Eligibility Traces

Tuesday, June 23, 2020 12:48 PM

Eligibility Trace:

$\vec{z}_t \in R^d$

A vector that parallels the weight vector $\vec{w}_t \in R^d$ through increasing a component of $\vec{z}_t$ when the corresponding component of $\vec{w}_t$ participates in producing an estimated value then begins to fade away over time

The name comes from the trace indicating the eligibility of each component to undergo learning changes should a reinforcing event occur

Trace Decay Parameter:

λ

This determines the rate at which the trace falls

Advantages of Eligibility Trace Methods:

Computational advantages over n-step methods since only a single vector $\vec{z}_t$ needs to be stored instead of the last n feature vectors

Learning occurs continually and uniformly in time instead of catching up after each episode or being delayed after a state is encountered by n-steps

Forward Views:

Algorithms that update a state based on the states after the state being updated

Backward Views:

Algorithms that update a state based on recently visited states

n-Step Return:

$$G_{t:t+n} \equiv \sum_{k=0}^{n-1} \gamma^k R_{t+k+1} + \gamma^n \hat{v}(S_{t+n}, \vec{w}_{t+n-1}) = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{n-1} R_{t+n} + \gamma^n \hat{v}(S_{t+n}, \vec{w}_{t+n-1})$$

Where $0 \leq t \leq T - n$ (T is the time of episode termination if the task is episodic)

Compound Update:

An update that averages simpler updates

Note that this can only be done when the longest of its component updates is done (e.g. doing a compound update of a 2-step and 4-step return has to wait until the 4-step return is available)

λ-Return:

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

Where $\lambda \in [0,1]$

This is an average of all n-step updates, each weighted proportionally to λ^{n-1} then normalized to ensure all weights sum to 1

Note that the weight fades by a factor of λ at each step

Note that the case of $\lambda = 1$ results in a MC method (the λ-return is just the conventional return) and the case of $\lambda = 0$ results in one-step TD (the λ-return is just $G_{t:t+1}$, the one-step return)

Note that this isn't known until the end of the episode for the end of time for the continuing case

Episodic Form:

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

Off-line λ-Return Algorithm:

$$\vec{w}_{t+1} \equiv \vec{w}_t + \alpha\left[G_t^\lambda - \hat{v}(S_t, \vec{w}_t)\right] \nabla \hat{v}(S_t, \vec{w}_t)$$

Semi-Gradient TD(λ):

$$\vec{z}_t \equiv \gamma \lambda \vec{z}_{t-1} + \nabla \hat{v}(S_t, \vec{w}_t)$$

$\delta_t \equiv R_{t+1} + \gamma \hat{v}(S_{t+1}, \vec{w}_t) - \hat{v}(S_t, \vec{w}_t)$
$\vec{w}_{t+1} \equiv \vec{w}_t + \alpha \delta_t \vec{z}_t$

This improves upon the off-line version since it can be done online, computations are spaced equally in time instead of just at the end of episodes, and it can be applied to continuing problems

Note that this is backward facing since each state is updated according to the eligibility trace, which is determined from past data

Note that this changes the entire weight vector, but the eligibility trace means that the temporally distant states are changed less

This can be stated as the more distant states being given less credit for the TD error

TD(1) is equivalent to MC methods but can be done online and in continuing tasks

TD(λ) Error Bound:

$$\overline{VE}(\vec{w}_\infty) \leq \frac{1-\gamma\lambda}{1-\gamma} \min_{\vec{w}} \overline{VE}(\vec{w})$$

This can be stated as the asymptotic error is no more than $\frac{1-\gamma\lambda}{1-\gamma}$ times the smallest possible error

Note that if $\lambda = 1$ then the asymptotic error will reach the minimum possible value

Truncated λ-Return:

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

Where $0 \leq t < h \leq T$ where h is called the horizon

This is a shortened version of the off-line λ-return algorithm

k-Step λ-Return:

$$G_{t:t+k}^{\lambda} \equiv \hat{v}(S_t, \vec{w}_{t-1}) + \sum_{i=t}^{t+k-1} (\gamma\lambda)^{i-t} \delta_i'$$

$\delta_i' \equiv R_{t+1} + \gamma \hat{v}(S_{t+1}, \vec{w}_t) - \hat{v}(S_t, \vec{w}_{t-1})$

This is equivalent to the n-step return seen before but using the λ-return and called k-step

Truncated TD(λ) (TTD(λ)):

$\vec{w}_{t+n} \equiv \vec{w}_{t+n-1} + \alpha [G_{t:t+n}^{\lambda} - \hat{v}(S_t, \vec{w}_{t+n-1})] \nabla \hat{v}(S_t, \vec{w}_{t+n-1})$

Online λ-Return Algorithm:

$\vec{w}_{t+1}^{h} \equiv \vec{w}_t^{h} + \alpha \left[G_{t:h}^{\lambda} - \hat{v}(S_t, \vec{w}_t^{h})\right] \nabla \hat{v}(S_t, \vec{w}_t^{h})$

Where $0 \leq t < h \leq T$ and $\vec{w}_t^h$ denotes the weights used at time t in the sequence up to horizon h and $\vec{w}_t \equiv \vec{w}_t^h$

Note that this passes over the portion of the episode experienced so far on every step

This is currently the best performing TD algorithm

True Online TD(λ):

$\vec{w}_{t+1} \equiv \vec{w}_t + \alpha \delta_t \vec{z}_t + \alpha (\vec{w}_t^T \vec{x}_t - \vec{w}_{t-1} \vec{x}_t)(\vec{z}_t - \vec{x}_t)$
$\vec{z}_t \equiv \gamma\lambda \vec{z}_{t-1} + (1 - \alpha\gamma\lambda \vec{z}_{t-1}^T \vec{x}_t) \vec{x}_t$

This produces exactly the same sequence of weight vectors as the online λ-return algorithm, but is much less computationally expensive (around 50%)

The eligibility trace used in this algorithm is called a Dutch trace while the eligibility trace used in TD(λ) is called an accumulating trace

Action-Value n-Step Return:

$$G_{t:t+n} \equiv \sum_{i=0}^{n-1} \gamma^i R_{t+i+1} + \gamma^n \hat{q}(S_{t+n}, A_{t+n}, \vec{w}_{t+n-1}) = R_{t+1} + \cdots \gamma^{n-1} R_{t+n} + \gamma^n \hat{q}(S_{t+n}, A_{t+n}, \vec{w}_{t+n-1})$$

Action-Value Off-Line λ-Return:

$\vec{w}_{t+1} \equiv \vec{w}_t + \alpha [G_t^{\lambda} - \hat{q}(S_t, A_t, \vec{w}_t)] \nabla \hat{q}(S_t, A_t, \vec{w}_t)$

Sarsa(λ):

$\vec{w}_{t+1} \equiv \vec{w}_t + \alpha \delta_t \vec{z}_t$
$\delta_t \equiv R_{t+1} + \gamma \hat{q}(S_{t+1}, A_{t+1}, \vec{w}_t) - \hat{q}(S_t, A_t, \vec{w}_t)$
$\vec{z}_t \equiv \gamma\lambda \vec{z}_{t-1} + \nabla \hat{q}(S_t, A_t, \vec{w}_t)$

Generally Defined Return:

$$G_t \equiv \sum_{k=t}^{\infty} \left(\prod_{i=t+1}^{k} \gamma_i \right) R_{k+1}$$

Where γ is the termination function, defined such that $\gamma_t \equiv \gamma(S_t)$ with the requirement that $\prod_{k=t}^{\infty} \gamma_k = 0$ for all t to ensure that the sum converges

Generally Defined State-Based λ-Return:

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

Where the superscript s simply denotes that this return is state-based (bootstraps from state values)

With Importance Sampling (for off-policy training):

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

Truncated Approximation:

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

$$\delta_i^s \equiv R_{t+1} + \gamma_{t+1} \hat{v}(S_{t+1}, \vec{w}_t) - \hat{v}(S_t, \vec{w}_t)$$

Generally Defined Action-Based λ-Return:

Sarsa Form:

$$G_t^{\lambda a} \equiv R_{t+1} + \gamma_{t+1} \left((1 - \lambda_{t+1}) \hat{q}(S_{t+1}, A_{t+1}, \vec{w}_t) + \lambda_{t+1} G_{t+1}^{\lambda a} \right)$$

Expected Sarsa Form:

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

$$\bar{V}_t(s) \equiv \sum_{a} \pi(a|s) \hat{q}(s, a, \vec{w}_t)$$

With Importance Sampling (for off-policy training):

$$G_t^{\lambda a} \equiv R_{t+1} + \gamma_{t+1} (\bar{V}_t(S_{t+1}) + \lambda_{t+1} \rho_{t+1} [G_{t+1}^{\lambda a} - \hat{q}(S_{t+1}, A_{t+1}, \vec{w}_t)])$$

Approximation:

$$G_t^{\lambda a} \approx \hat{q}(S_t, A_t, \vec{w}_t) + \sum_{k=t}^{\infty} \delta_k^a \prod_{i=t+1}^{k} \gamma_i \lambda_i \rho_i$$

$$\delta_t^a = R_{t+1} + \gamma_{t+1} \bar{V}_t(S_{t+1}) - \hat{q}(S_t, A_t, \vec{w}_t)$$

Generally Defined Eligibility Trace:

$$\vec{z}_t \equiv \rho_t \left(\gamma_t \lambda_t \vec{z}_{t-1} + \nabla \hat{v}(S_t, \vec{w}_t) \right)$$

Combining this with semi-gradient TD(λ) as defined above means that TD(λ) can be used for off-policy training as well

Eligibility Trace for Action Values:

$$\vec{z}_t \equiv \gamma_t \lambda_t \rho_t \vec{z}_{t-1} + \nabla \hat{q}(S_t, A_t, \vec{w}_t)$$

Watson's Q(λ):

A way of extending Q-learning to eligibility traces where eligibility traces are decayed in the usual way when greedy actions are taken then reset to 0 when non-greedy actions are taken

Eligibility Trace Tree Backup (TB(λ)):

$$G_t^{\lambda a} \equiv R_{t+1} + \gamma_{t+1} \left((1 - \lambda_{t+1}) \bar{V}_t(S_{t+1}) + \lambda_{t+1} \left[\sum_{a \neq A_t+1} \pi(a|S_{t+1}) \hat{q}(S_{t+1}, a, \vec{w}_t) + \pi(A_{t+1}|S_{t+1}) G_{t+1}^{\lambda a} \right] \right)$$

$$= R_{t+1} + \gamma_{t+1} \left(\bar{V}_t(S_{t+1}) + \lambda_{t+1} \pi(A_{t+1}|S_{t+1}) \left(G_{t+1}^{\lambda a} - \hat{q}(S_{t+1}, A_{t+1}, \vec{w}_t) \right) \right)$$

Approximation:

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

$\vec{z}_t \equiv \gamma_t \lambda_t \pi(A_t, S_t)\vec{z}_{t-1} + \nabla\hat{q}(S_t, A_t, \vec{w}_t)$

GTD(λ):

$\vec{w}_{t+1} \equiv \vec{w}_t + \alpha\delta_t^s \vec{z}_t - \alpha\gamma_{t+1}(1 - \lambda_{t+1})(\vec{z}_t^T \vec{v}_t)\vec{x}_{t+1}$

$\vec{v}_{t+1} \equiv \vec{v}_t + \beta\delta_t^s \vec{z}_t - \beta(\vec{v}_t^T \vec{x}_t)\vec{x}_t$

Where $\vec{v}_0 = 0$

This is analogous to TDC (off-policy state-value gradient TD method discussed in the previous chapter)

GQ(λ):

$\vec{w}_{t+1} \equiv \vec{w}_t + \alpha\delta_t^a \vec{z}_t - \alpha\gamma_{t+1}(1 - \lambda_{t+1})(\vec{z}_t^T \vec{v}_t)\bar{\vec{x}}_{t+1}$

$\bar{\vec{x}} \equiv \sum_a \pi(a|S_t)\vec{x}(S_t, a)$

$\delta_t^a \equiv R_{t+1} + \gamma_{t+1}\vec{w}_t^T \bar{\vec{x}}_{t+1} - \vec{w}_t^T \vec{x}_t$

$\bar{\vec{x}}_t$ is the average feature vector for S_t under the target policy and δ_t^a is the expectation form of the TD error

This is the Gradient-TD algorithm for action values with eligibility traces

If the target policy is biased toward the greedy policy then this can be used as a control algorithm

HTD(λ):

$\vec{w}_{t+1} \equiv \vec{w}_t + \alpha\delta_t^s \vec{z}_t + \alpha\left(\left(\vec{z}_t - \vec{z}_t^b\right)^T \vec{v}_t\right)(\vec{x}_t - \gamma_{t+1}\vec{x}_{t+1})$

$\vec{v}_{t+1} \equiv \vec{v}_t + \beta\delta_t^s \vec{z}_t - \beta\left({\vec{z}_t^b}^T \vec{v}_t\right)(\vec{x}_t - \gamma_{t+1}\vec{x}_{t+1})$

$\vec{z}_t \equiv \rho_t(\gamma_t \lambda_t \vec{z}_{t+1} + \vec{x}_t)$

$\vec{z}_t^b \equiv \gamma_t \lambda_t \vec{z}_{t-1}^b + \vec{x}_t$

Where $\vec{v}_0 = 0$, β is another step-size parameter, and $\vec{z}_t^b$ are a second set of eligibilty traces (these

This is a hybrid state-value algorithm that combines aspects of GTD(λ) and TD(λ)

This is a strict generalization of TD(λ) to off-policy learning, so if the behavior policy and target policy are the same then this becomes TD(λ)

Emphatic TD(λ):

$\vec{w}_{t+1} \equiv \vec{w}_t + \alpha\delta_t \vec{z}_t$

$\delta_t \equiv R_{t+1} + \gamma_{t+1}\vec{w}_t^T \vec{x}_{t+1} - \vec{w}_t^T \vec{x}_t$

$\vec{z}_t \equiv \rho_t(\gamma_t \lambda_t \vec{z}_{t-1} + M_t \vec{x}_t)$

$M_t \equiv \lambda_t I_t + (1 - \lambda_t)F_t$

$F_t \equiv \rho_{t-1}\gamma_t F_{t-1} + I_t$

Where $\vec{z}_{-1} \equiv 0$ and $F_0 \equiv i(S_0)$

F_t is called the followon trace