Università di Bologna
Campus di Cesena
Ingegneria e Scienze Informatiche LM

Progetto di Sistemi Intelligenti Robotici

Q-learning in Lua: libreria per l'apprendimento per rinforzo

Magnini Matteo
matteo.magnini@studio.unibo.it

2 luglio 2020

Sommario

In questo articolo è presentata una libreria, scritta in linguaggio Lua, che permette di scrivere controlli per robot che utilizzano il Q-learning. Nel seguito vengono mostrati alcuni esempi applicativi di utilizzo della libreria. Per ognuno degli esempi viene fornita una valutazione quantitativa della bontà del comportamento appreso. Infine viene confrontato un controllo a sussunzione con competenze apprese tramite Q-learning con uno con competenze scritte dal progettista. Per tutto l'articolo si usa il termine generico robot, nelle sezioni degli esperimenti si usa il termine robot con specifico riferimento al *footbot*. Gli esperimenti sono stati eseguiti in simulazione tramite *ARGoS*.

1 Libreria

Scopo della libreria è quello di fornire delle funzioni di facile utilizzo e comprensione per la realizzazione di progetti che fanno uso di apprendimento per rinforzo tramite Q-learning. Le funzioni fornite sono incentrate sulla *Q-table*, cioè la matrice stato per azione che raccoglie la conoscenza appresa durante l'addestramento.

Attraverso le funzioni in tabella è possibile scrivere controlli per robot che apprendono uno specifico comportamento molto facilmente. Lo sviluppatore ha

Funzione	Parametri	Ritorno	Sintesi
create_Q_table	n° stati, n° azioni	Q-table	Inizializzazione di una matrice n° stati x n° azioni con tutti i valori di cella a zero.
save_Q_table	nome file, Q-table		Salva in formato csv la matrice nel file specificato.
load_Q_table	nome file	Q-table	Importa da file csv la Q-table.
get_best_action	stato, Q-table	azione	Fornisce l'indice dell'azione migliore dato uno stato.
update_Q_table	alpha, gamma, stato, azione, ricompensa, stato futuro, Q-table	Q-table	Aggiorna la Q-table secondo i parametri dell'algoritmo di Q-learning.
get_random_action	epsilon, stato, Q-table	azione	Con probabilità epsilon fornisce l'indice di un'azione casuale, con 1 - epsilon invece fornisce la migliore.
get_weighted_action	k, stato, Q-table	azione	Fornisce l'indice di un'azione sulla base di una selezione pesata in funzione di k e del corrispettivo valore nella Q-table.

Tabella 1: descrizione funzioni della libreria.

unicamente il compito di definire lo spazio degli stati e delle azioni, nonché la funzione di ricompensa, del problema di apprendimento che vuole affrontare.

La struttura del controllo è sempre la medesima a prescindere dallo scenario. Ad ogni chiamata della funzione `step` del controllo si esegue aggiornamento della Q-table ed esecuzione dell'azione. L'aggiornamento prevede di chiamare una funzione che calcola lo stato attuale del robot e successivamente chiamare `update_Q_table`. L'esecuzione assegna ad una variabile di appoggio il valore dello stato, chiama la funzione `get_random_action` o `get_weighted_action` per ottenere l'azione da compiere, ed infine la esegue.

```
1  function step()
2    n_steps = n_steps + 1
3
4    if n_steps % MOVE_STEPS == 0 then
5
6      -- Update
7      state = get_state()
8      Q_table = Qlearning.update_Q_table(alpha, gamma, old_state,
       ↪  action, get_reward(), state, Q_table)
9
10     -- Perform action
11     old_state = state
```

```
12      action = Qlearning.get_random_action(epsilon, old_state,
        ↪  Q_table)
13      --action = Qlearning.get_weighted_action(k, old_state,
        ↪  Q_table)
14      perform_action(action)
15
16    end
17
18  end
```

`MOVE_STEPS` non dovrebbe assumere valori troppo elevati, durante gli addestramenti si è scelto come valore 5 (stesso valore come massima velocità delle ruote).

2 Applicazioni

La libreria è stata utilizzata per la realizzazione di controlli in grado di permettere al robot di apprendere uno specifico comportamento. Parte critica di queste applicazioni è la corretta definizione degli stati, delle azioni e della funzione di ricompensa. Come linea di principio la Q-table non deve essere di grandi dimensioni per permettere all'algoritmo di convergere in tempi ragionevolmente brevi. Al contempo, una Q-table di dimensioni ridotte può risultare essere poco espressiva, impedendo l'apprendimento del comportamento desiderato.

2.1 Path Following

Obiettivo: il robot deve essere in grado di muoversi su di un tracciato disegnato sulla mappa.

Stati: sono utilizzati gli 8 sensori di `base_ground` per la costruzione dello spazio degli stati. I sensori sono disposti circolarmente attorno al robot ed ognuno di essi può assumere valore 0 se il colore del suolo è nero, 1 se è bianco. Di conseguenza gli stati del robot sono $2^8 = 256$.

Azioni: sono possibili solo 5 azioni, ovvero muoversi di un vettore costante in modulo[1] ma con 5 diverse direzioni. Le direzioni sono espresse in radianti rispetto alla testa del robot e sono: $\pi/4$, $\pi/8$, 0, $-\pi/8$ e $-\pi/4$. Tali angoli corrispondono alle direzioni WNW, NNW, N, NNE ed ENE nella rosa dei venti. Queste azioni permettono al robot di muoversi in modo rettilineo, oppure di compiere delle virate più o meno accentuate a destra o a sinistra. È importante notare come le azioni scelte siano prive di simmetrie, cioè di azioni in grado di opporsi ad altre azioni. In questo modo si evita di far compiere al robot comportamenti non graditi che tuttavia soddisfano l'obiettivo (es: muoversi avanti ed indietro ripetutamente sul tracciato rimanendo di fatto nello stesso intorno).

[1] per rispettare il vincolo di `MAX_VELOCITY` durante la conversione al modello differenziale, i vettori con direzione diversa da zero radianti hanno in verità modulo inferiore.

Q-table: la dimensione della Q-table è $2^8 * 5 = 1280$.

Ricompensa: sia S_t il numero di sensori che percepiscono il colore nero all'istante t, allora la ricompensa per tale istante è $R_t = (S_t/8)^2$. Il valore della ricompensa va da 0 ad 1 inclusi e non è lineare, da maggior peso quando il robot è quasi o interamente sul circuito.

2.2 Obstacle Avoidance

Obiettivo: il robot deve muoversi evitando gli ostacoli nelle sue prossimità.

Stati: il robot possiede 24 sensori `proximity` disposti circolarmente. I sensori forniscono valori da 0, nessun ostacolo entro 10 cm, ad 1, contatto con un ostacolo. Essendo il valore continuo, questo viene binarizzato con soglia 0.05 (a discrezione del progettista) per poter avere un numero di stati finito. Tuttavia usando tutti e 24 i sensori si ottengono 2^{24} stati, cioè più di 16 milioni. Per ridurre tale numero, i sensori sono raggruppati in 8 settori da 3 sensori ciascuno. Se almeno uno dei 3 sensori del settore ha valore 1, allora l'intero settore ha valore 1, 0 altrimenti. Così facendo gli stati sono $2^8 = 256$.

Azioni: definite come nel Path Following.

Q-Table: la dimensione della Q-table è $2^8 * 5 = 1280$.

Ricompensa: sia S_t il numero di settori che percepiscono un ostacolo all'istante t, allora la ricompensa per tale istante è $R_t = 1 - (S_t/8)^2$.

2.3 Phototaxis

Obiettivo: il robot deve avvicinarsi il più possibile ad una sorgente luminosa.

Stati: il robot possiede 24 sensori `light` disposti circolarmente. I sensori forniscono valori da 0, assenza di luce, ad 1, intensità luminosa massima. A differenza dei problemi precedenti, in questo caso il fattore che influenza il valore dei sensori è posto ad una distanza elevata rispetto al robot. Una soluzione che preveda la semplice binarizzazione per soglia dei valori dei sensori risulterebbe inappropriata poiché troppo poco espressiva. Per questo motivo il segnale viene quantizzato in 4 intervalli. Gli intervalli scelti sono:

- $[0, 0.05[$ assenza di luce o troppa poca luce per essere rilevante;
- $[0.05, 0.33[$ luce a debole intensità;
- $[0.33, 0.66[$ luce a media intensità;
- $[0.66, 1.0]$ luce a forte intensità.

I sensori sono raggruppati similmente come per l'Obstacle Avoidance, viene considerato il valore più alto tra i sensori del settore per determinare il valore del settore stesso. Il numero di stati è $4^8 = 65536$, considerevolmente più alto rispetto ai precedenti due problemi, ma ancora gestibile.

Azioni: definite come nel Path Following.

Q-Table: la dimensione della Q-table è $4^8 * 5 = 327680$.

Ricompensa: sia $V_{t,i}$ il valore del settore i-esimo all'istante t, il valore è 0 per il primo intervallo, 1 per il secondo intervallo, 2 per il terzo intervallo e 3 per l'ultimo intervallo. Allora la ricompensa per tale istante è

$$R_t = \left(\frac{\sum_{i=1}^{8} V_{t,i}}{8 * 3} \right)^2$$

Al denominatore 8 è il numero di settori e 3 il valore massimo del settore. Come per le ricompense precedenti, tale valore è compreso tra 0 ed 1 inclusi.

2.4 Risultati

Tutti gli addestramenti sono stati effettuati con i seguenti parametri statici:

- alpha = 0.1;
- gamma = 0.9;
- epsilon = 0.9;
- numero di epoche = 50;
- tick per epoca = 10000.

I test hanno previsto per ogni problema 100 epoche, ognuna con 5000 tick. Nei test il controllo ha un'architettura a sussunzione nella quale di default viene eseguito un random walk, mentre se il robot non si trova nello stato di assenza di stimoli (tutti i sensori o settori a 0) viene eseguito il comportamento appreso. Ogni problema ha una sua specifica metrica di valutazione.

Per il Path Following nell'apprendimento si è usata un'arena 4 x 4 con l'immagine in figura 1, che presenta un circuito piuttosto regolare a livello di linee. Il test è stato effettuato sul circuito disegnato a mano libera in figura 2. Sia per l'apprendimento che per il test il robot parte al centro dell'arena. La metrica usata è banalmente il numero di acquisizione di stati in cui il robot è sul circuito, anche solo parzialmente, sul numero totale di stati acquisiti.

Nell'Obstacle Avoidance, sia per l'apprendimento che per il test, si è usata un'arena 6 x 6 con generazione randomica di ostacoli. Il robot parte sempre dal centro. La metrica è il numero di acquisizione di stati in cui il robot non ha ostacoli vicini sul numero totale di stati acquisiti.

Anche nella Phototaxis l'arena è sempre la stessa: 6 x 6 con la sorgente luminosa fissa e la posizione di partenza del robot randomica entro uno specifico intervallo. La metrica è la distanza euclidea tra robot e sorgente luminosa al termine dell'epoca.

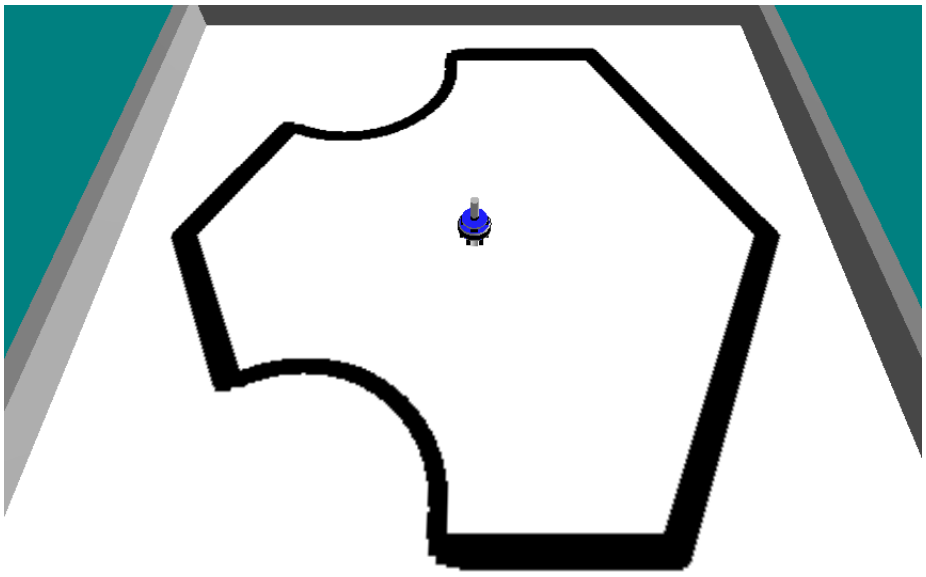

Figura 1: Arena con circuito di apprendimento.

Figura 2: Arena con circuito di test.

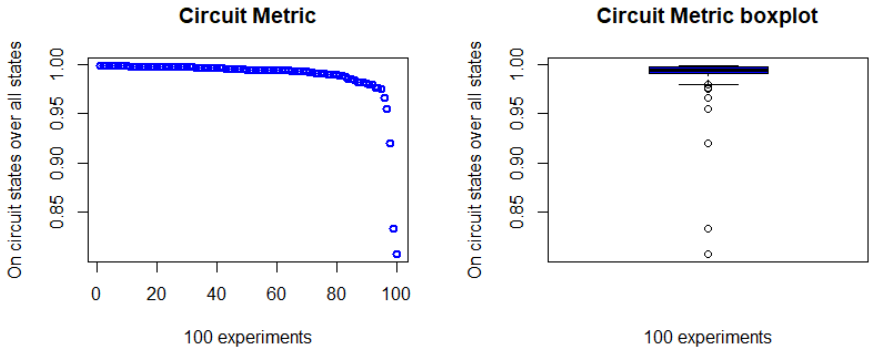

Figura 3: Risultati dei test su Path Following.

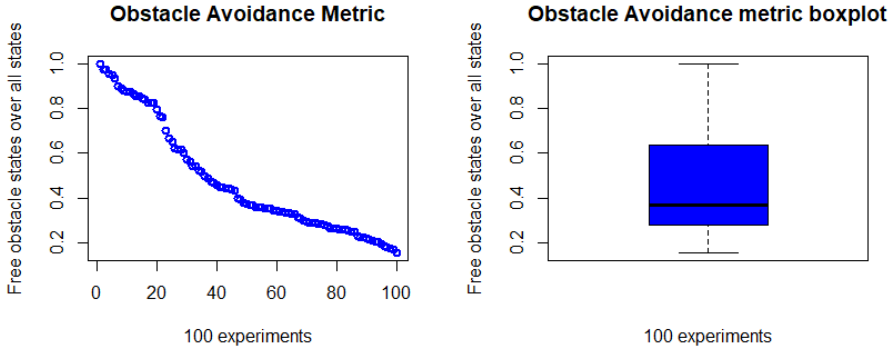

Figura 4: Risultati dei test su Obstacle Avoidance.

Nel Path Following e nell'Obstacle Avoidance è stato inserito un rumore limitato sulle azioni che riduce randomicamente le velocità delle ruote dallo 0% al 10% di `MAX_VELOCITY`. È stata fatta tale scelta poiché è frequente in questi scenari che il robot si imbatta in punti di stallo. Non è raro che il robot possa rimanere incastrato tra degli ostacoli o che entri in stato stazionario ciclico su un tratto di percorso con una curva particolarmente accentuata.

Complessivamente in tutti gli esperimenti il robot è stato in grado di apprendere il comportamento desiderato in un ragionevole ammontare di tempo (nell'ordine del minuto). Per quanto riguarda Obstacle Avoidance, la configurazione dell'arena è particolarmente ostile al raggiungimento di alti valori nella metrica poiché sono presenti molti ostacoli, oltre ai bordi stessi dell'arena.

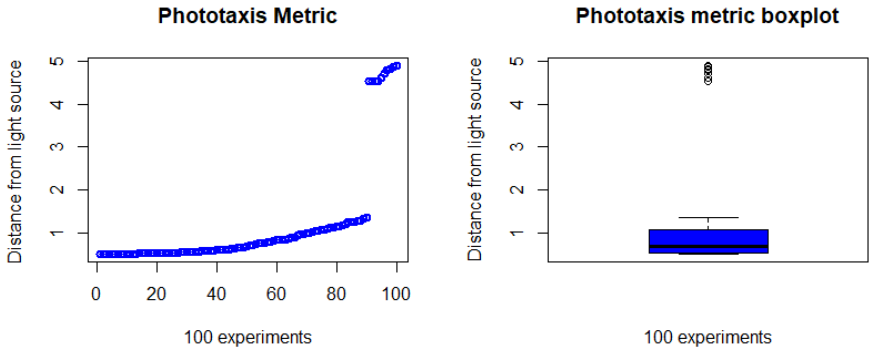

Figura 5: Risultati dei test su Phototaxis.

2.5 Extra

Visti i buoni risultati ottenuti, si confrontano due controlli a sussunzione per avere sia Phototaxis sia Obstacle Avoidance: uno le cui competenze sono scritte dal progettista, l'altro ha le competenze apprese tramite Q-learning. Entrambi i controlli, tranne per la distinzione precedentemente fatta, sono posti nelle stesse condizioni. L'arena ha dimensione 6 x 6 con una sorgente luminosa fissa, mentre posizione del robot e degli ostacoli è randomica in una specifica area. Sono stati effettuate 200 epoche, ognuna da 10000 tick. Come metrica è stata utilizzata la distanza euclidea del robot dalla sorgente luminosa al termine dell'epoca.

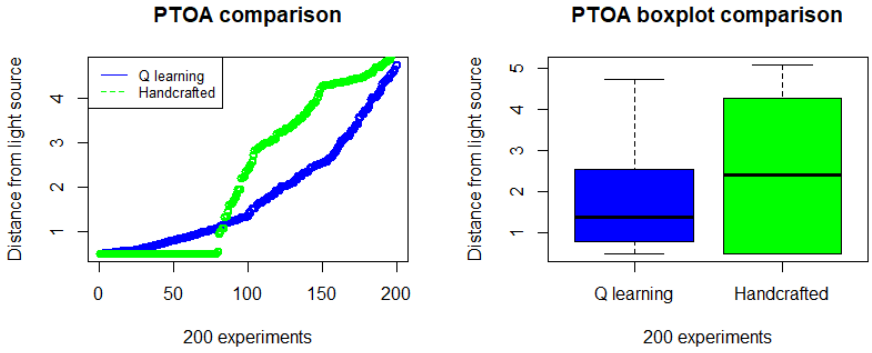

Figura 6: Confronto risultati Phototaxis con Obstacle Avoidance.

Il test di Wilcoxon tra le due popolazioni di valori ottenuti rigetta l'ipotesi nulla con significatività 0.01, pertanto il controllo col Q-learning è statisticamente migliore rispetto a quello handcrafted.