Eligibility traces

Corrado Possieri

Machine and Reinforcement Learning in Control Applications

Introduction

Servono a mantenere una memoria degli stati che ho scelto: una memoria degli stati che ho visitato e la probabilità di dover cambiare valore.
Permette di colmare il vuoto tra liaddestramento e l'apprendimento. Sono più efficienti poiché con un singolo aggiornamento se ne aggiornano molte altre. Quindi, l'apprendimento risulta molto più veloce.

- 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.

one-step TD $\quad 0 \xleftrightarrow{\quad\lambda\quad} 1 \quad$ MC

Permette di spostarsi tra l'apprendimento
di un singolo passo e fra tutti i passi,
Da monte carlo si passa alla differenze temporali
e viceversa.

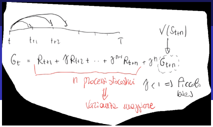
T

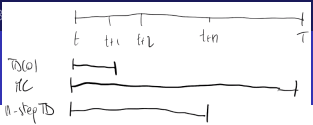
$$\underbrace{\gamma R_{t+2} + \dots + \gamma^{n-1} R_{t+n}}_{\text{in processi stocastici}} + \gamma$$

γ

$\Vert$

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.

stato corrente

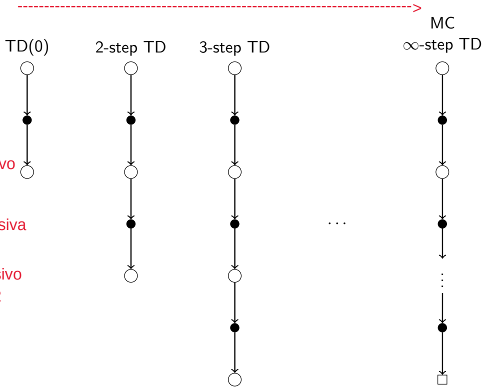
azione

stato successivo
gamma

azione successiva

stato successivo
gamma^2

...



La differenza è il lviello in cui si fa bootstrapping, quando utilizzo il valore per stimare il ritorno negli stati successivi.

Più scendo di livello meno sono influenzato dai valore precedenti

n-step target

- MC target is the complete return

all visit

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

- 1-step TD target is the one step return

N.B.:
- 0<gamma<1

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

dove parto

dove arrivo

- 2-step TD target is the one step return

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

- n-step TD target is the one step return

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

Dobbiamo aspettare di collezionare un numero sufficiente di reward per costruire la stime del ritorno ed effettuare l'aggiornamento. Nella prima parte, aspetto n campioni (non necessariamente la fne) dopodiche faccio l'aggiornamento.

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

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

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
$T \leftarrow \infty$ non so se il task termina o è continuativo
for $t = 0, 1, 2, \ldots$ **do**
    take an action according to $\pi(\cdot|S)$ Perché sto stimando il valore della policy pi
    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** aggiorno lo stato n passi fa
        $G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n,T)} \gamma^{i-\tau-1} R_i$
        if $\tau + n < T$ **then**
            $G \leftarrow G + \gamma^n V(S_{\tau+n})$ bootstrapping stima di tutto quello che segue per lo stato in cui sono arrivato non è terminale
aggiornamento $V(S_\tau) \leftarrow V(S_\tau) + \alpha(G - V(S_\tau))$
    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}

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

diminuisco sempre di più il preconcetto

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

n-step SARSA

CONTROLLO

- 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

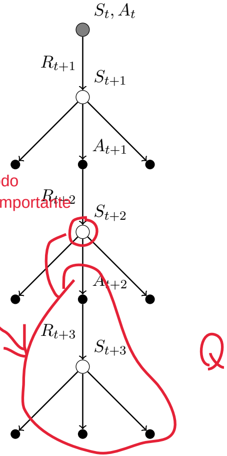
Mantengo un approccio probabilistico che prende in considerazione anche le azioni che non ho presto. Costruisco un albero rispetto alle coppia (s,a), dall'azione terminale,

- 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

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

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

Condenso le infomazini nel nodo superiore. Non ho bisogna di importante samplig



Deve essere visto al contrario, dal basso verso l'alto.

Average of n-step returns

Ho a disposizione tante TD ed utilizzo una media pesata del ritorno dei vari TD-step

- The target can be selected averaging n-step returns

Per il calcolo dei ritorni si procede in questa maniera: si parte dall'istante di partenza e si fa bootstrapping (t+n)

bootstrapping 1 passo

$G_{t:t+1} = R_{t+1} + \gamma V(S_{t+1})$

ist. di bootstrapping

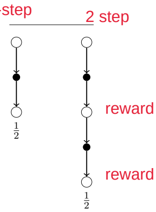
ist. di partenza

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

bootstrapping 2 passi

$G_{t:t+2} = R_{t+1} + \gamma R_{t+2} + \gamma^2 V(S_{t+2})$

Mediando tra questi due ritorni, posso ridurre il bias dovuto al preconcetto sulla funzione valore.



- As long as weights are non-negative and sum to 1 la somma dei pesi = 1
  - we have an error reduction property.

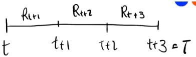
The λ-return

I metodi TD(lambad) effettua la somma pesata tramite lambda che determina quanto peso il futuro. Si fa la somma pesata di infiniti ritorni. Questo mi garantisce l'equivalenza della slide precedenti. Anche qui i pesi sommati devono fare 1,

- The TD(λ) algorithm computes averages on n-step backups
  - the n-step backup is weighted by $(1-\lambda)\lambda^{n-1}$; poiché la sommatoria deve tornare 1/(1-lamda)*lambda =1
  - the corresponding λ-return is

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

Nel caso in cui il task è episodico si assegna all'episodio terminare lambda^(T-t-1) assegnando ai precedente i pesi

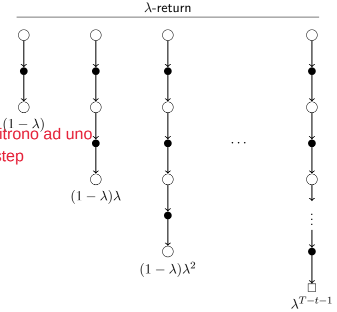


$G_{t:t+1} = R_{t+1} + \gamma V(S_{t+1})$

$G_{t:t+2} = R_{t+1} + \gamma R_{t+2} + \gamma^2 V(S_{t+2})$

$G_{t:t+3} = G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3}$

lambda = 1->montecarlo
lambda=0->TD(0)



ritrono ad uno step

ritorno che utilizzerei in MC

off-line solo nel caso episodico.

On-line vs off-line updates

Vogliamo effettuare un aggiornamento

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

aggiorno direttamente la stima della funzione valore, appena calcolo l'incremento delta.
Ne sono un esempio il SARSA e il Q-learning appena calcolo la stima ed effettuare gli aggiornamento

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

- In off-line updating, we have

Guardo tutti gli aggiornamenti e non cambio la funzione valore.
Una volta che sono allo stato terminale effettuo tutti gli aggiornamenti necessari.

$$V_{t+1}(s) = V_t(s), \quad \forall t < T$$

Ne sono un esempio i metodi di tipo batch [andare a rivedere].

$$V_T(s) = V_{T-1}(s) + \sum_{t=0}^{T} \Delta_t(s).$$

The forward TD(λ) algorithm

Algoritmo off-line
Siccome il ritorno è definito sulla base di un oggetto che potrebbe essere infinito

- For episodic tasks, we have

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

Calcolato in ogni istante di tempo.

- For $\lambda = 1$, we obtain an MC algorithm. i ritorni intermedi non son considerati, ma guardo solo quello dell'ultimo episodio
- For $\lambda = 0$, we obtain an one-step TD algorithm. utilizzo solo il ritorno ad una differenza temporale
- The natural *off-line* forward TD(λ) learning algorithm is

stima del ritorno — stima vecchia del ritorno

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

aggiornamento solo su uno stato

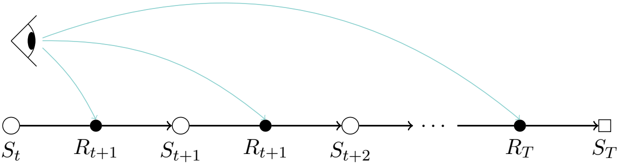
Lo posso implementare solo quando l'episodio è finito.

Forward view of TD(λ)

Viene fatto stato per stato per ogni istante di tempo come se non fosse sjuccesso nulla.

- V is not changed until the end of the episode.
- At the end of the episode, we compute G_t^λ and make updates.
  Media pesata tramite lambda
- 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)).
    In caso task continuativo tronco ad h passo

Vedere esempio fatto a lezione mecordì 21/04/2021 minuto 44:00



Backward view of TD(λ)

- The forward view is not implementable
  - acausal. poiché l'aggiornamento corrente dipende dalla conoscenza del futuro.
- 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

perché gli stati che visito sono random

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

Andamento traccia di eleggibilità: decresce in maniera esponenziale se non scelto

$$E_t(s) \in \mathbb{R}^+.$$ >=0

- 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

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

TD(0)

$V(S_t) \leftarrow V(S_t) + \alpha(R_{t+1} + \gamma V(S_{t+1}) - V(S$

$V(S_{t+1}) \leftarrow V(S_{t+1}) + \alpha(R_{t+2} + \gamma V(S_{t+2}) - V($

$V(S_{t+2}) \leftarrow V(S_{t+2}) + \alpha(R_{t+3} + \gamma V(S_{t+3})$

$\alpha\gamma\lambda$

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

Gli stati visitati istanti di tempo molto vecchi, laggiornamento ha poca influenza mentre gli altri hanno molta influenza.

The TD(λ) algorithm

- The TD error for state-value prediction is

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

errore commessio nell'aggiornamento in TD.

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

incremento a tutti gli stati l'errore nell'istante di tempo per la traccia di eliggibilità

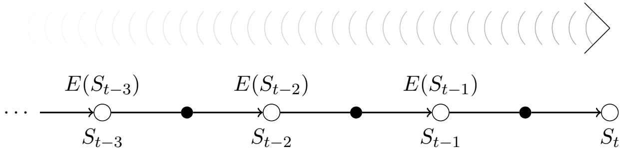
$$\Delta_t(s) = \alpha \delta_t E_t(s), \quad \forall s \in \mathcal{S}.$$

- These increments can be bone both on-line and off-line.

Se in off-line-->forward view

Se on-line->backyard view

- 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

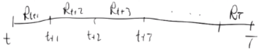
$V(s) \leftarrow \text{arbitrary}, \forall s \in \mathcal{S}$
$V(\text{terminal}) \leftarrow 0$
$E(s) \leftarrow 0, \forall s \in \mathcal{S}$

Loop

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

Usato per predizione

Forward view



$G_{t:t+1} = R_{t+1} + \gamma \underbrace{V(S_{t+1})}_{G_{t+1}}$ ← alta bias, bassa varianza

$G_{t:t+2} = R_{t+1} + \gamma R_{t+2} + \gamma^2 \underbrace{V(S_{t+2})}_{G_{t+2}}$

$G_{t:t+3} = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 \underbrace{V(S_{t+3})}_{G_{t+3}}$

$G_{t:T} = G_t$ ← zero bias, alta varianza

$G_t^\lambda = (\lambda - 1) G_{t:t+1} + (\lambda - 1)\lambda G_{t:t+2} + (\lambda - 1)\lambda^2 G_{t:t+3} \dots + \lambda^{T-t-1}$

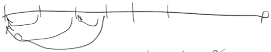
$\downarrow$

$V(S_t) \leftarrow V(S_t) + \alpha (G_t^\lambda - V(S_t))$

$\downarrow$

$V(S_t) \leftarrow V(S_t) + \alpha (G_t^\lambda - V(S_t))$

Backward view



$\lambda \le 1 \quad \gamma \le 1$

$E(s)$

$E(s) \leftarrow \gamma\lambda\, E(s) \quad \forall s \ne S_t$, ≤ 1

$E(S_t) \leftarrow E(S_t) + 1$

1

$(1-\alpha)\gamma\lambda E(S_t) + 1$

$\delta = R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$

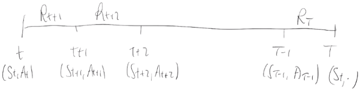
Errore TD a singolo step

$V(s) \leftarrow V(s) + \alpha\, \delta\, E(s) \quad \forall s$

Nel controllo utilizziamo la stima della funzione qualità

Controllo passo tra coppie stato azione

$$\text{Stimare } Q(s,a) \longrightarrow \pi(s) = \arg\max_a Q(s,a)$$



Funzione qualità =0
nello stato terminale

$$G_{t:t+1} = R_{t+1} + \gamma Q(S_{t+1}, A_{t+1})$$

funzione qualità per predire il futuro
Effettuo qui il bootstrapping
es.SARSA

$$G_{t:t+2} = R_{t+1} + \gamma R_{t+2} + \gamma^2 Q(S_{t+2}, A_{t+2})$$

G_{t+2} $= 0$

$$G_{t:T} = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{T-t-1} R_{T-1} + \cancel{\gamma^{T-t} Q(S_T, A_T)}$$

RItorno ad n passi. Da ora devo mediare

$$G_t^\lambda = (1-\lambda) G_{t:t+1} + (1-\lambda)\lambda G_{t:t+2} + \cdots + \lambda^{T-t-1} G_t$$

Legge di aggiornamento

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left(G_t^\lambda - Q(S_t, A_t) \right)$$

Tuttto ciò deve essere fatto per ogni istante di tempo, fino alla fine dell'episodio. Converge alla funzione di qualità vera

Infine:

$$\pi \leftarrow \pi(s) = \arg\max_a Q(s,a)$$

Backward view per il controllo

$E(s,a) \leftarrow 0$ tracce di eliggibilità per la funzione qualità

$E(s,a) \leftarrow \gamma\lambda E(s,a) \quad \forall (s,a) \neq (S_t, A_t)$ Per tutte le coppie state-azione non selezionato, anche per lo stato corrento

$E(S_t, A_t) \leftarrow E(S_t, A_t) + 1$ accumulating

Tracce di eliggivilità accumulative

1 replacing

$(1-\alpha)\gamma\lambda E(S_t, A_t) + 1$ dutch

Legge di aggiornamento, devo definire l'errore TD ad un passo

$\delta = R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)$ Errore TD per la funzione qualità

$Q(s,a) \leftarrow Q(s,a) + \alpha\delta E(s,a) \quad \forall s, \forall a$ Ripropago all'indietro per tutte le coppie stato azione. Riempendo Q(s,a). Sapendo Q(s,a) aggiorno la policy

Legge di aggiornamneot della policy

$\pi \leftarrow \pi(s) = \arg\max_a Q(s,a)$

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)

A differenza del SARSA, utilizziamo una TD di lambda. In particolare, si utilizza una visione all'indietro tramite le tracce di eliggibilità

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

In base alla funzione qualità

Successivo

Aspetto il prox stato

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

- Traces $E_t(s, a)$ for state-action pairs

Devo garantire che decrescono per tutte le coppie stato azione non selezionate

  - 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

Se processo stazione e alpha decresce e la policy è GLIE (Greedy in the limit) questo argomente con probabilità 1 a q*(s,a)

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

lamba prosismo a 0->TD a un passo (SARSA(0))
lambda = 1-> MC

Initialization

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

Loop

$E(s,a) \leftarrow 0, \forall a \in \mathcal{A}(s), \forall s \in \mathcal{S}$ All'inizio dell'episodio
initialize S
$A \leftarrow$action derived by $Q(S,\cdot)$ (e.g., ε-greedy) rispetto a Q(s,a) in quello stato
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) rispetto allo stato successivo
    $\delta \leftarrow R + \gamma Q(S',A') - Q(S,A)$ errore alle differenze temporali
    $E(S,A) \leftarrow (1-\alpha)E(S,A) + 1$ (dutch trace) Della coppia stazio azione appena visitato
    for all $s \in \mathcal{S}$ and all $a \in \mathcal{A}(s)$ **do**
        $Q(s,a) \leftarrow Q(s,a) + \alpha\delta E(s,a)$ più la soglia è alta meno propago, ma devo fare meno aggiornamento e viceversa
        $E(s,a) \leftarrow \gamma\lambda E(s,a)$
    $S \leftarrow S'$
    $A \leftarrow A'$
    if S is terminal **then**
        reinitialize the episode

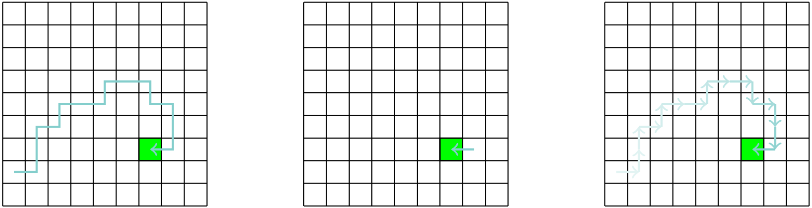
Propago all' indietro

Advantages of SARSA(λ)

è un algoritmo policy perché si utilizza Q(s,a) per la scelta dell'azione corrente e successiva. Sfrutto la policy che sto apprendendo per generare l'esperienza

L'informazione appresa dal SARSA(0) è relativo all'ultimo passo poiché l'utlimo valore è quello efficace: ha il reward minmo.
L'algoritmo propagherà all'indietrol'azione migliore ogni qual volta raggiungo l'informazione rilevante fino all'inizio.
Con un altro episodio e tornando allo stato della freccia ho una informazione rilevante.
Nel sarsa(lambda) ripropago lìinformazione per tutti i passi precedendo: ad ogni trasmissione allìindietro l'informazione diventa meno importanto. Non sfrutto solamente l'ultima informazione, ma la sfrutto al massimo avnedo una convergenza più veloce..

Path taken | Learnt from SARSA(0) | Learnt from SARSA(λ)



EXPECTED SARSA(lambda): occorre cambiere solamente il target. Il resto è uguale.
Nel SARSA classico:

$$Q(S,A) \leftarrow Q(S,A) + \alpha\left(R_{t+1} + \gamma Q(S',A') - Q(S,A)\right)$$

In expected SARSA non devo scegliere A':

$$Q(S,A) \leftarrow Q(S,A) + \alpha\left(R_{t+1} + \gamma E_{\pi}[Q(S',a)] - Q(S,A)\right)$$

$$\sum_{a} \pi(a|S')\, Q(S',a)$$

$$A^* = \arg\max_{a} Q(S',a)$$

$$\left(\sum_{a} \frac{\varepsilon}{|A|} Q(S',a)\right) + (1-\varepsilon)\, Q(S',A^*)$$

$$\left\{ R_{t+1} + \gamma E_{\pi}[Q(S',a)] - Q(S,A) \right\} \delta_t$$

Utilizzando questo termine di errore ottengo un SARSA(lambda)

. A* azione greedy rispetto a Q(S',) nel SARSA normale

Questo algoritmo è più efficiente del SARSA classico

Q(λ)

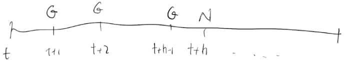
Non si può utilizzare l'aggiornamento fin'ora utilizzato perché Q learning era efficace perché non dovevamo considerare l'importance sampling: quando prendevo un azione di tipo diverao da pi, l'importance samplig diventava zero perché la policy era di tipo deterministico.

Possiamo usare TD con Q learing utilizzando l'esperienza a disposizione fino all'azione greedy.in caso di azione esploratoria devo troncare. In una visione in avanti mi fermo alla prima azione esploratoria presa.

- 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.



Utilizzo fl'esperienza fino alla azione non-greedy t+h-1

$$\frac{\pi(A_{t+n})}{b(A_{t+n})} = 0$$

Errore nel Q-learning:

$$\delta_Q = R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)$$

PEnsando di estrarre informazioni a più passi:

$$\delta = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots + \gamma^{n-1} R_{t+n} + \gamma^n \max_a Q(S_{t+n}, a) - Q(S_t, A_t)$$

Visione in avanti

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, $\lambda \to 1$.