \mathbf{F}_{t-1} \cdots \mathbf{F}_{k+1} \mathbf{x}_k + \alpha \mathbf{x}_t \\
&= \mathbf{F}_t \sum_{k=0}^{t-1} \mathbf{F}_{t-1} \mathbf{F}_{t-2} \cdots \mathbf{F}_{k+1} \mathbf{x}_k + \alpha \mathbf{x}_t \\
&= \mathbf{F}_t \mathbf{e}_{t-1} + \mathbf{x}_t \\
&= \left(\mathbf{I} - \alpha \mathbf{x}_t \mathbf{x}_t^T\right) \mathbf{e}_{t-1} + \mathbf{x}_t \\
&= \mathbf{e}_{t-1} - \alpha \mathbf{x}_t \mathbf{x}_t^T \mathbf{e}_{t-1} + \mathbf{x}_t \\
&= \mathbf{e}_{t-1} - \alpha (\mathbf{e}_{t-1}^T \mathbf{x}_t) \mathbf{x}_t + \mathbf{x}_t \\
&= \mathbf{e}_{t-1} + (1 - \alpha \mathbf{e}_{t-1}^T \mathbf{x}_t) \mathbf{x}_t
\end{aligned}
$$

Dutch traces in MC

Hence,

$$\mathbf{a}_t \triangleq \mathbf{F}_t \mathbf{F}_{t-1} \cdots \mathbf{F}_0 \mathbf{w}_0 = \mathbf{F}_t \mathbf{a}_{t-1} = \mathbf{a}_{t-1} - \alpha \mathbf{x}_t \mathbf{x}_t^T \mathbf{a}_{t-1}, \quad 1 \leq t < T$$

- The auxiliary vectors $\mathbf{a}_t$ and $\mathbf{e}_t$ are updated on each time step $t < T$ and then at time T when G is observed, they are used to compute $\mathbf{w}_T$.
- Note that the notion of an eligibility trace has arisen without TD. Hence, this is a more fundamental concept.

The 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: D. Silver, Slides at UCL

Sarsa(λ) backup diagram

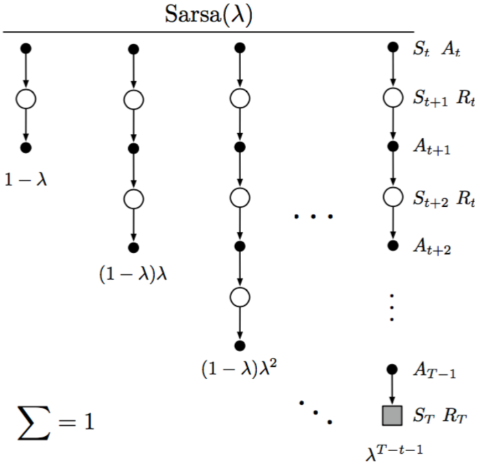

Figure: R. Sutton and A. Barto, MIT Press, 2017

Gridworld with Sarsa(λ)

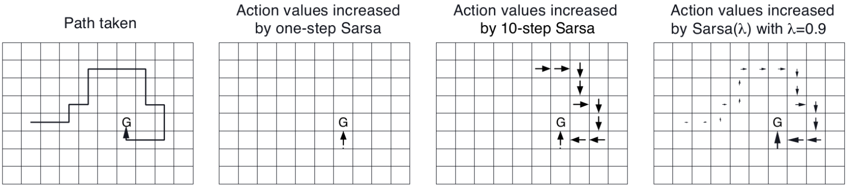

Figure: R. Sutton and A. Barto, MIT Press, 2017

True online TD(λ)

True Online TD(λ) for estimating $\mathbf{w}^\top \mathbf{x} \approx v_\pi$

Input: the policy π to be evaluated

Initialize value-function weights $\mathbf{w}$ arbitrarily (e.g., $\mathbf{w} = \mathbf{0}$)
Repeat (for each episode):
  Initialize state and obtain initial feature vector $\mathbf{x}$
  $\mathbf{e} \leftarrow \mathbf{0}$ (an n-dimensional vector)
  $V_{old} \leftarrow 0$ (a scalar temporary variable)
  Repeat (for each step of episode):
  | Choose $A \sim \pi$
  | Take action A, observe R, $\mathbf{x}'$ (feature vector of the next state)
  | $V \leftarrow \mathbf{w}^\top \mathbf{x}$
  | $V' \leftarrow \mathbf{w}^\top \mathbf{x}'$
  | $\delta \leftarrow R + \gamma V' - V$
  | $\mathbf{e} \leftarrow \gamma\lambda\mathbf{e} + \left(1 - \alpha\gamma\lambda\mathbf{e}^\top\mathbf{x}\right)\mathbf{x}$
  | $\mathbf{w} \leftarrow \mathbf{w} + \alpha(\delta + V - V_{old})\mathbf{e} - \alpha(V - V_{old})\mathbf{x}$
  | $V_{old} \leftarrow V'$
  | $\mathbf{x} \leftarrow \mathbf{x}'$
  until $\mathbf{x}' = \mathbf{0}$ (signaling arrival at a terminal state)

Figure: R. Sutton and A. Barto, MIT Press, 2017

True online Sarsa(λ)

True Online Sarsa(λ) for estimating $\mathbf{w}^\top\mathbf{x} \approx q_\pi$ or q_*

Input: a feature function $\mathbf{x} : \mathcal{S}^+ \times \mathcal{A} \to \mathbb{R}^d$ s.t. $\mathbf{x}(terminal, \cdot) = \mathbf{0}$
Input: the policy π to be evaluated, if any

Initialize parameter $\mathbf{w}$ arbitrarily (e.g., $\mathbf{w} = \mathbf{0}$)
Loop for each episode:
  Initialize S
  Choose $A \sim \pi(\cdot|S)$ or near greedily from S using $\mathbf{w}$; $\mathbf{x} \leftarrow \mathbf{x}(S, A)$
  $\mathbf{e} \leftarrow \mathbf{0}$
  $Q_{old} \leftarrow 0$ (a scalar temporary variable)
  Loop for each step of episode:
  | Take action A, observe R, S'
  | Choose $A' \sim \pi(\cdot|S')$ or near greedily from S' using $\mathbf{w}$; $\mathbf{x}' \leftarrow \mathbf{x}(S', A')$
  | $Q \leftarrow \mathbf{w}^\top\mathbf{x}$
  | $Q' \leftarrow \mathbf{w}^\top\mathbf{x}'$
  | $\delta \leftarrow R + \gamma Q' - Q$
  | $\mathbf{e} \leftarrow \gamma\lambda\mathbf{e} + \left(1 - \alpha\gamma\lambda\mathbf{e}^\top\mathbf{x}\right)\mathbf{x}$
  | $\mathbf{w} \leftarrow \mathbf{w} + \alpha(\delta + Q - Q_{old})\mathbf{e} - \alpha(Q - Q_{old})\mathbf{x}$
  | $Q_{old} \leftarrow Q'$
  | $\mathbf{x} \leftarrow \mathbf{x}'$
  | $A \leftarrow A'$
  until S' is terminal

Figure: R. Sutton and A. Barto, MIT Press, 2017

Mountain car example

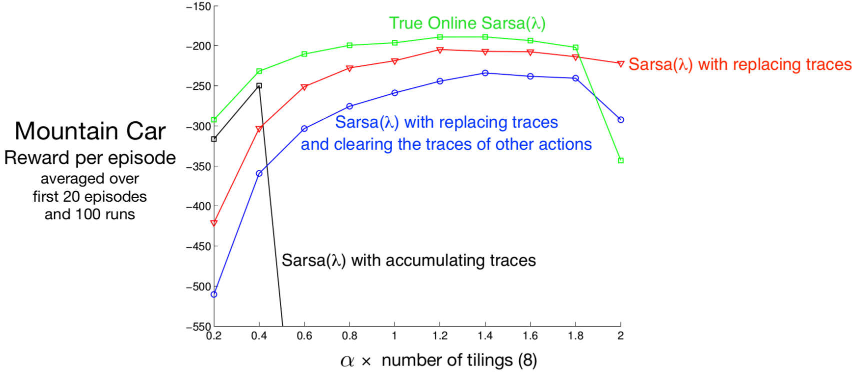

Figure: R. Sutton and A. Barto, MIT Press, 2017

Forward-backward TD

Offline updates	$\lambda = 0$	$\lambda \in (0,1)$	$\lambda = 1$
Backward view	TD(0)	TD(λ)	TD(1)
	$\parallel$	$\parallel$	$\parallel$
Forward view	TD(0)	Forward TD(λ)	MC
Online updates	$\lambda = 0$	$\lambda \in (0,1)$	$\lambda = 1$
Backward view	TD(0)	TD(λ)	TD(1)
	$\parallel$	$\nparallel$	$\nparallel$
Forward view	TD(0)	Forward TD(λ)	MC
	$\parallel$	$\parallel$	$\parallel$
Exact Online	TD(0)	Exact Online TD(λ)	Exact Online TD(1)

Figure: D. Silver, Slides at UCL