The story of eligibility traces with the proof's !

Pierre Thodoroff

April 7, 2017

Outline

1. Backward = Forward ??
   - Background
   - Proof's
   - So what's the problem with TD(λ)
   - True online TD
   - Proof's

2. Eligibility traces and RNN
   - Exact Gradient
   - Approximate Gradient

The Lambda Return

- N-step returns:

$$G_t^{(n)} \doteq R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{n-1} R_{t+n} + \gamma^n \hat{v}(S_{t+n}, \boldsymbol{\theta}_{t+n-1}),$$

- λ -return:

$$G_t^{\lambda} \doteq (1-\lambda) \sum_{n=1}^{\infty} \lambda^{n-1} G_t^{(n)}.$$

Offline lambda return

- At the end of each episode we make our updates according to the following scheme:

$$\boldsymbol{\theta}_{t+1} \doteq \boldsymbol{\theta}_t + \alpha\Big[G_t^{\lambda} - \hat{v}(S_t,\boldsymbol{\theta}_t)\Big]\nabla\hat{v}(S_t,\boldsymbol{\theta}_t), \quad t = 0,\ldots,T-1.$$

- It is considered offline because the update are only made at the end of the episode

TD(λ)

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

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

Initialize value-function weights $\boldsymbol{\theta}$ arbitrarily (e.g., $\boldsymbol{\theta} = \mathbf{0}$)
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'
    . $\mathbf{e} \leftarrow \gamma\lambda\mathbf{e} + \nabla\hat{v}(S,\boldsymbol{\theta})$
    . $\delta \leftarrow R + \gamma\hat{v}(S',\boldsymbol{\theta}) - \hat{v}(S,\boldsymbol{\theta})$
    . $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha\delta\mathbf{e}$
    . $S \leftarrow S'$
    until S' is terminal

TD(λ)==λ-return ?

$$\sum_{t=0}^{T-1} \Delta V_t^{TD}(s) = \sum_{t=0}^{T-1} \Delta V_t^{\lambda}(s_t)\mathcal{I}_{ss_t}, \qquad \text{for all } s \in \mathcal{S},$$

We want to show that if we only update the parameters at the end then TD(λ) performs the same update than offline λ-return.

The assumption that we do not change the parameter during the trajectory is essential.

Proof on the board !

TD(λ)

$$e_t(s) = \sum_{k=0}^{t} (\gamma\lambda)^{t-k} \mathcal{I}_{ss_k}.$$

$$\begin{aligned}
\sum_{t=0}^{T-1} \Delta V_t^{TD}(s) &= \sum_{t=0}^{T-1} \alpha \delta_t \sum_{k=0}^{t} (\gamma\lambda)^{t-k} \mathcal{I}_{ss_k} && (7.9)\\
&= \sum_{k=0}^{T-1} \alpha \sum_{t=0}^{k} (\gamma\lambda)^{k-t} \mathcal{I}_{ss_t} \delta_k && (7.10)\\
&= \sum_{t=0}^{T-1} \alpha \sum_{k=t}^{T-1} (\gamma\lambda)^{k-t} \mathcal{I}_{ss_t} \delta_k && (7.11)\\
&= \sum_{t=0}^{T-1} \alpha \mathcal{I}_{ss_t} \sum_{k=t}^{T-1} (\gamma\lambda)^{k-t} \delta_k. && (7.12)
\end{aligned}$$

λ-return

$$
\begin{aligned}
\frac{1}{\alpha}\Delta V_t^{\lambda}(s_t) &= R_t^{\lambda} - V_t(s_t) \\
&= -V_t(s_t) + (1-\lambda)\lambda^0[r_{t+1} + \gamma V_t(s_{t+1})] \\
&\qquad + (1-\lambda)\lambda^1[r_{t+1} + \gamma r_{t+2} + \gamma^2 V_t(s_{t+2})] \\
&\qquad + (1-\lambda)\lambda^2[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \gamma^3 V_t(s_{t+3})] \\
&\qquad\qquad \vdots \qquad \vdots \qquad \vdots \qquad \ddots
\end{aligned}
$$

λ-return

$$
\begin{aligned}
\frac{1}{\alpha}\Delta V_t^{\lambda}(s_t) &= -V_t(s_t) \\
&\quad + (\gamma\lambda)^0 \left[r_{t+1} + \gamma V_t(s_{t+1}) - \gamma\lambda V_t(s_{t+1})\right] \\
&\quad + (\gamma\lambda)^1 \left[r_{t+2} + \gamma V_t(s_{t+2}) - \gamma\lambda V_t(s_{t+2})\right] \\
&\quad + (\gamma\lambda)^2 \left[r_{t+3} + \gamma V_t(s_{t+3}) - \gamma\lambda V_t(s_{t+3})\right] \\
&\quad \vdots \\
&= \quad (\gamma\lambda)^0 \left[r_{t+1} + \gamma V_t(s_{t+1}) - V_t(s_t)\right] \\
&\quad + (\gamma\lambda)^1 \left[r_{t+2} + \gamma V_t(s_{t+2}) - V_t(s_{t+1})\right] \\
&\quad + (\gamma\lambda)^2 \left[r_{t+3} + \gamma V_t(s_{t+3}) - V_t(s_{t+2})\right] \\
&\quad \vdots \\
&\approx \sum_{k=t}^{\infty} (\gamma\lambda)^{k-t}\delta_k \\
&\approx \sum_{k=t}^{T-1} (\gamma\lambda)^{k-t}\delta_k.
\end{aligned}
$$

Problems with TD(λ)

- If the weights are changed during training than TD(λ) is approximate
- In the RL case this can have a big impact because big weight change can impact the policy behavior
- Maybe it is possible to correct for this difference ?? the answer is true online TD

True online TD

- True online TD first modify the return suc that it can be used online
$$G_t^{\lambda|h} \doteq (1-\lambda)\sum_{n=1}^{h-t-1}\lambda^{n-1}G_t^{(n)} + \lambda^{h-t-1}G_t^{(h-t)},$$

- You now need to restart the update from scratch at every time step though unless there is a clever way to correct for the weights change of the past !!

True online TD

$$h = 1: \quad \boldsymbol{\theta}_1^1 \doteq \boldsymbol{\theta}_0^1 + \alpha \left[G_0^{\lambda|1} - \hat{v}(S_0, \boldsymbol{\theta}_0^1) \right] \nabla \hat{v}(S_0, \boldsymbol{\theta}_0^1),$$

$$h = 2: \quad \boldsymbol{\theta}_1^2 \doteq \boldsymbol{\theta}_0^2 + \alpha \left[G_0^{\lambda|2} - \hat{v}(S_0, \boldsymbol{\theta}_0^2) \right] \nabla \hat{v}(S_0, \boldsymbol{\theta}_0^2),$$

$$\boldsymbol{\theta}_2^2 \doteq \boldsymbol{\theta}_1^2 + \alpha \left[G_1^{\lambda|2} - \hat{v}(S_1, \boldsymbol{\theta}_1^2) \right] \nabla \hat{v}(S_1, \boldsymbol{\theta}_1^2),$$

$$h = 3: \quad \boldsymbol{\theta}_1^3 \doteq \boldsymbol{\theta}_0^3 + \alpha \left[G_0^{\lambda|3} - \hat{v}(S_0, \boldsymbol{\theta}_0^3) \right] \nabla \hat{v}(S_0, \boldsymbol{\theta}_0^3),$$

$$\boldsymbol{\theta}_2^3 \doteq \boldsymbol{\theta}_1^3 + \alpha \left[G_1^{\lambda|3} - \hat{v}(S_1, \boldsymbol{\theta}_1^3) \right] \nabla \hat{v}(S_1, \boldsymbol{\theta}_1^3),$$

$$\boldsymbol{\theta}_3^3 \doteq \boldsymbol{\theta}_2^3 + \alpha \left[G_2^{\lambda|3} - \hat{v}(S_2, \boldsymbol{\theta}_2^3) \right] \nabla \hat{v}(S_2, \boldsymbol{\theta}_2^3).$$

True online TD

True Online TD(λ) for estimating $\theta^\top \phi \approx v_\pi$

Input: the policy π to be evaluated

Initialize value-function weights $\boldsymbol{\theta}$ arbitrarily (e.g., $\boldsymbol{\theta} = \mathbf{0}$)
Repeat (for each episode):
  Initialize state and obtain initial feature vector $\boldsymbol{\phi}$
  $\mathbf{e} \leftarrow \mathbf{0}$ (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, $\boldsymbol{\phi}'$ (feature vector of the next state)
  . $V \leftarrow \boldsymbol{\theta}^\top \boldsymbol{\phi}$
  . $V' \leftarrow \boldsymbol{\theta}^\top \boldsymbol{\phi}'$
  . $\mathbf{e} \leftarrow \gamma\lambda\mathbf{e} + \left(1 - \alpha\gamma\lambda\mathbf{e}^\top\boldsymbol{\phi}\right)\boldsymbol{\phi}$
  . $\delta \leftarrow R + \gamma V' - V$
  . $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha(\delta + V - V_{old})\mathbf{e} - \alpha(V - V_{old})\boldsymbol{\phi}$
  . $V_{old} \leftarrow V'$
  . $\boldsymbol{\phi} \leftarrow \boldsymbol{\phi}'$
  until $\boldsymbol{\phi}' = \mathbf{0}$ (signaling arrival a terminal state)

True online TD

Proof Idea on the board if time permits !!

Connection between RNN and eligibility traces

- Conceptually a correspondence can be established between the gradient used in RNN to update the gradients and eligibility traces
- Update for TD(λ) : $\theta_{t+1} = \theta_t + \alpha \delta_t e_t$
- Update for RNN : $\theta_{t+1} = \theta_t + \alpha \frac{\partial L}{\partial h} \frac{\partial h}{\partial \theta}$

Exact Gradient

- Backpropagation through time can be seen as offline λ-return. It is an offline technique that modify's all the parameter at the end. It calculates the exact gradient
- The parameters are not changed during learning so the gradient is exact
- Using offline TD(λ) also yields the exact gradient so it is equivalent

Approximate Gradient

- Online TD(λ) is an approximate gradient algorithm in the sense that it uses an approximation of the gradient by making the assumptions the weights are fixed during training

- The same strategy is used in Real Time Recurrent Learning

- True online TD has found a strategy to correct the gradient estimate when changing the parameter during the learning by approximating a different but similar return. Maybe the math translate to RTRL ?? (still open problem)

For Further Reading I

Sutton and Barto.
Reinforcement Learning: An Introduction.
Draft 2017.

Williams, Zipser
Gradient-Based Learning Algorithms for Recurrent Networks and Their Computational Complexity
1995