Eligibility traces

Corrado Possieri

Machine and Reinforcement Learning in Control Applications

Introduction

- An eligibility trace is a record of the occurrence of an event
  - tracks the eligibility of undergoing a learning event;
  - help bridge the gap between events and training information.
- More general method that may learn more efficiently.
- Bridge from TD to Monte Carlo methods.

$$\text{one-step TD} \quad \overset{0}{\longleftarrow}\!\!\overset{\lambda}{\longrightarrow}\!\overset{1}{} \quad \text{MC}$$

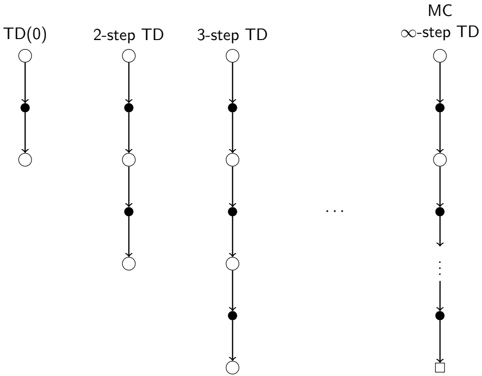
n-step methods

- With one-step TD methods the same time step
  - determines how often the action can be changed;
  - the time interval over which bootstrapping is done.

- What if we bootstrap over multiple steps?

n-step TD prediction

- MC performs updates based on the entire sequence of rewards.
- TD(0) is just based on the next reward and it bootstraps
  - value of next state is used as a proxy for future rewards.



n-step target

- MC target is the complete return

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots + \gamma^{T-t-1} R_T.$$

- 1-step TD target is the one step return

$$G_{t:t+1} = R_{t+1} + \gamma \underbrace{V_t(S_{t+1})}_{\text{estimate of } G_{t+1}} .$$

- 2-step TD target is the one step return

$$G_{t:t+2} = R_{t+1} + \gamma R_{t+2} + \gamma^2 \underbrace{V_{t+1}(S_{t+2})}_{\text{estimate of } G_{t+2}} .$$

- n-step TD target is the one step return

$$G_{t:t+n} = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{n-1} R_{t+n} + \gamma^n \underbrace{V_{t+n-1}(S_{t+n})}_{\text{estimate of } G_{t+n}}.$$

Future rewards

- If $t + n \geqslant T$, then all the missing terms are taken as zero

$$G_{t:t+n} = G_t, \text{ if } t + n \geqslant T.$$

- n-step update uses future rewards and states.
- Must wait until $t + n$ to see R_{t+n} and compute V_{t+n}.
- The natural learning algorithm is

$$\begin{aligned} V_{t+n}(S_t) &\leftarrow V_{t+n-1}(S_t) + \alpha(G_{t:t+n} - V_{t+n-1}(S_t)), \\ V_{t+n}(s) &\leftarrow V_{t+n-1}(s), \quad \forall s \neq S_t. \end{aligned}$$

n-step TD for estimating v_π

n-step TD prediction algorithm

Input: $\alpha > 0$, a positive integer n, a policy π
Output: v_π

Initialization

$V(s) \leftarrow$ arbitrary, $\forall s \in \mathcal{S}$
$V(\text{terminal}) \leftarrow 0$

Loop

initialize $S_0 \neq$ terminal
$T \leftarrow \infty$
for $t = 0, 1, 2, \ldots$ **do**
    take an action according to $\pi(\cdot|S)$
    observe and store R_{t+1} and S_{t+1}
    if S_{t+1} is terminal **then**
        $T \leftarrow t + 1$
    $\tau = t - n + 1$
    if $\tau \geq 0$ **then**
        $G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n,T)} \gamma^{i-\tau-1} R_i$
        if $\tau + n < T$ **then**
            $G \leftarrow G + \gamma^n V(S_{\tau+n})$
        $V(S_\tau) \leftarrow V(S_\tau) + \alpha(G - V(S_\tau))$
    if $\tau = T - 1$ **then**
        proceed to next episode

Error reduction property

- Expectation is a better estimate of v_π than V_{t+n-1}

$$\max_s \left|\mathbb{E}_\pi[G_{t:t+n}|S_t = s] - v_\pi(s)\right| \leqslant \max_s \gamma^n \left|V_{t+n-1}(s) - v_\pi(s)\right|.$$

- n-step TD methods converge to the correct predictions.

n-step SARSA

- n-step returns can be framed in terms of action values

$$G_{t:t+n} = R_{t+1} + \cdots + \gamma^{n-1} R_{t+n} + \gamma^n \underbrace{Q_{t+n-1}(S_{t+n}, A_{t+n})}_{\text{estimate of } G_{t+n}}.$$

- The natural learning algorithm is

$$\begin{aligned} Q_{t+n}(S_t, A_t) &\leftarrow Q_{t+n-1}(S_t, A_t) + \alpha(G_{t:t+n} - Q_{t+n-1}(S_t, A_t)), \\ Q_{t+n}(s, a) &\leftarrow Q_{t+n-1}(s, a), \quad \forall s \neq S_t, \forall a \neq A_t. \end{aligned}$$

- A_t selected according to an ε-greedy policy on Q.

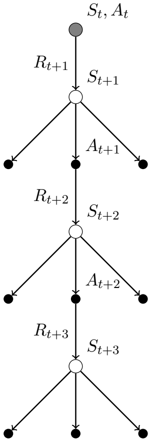
- Using importance sampling we obtain an off-policy algorithm.

Tree backups

- Avoid importance sampling.
- Consider actions that have not been selected.
- The update is from the leaf nodes of the tree.
- The tree-backup n-step return is

$$G_{t:t+n} = R_{t+1} + \gamma \sum_{a \neq A_{t+1}} \pi(a|S_{t+1}) Q(S_{t+1}, a) + \gamma \pi(A_{t+1}|S_{t+1}) G_{t+1:t+n}.$$

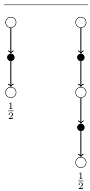
- n-step TD and tree backups can be combined as in $Q(\sigma)$.



Average of n-step returns

- The target can be selected averaging n-step returns

$$\frac{1}{2}\underbrace{G_{t:t+1}}_{\text{1-step return}}+\frac{1}{2}\underbrace{G_{t:t+2}}_{\text{2-step return}}.$$

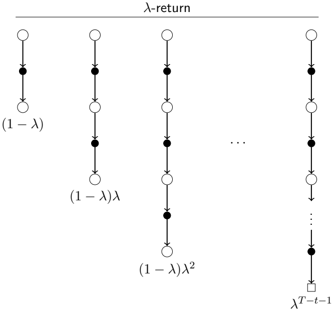


- As long as weights are non-negative and sum to 1
  - we have an error reduction property.

The λ-return

- The TD(λ) algorithm computes averages on n-step backups
  - the n-step backup is weighted by $(1-\lambda)\lambda^{n-1}$;
  - the corresponding λ-return is

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



On-line vs off-line updates

- Let $\Delta_t(s)$ be the update to be carried out.
- In on-line updating, we have

$$V_{t+1}(s) = V_t(s) + \Delta_t(s).$$

- In off-line updating, we have

$$\begin{aligned} V_{t+1}(s) &= V_t(s), \quad \forall t < T \\ V_T(s) &= V_{T-1}(s) + \sum_{t=0}^{T} \Delta_t(s). \end{aligned}$$

The forward TD(λ) algorithm

- For episodic tasks, we have

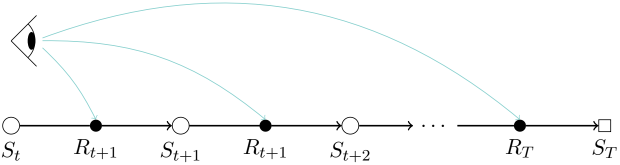
$$G_t^\lambda = (1-\lambda)\sum_{n=1}^{T-t-1} \lambda^{n-1} G_{t:t+n} + \lambda^{T-t-1} G_t.$$

- For $\lambda = 1$, we obtain an MC algorithm.
- For $\lambda = 0$, we obtain an one-step TD algorithm.
- The natural *off-line* forward TD(λ) learning algorithm is

$$\Delta_t(S_t) = \alpha(G_t^\lambda - V_t(S_t)),$$
$$\Delta_t(s) = 0, \quad \forall s \neq S_t.$$

Forward view of TD(λ)

- V is not changed until the end of the episode.
- At the end of the episode, we compute G_t^λ and make updates.
- For each state visited, we look forward in time to all the future rewards
  - future states are processed repeatedly;
  - we never look back;
  - we can truncate after h steps (truncated TD(λ)).



Backward view of TD(λ)

- The forward view is not implementable
  - acausal.
- The **backward view** provides a causal, incremental mechanism for approximating the forward view.
- In the off-line case it achieves the forward view exactly.

Eligibility trace

- Add an additional memory (random) variable for each state

$$E_t(s) \in \mathbb{R}^+.$$

- At each step, the eligibility trace of non-visited states decays

$$E_{t+1}(s) = \gamma\lambda E_t(s), \quad \forall s \neq S_t.$$

- The eligibility trace of S_t is additionally incremented by 1

$$E_{t+1}(S_t) = \gamma\lambda E_t(S_t) + 1.$$

- Eligibility traces keep a simple record of visited states
  - indicates the degree of eligibility of a learning event.

The TD(λ) algorithm

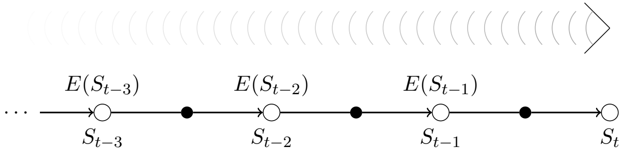
- The TD error for state-value prediction is

$$\delta_t = R_{t+1} + \gamma V_t(S_{t+1}) - V_t(S_t).$$

- In the backward view, updates are proportional to $E_t(s)$

$$\Delta_t(s) = \alpha \delta_t E_t(s), \quad \forall s \in \mathcal{S}.$$

- These increments can be bone both on-line and off-line.
- The TD error is streamed to the previously visited states.



On-line TD(λ)

On-line TD(λ) prediction algorithm

Input: $\alpha > 0$, $\lambda > 0$, a policy π
Output: v_π

Initialization

$V(s) \leftarrow \text{arbitrary}, \forall s \in \mathcal{S}$
$V(\text{terminal}) \leftarrow 0$

Loop

$E(s) \leftarrow 0, \forall s \in \mathcal{S}$
initialize S
repeat
  $A \leftarrow \pi(\cdot|S)$
  take action A and observe R and S'
  $\delta \leftarrow R + \gamma V(S') - V(S)$
  $E(S) \leftarrow E(S) + 1$
  for all $s \in \mathcal{S}$ **do**
    $V(s) \leftarrow V(s) + \alpha\delta E(s)$
    $E(s) \leftarrow \gamma\lambda E(s)$
  $S \leftarrow S'$
until S is terminal

Notes on the backward view of TD(λ)

- If $\lambda = 0$, then $E(s) = 0$ for all $s \neq S_t$ and $E(S_t) = 1$
  - one-step TD update TD(0).
- if $0 < \lambda < 1$, more preceding states are changed
  - temporally distant states are changed less (have less *credit*).
- If $\lambda = 1$, the credit falls only by γ per step
  - passing R_{t+1} back k steps discounts it by γ^k;
  - this is exactly the same as in MC methods;
  - TD(1) is a more general MC method
    - can be used for continuing tasks;
    - learn during the episode, not at its end;
    - can be implemented on-line.

Alternative updates of eligibility traces

- If a state is revisited before its trace go to zero, with *accumulating traces* its eligibility can become greater than 1.

- *Replacing trace* avoids this problem
  - each time a state is visited, its trace is reset to 1,

$$E_t(S_t) = 1.$$

- *Dutch trace* is an intermediate between the two

$$E_t(S_t) = (1 - \alpha)\gamma\lambda E_{t-1}(S_t) + 1$$

  - for $\alpha = 0$ it is the accumulating trace;
  - for $\alpha = 1$ it is the replacing trace.

SARSA(λ)

- Apply the TD(λ) prediction method to state–action pairs.
- The TD error for state-value prediction is

$$\delta_t = R_{t+1} + \gamma Q_t(S_{t+1}, A_{t+1}) - Q_t(S_t, A_t).$$

- Traces $E_t(s, a)$ for state-action pairs
  - accumulating;
  - dutch;
  - replacing.
- The updates are

$$Q_{t+1}(s, a) = Q_t(s, a) + \alpha\delta_t E_t(s, a), \quad \forall s, a.$$

SARSA(λ) algorithm

SARSA(λ) algorithm

Input: $\alpha > 0$, $\lambda > 0$
Output: q_*, π_*

Initialization

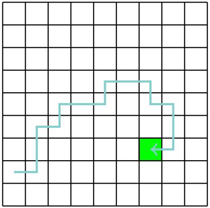
$Q(s, a) \leftarrow$ arbitrary, $\forall a \in \mathcal{A}(s), \forall s \in \mathcal{S}$
$Q(\text{terminal}, \cdot) \leftarrow 0$

Loop

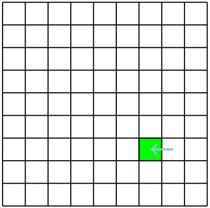
$E(s, a) \leftarrow 0, \forall a \in \mathcal{A}(s), \forall s \in \mathcal{S}$
initialize S
$A \leftarrow$action derived by $Q(S, \cdot)$ (e.g., ε-greedy)
for each step of the episode **do**
    take action A and observe R, S'
    $A' \leftarrow$action derived by $Q(S', \cdot)$ (e.g., ε-greedy)
    $\delta \leftarrow R + \gamma Q(S', A') - Q(S, A)$
    $E(S, A) \leftarrow (1 - \alpha)E(S, A) + 1$ (dutch trace)
    for all $s \in \mathcal{S}$ and all $a \in \mathcal{A}(s)$ **do**
        $Q(s, a) \leftarrow Q(s, a) + \alpha\delta E(s, a)$
        $E(s, a) \leftarrow \gamma\lambda E(s, a)$
    $S \leftarrow S'$
    $A \leftarrow A'$
    if S is terminal **then**
        reinitialize the episode

Advantages of SARSA(λ)

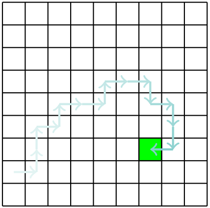
Path taken



Learnt from SARSA(0)



Learnt from SARSA(λ)



Q(λ)

- SARSA(λ) is on-policy.
- We also want an off-policy method
  - in learning about the value of the greedy policy
    - we can use subsequent experience as long as it is followed;
    - if A_{t+n} is the first exploratory action, the longest backup is

$$R_{t+1} + \cdots + \gamma^{n-1} R_{t+n} + \gamma^n \max_a Q(S_{t+n}, a)$$

- Q(λ) looks ahead up to the next exploratory action.
- The update works as in SARSA(λ)
  - traces are zeroed if an exploratory action is taken.

Q(λ) algorithm

Q(λ) algorithm

Input: $\alpha > 0$, $\lambda > 0$
Output: q_*, π_*

Initialization

$Q(s, a) \leftarrow$ arbitrary, $\forall a \in \mathcal{A}(s), \forall s \in \mathcal{S}$; $Q(\text{terminal}, \cdot) \leftarrow 0$

Loop

```
E(s, a) ← 0, ∀a ∈ A(s), ∀s ∈ S
initialize S
A ←action derived by Q(S, ·) (e.g., ε-greedy)
for each step of the episode do
    take action A and observe R, S'
    A' ←action derived by Q(S', ·) (e.g., ε-greedy)
    A* ← arg max_a Q(S', a)
    δ ← R + γQ(S', A*) − Q(S, A)
    E(S, A) ← (1 − α)E(S, A) + 1 (dutch trace)
    for all s ∈ S and all a ∈ A(s) do
        Q(s, a) ← Q(s, a) + αδE(s, a)
        if A' = A* then
            E(s, a) ← γλE(s, a)
        else
            E(s, a) ← 0
    S ← S'
    A ← A'
    if S is terminal then
        reinitialize the episode
```

Notes on Q(λ)

- Cutting traces loses the advantage of eligibility traces.
- Learning is slow.
- Learning will be litter faster than classical Q-learning.

Notes on TD(λ) methods

- Seem to be much more complex than one-step TD
  - every state has to be updated.
- Traces of almost all states are almost always nearly zero
  - few states really need to be updated.
- The parameter λ can be made a function of S_t
  - if a state's value is believed to be known with high certainty
    - it is reasonable to cut the traces, $\lambda \to 0$;
  - if a state's value is highly uncertain
    - it is reasonable to update it more often, $\lambda \to 1$.