3.2 MARKOV DECISION PROCESS

I problemi di *Reinforcement Learning* possono essere definiti formalmente utilizzando i *Markov Decision Process* (MDP), in quanto permette di ottenere una descrizione dell'ambiente. Di seguito vengono illustrati gli aspetti fondamentali, ma occorre chiarire che per semplicità di notazione si assume che gli stati e le azioni siano discrete.

3.2.1 Proprietà di Markov

Uno stato S al tempo t, definito come S_t, presenta la proprietà di *Markov* se e solo se:

$$\mathbb{P}[S_{t+1}|S_t] = \mathbb{P}[S_{t+1}|S_1, S_2, ..., S_t] \tag{3.4}$$

Questo significa che è necessario solamente lo stato attuale per ottenere tutte le informazioni rilevanti dagli stati precedenti.

3.2.2 Markov Decision Process

Un *Markov Decision Process* rappresenta un ambiente in cui tutti gli stati hanno la proprietà di *Markov* e può essere descritto come una tupla $< \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma >$, dove:

- $\mathcal{S}$ è l'insieme finito degli stati;
- $\mathcal{A}$ è l'insieme finito delle azioni;
- $\mathcal{P}$ è la *State-Transition Probability* (descritta in 3.1.3);
- $\mathcal{R}$ è la *Reward Function* (descritta in 3.1.4);
- γ è il *Discount Factor* per i *reward* futuri, definito come $\gamma \in [0, 1]$.

Di seguito vengono illustrati i concetti necessari per comprendere la risoluzione di un *Markov Decision Process*.

3.2.3 Policy

Per poter interagire con l'ambiente, l'agente utilizza una funzione chiamata *policy*, in grado di descrivere il suo comportamento, in quanto permette di mappare gli stati s in azioni a.

Esistono due tipologie di *policy* a seconda che l'azione scelta sia deterministica o stocastica.

La *policy* deterministica denominata con $\pi(s)$ associa ad ogni stato un'unica azione.

$$\pi(s) = a \tag{3.5}$$

Mentre la *policy* stocastica denominata con $\pi(a|s)$ viene definita come la probabilità di compiere un'azione a dato uno stato s.

$$\pi(a|s) = \mathbb{P}[A_t = a | S_t = s] \tag{3.6}$$

Considerata l'importanza della *policy* nei problemi di RL, è possibile ridefinire le *State-Transition Probabilities* e la *Reward Function* come segue:

$$\mathcal{P}^{\pi}_{ss'} = \sum_{a \in A} \pi(a|s) \mathcal{P}^{a}_{ss'} \tag{3.7}$$

$$\mathcal{R}^{\pi}_{s} = \sum_{a \in A} \pi(a|s) \mathcal{R}^{a}_{s} \tag{3.8}$$

3.2.4 Value Function

Se la *Reward Function* esprime numericamente la qualità di un'azione o di uno stato in un determinato istante temporale, la *Value Function* invece ne specifica la qualità nel lungo periodo.

Return

Per stimare la *Value Function*, è necessario utilizzare il *return* G_t, il quale viene definito come il *Discounted Reward* totale a partire dal tempo t e viene illustrato come segue:

$$G_t = R_{t+1} + \gamma R_{t+2} + ... = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{3.9}$$

Dove $\gamma \in [0, 1]$, denominato *Discount Factor*, penalizza i *reward* futuri per le seguenti motivazioni:

- hanno una componente maggiore di incertezza;
- non forniscono benefici nell'immediato;
- da un punto di vista matematico risulta conveniente.

Esistono due tipologie di *Value Function*:

1. la *State-Value Function* che permette di valutare la qualità di uno stato;
2. la *Action-Value Function* che consente di valutare la qualità di un'azione in uno stato.

State-Value Function

La **State-Value Function** $v_\pi(s)$ è il valore atteso del *return* a partire da uno stato s e seguendo una *policy* π.

$$v_\pi(s) = \mathbb{E}_\pi[G_t|S_t = s] \tag{3.10}$$

Action-Value Function

La **Action-Value Function** $q_\pi(s, a)$ è il valore atteso del *return* a partire da uno stato s utilizzando un'azione a e seguendo una *policy* π.

$$q_\pi(s, a) = \mathbb{E}_\pi[G_t|S_t = s, A_t = a] \tag{3.11}$$

3.2.5 Equazione di Bellman

Sfruttando le proprietà ricorsive di una *Value Function*, l'equazione di *Bellman* permette di scomporla in due componenti più semplici da risolvere: il primo è il *reward* immediato, mentre il secondo è la *Discounted Value Function*, ovvero la *Value Function* scontata di un fattore γ allo stato successivo.

La *Bellman Expectation Equation* per la *State-Value Function* può essere definita come segue:

$$v_\pi(s) = \sum_{a \in A} \pi(a|s)(\mathcal{R}_s^a + \gamma \sum_{s' \in S} \mathcal{P}_{ss'}^a \, v_\pi(s')) \tag{3.12}$$

In modo analogo per la *Action-Value Function*:

$$q_\pi(s, a) = \mathcal{R}_s^a + \gamma \sum_{s' \in S} \mathcal{P}_{ss'}^a \sum_{a' \in A} \pi(a', s') \, q_\pi(s', a') \tag{3.13}$$

L'equazione di *Bellman* permette di esprimere la relazione tra il valore di uno stato e il valore dei suoi stati successivi. Pertanto risulta essere un metodo molto importante da applicare nell'ambito del RL, in quanto fornisce le basi per il calcolo, l'approssimazione e l'apprendimento delle *Value Function*.

3.2.6 Value Function Ottimale

Risolvere un problema di *Reinforcement Learning* significa individuare una *policy* capace di massimizzare i *reward* nel lungo periodo.

Le *Value Functions* definiscono un ordine parziale sulle *policy*, il quale viene definito come:

$$\pi \geqslant \pi' \ \text{ if } \ v_\pi(s) \geqslant v_{\pi'}(s), \quad \forall s \tag{3.14}$$

Esiste sempre una *policy* π_*, denominata *policy ottimale*, che risulta essere maggiore o uguale a tutte le altre *policy*.

$$\pi_* \geqslant \pi, \quad \forall \pi \tag{3.15}$$

Tutte le *policy ottimali* condividono la stessa *State-Value Function* che viene chiamata *State-Value Function ottimale* indicata con v_*.

$$v_*(s) = \max_\pi v_\pi(s) \tag{3.16}$$

Pertanto, l'equazione di *Bellman* ottimale per v_* può essere riscritta come segue:

$$v_*(s) = \max_a \mathcal{R}_s^a + \gamma \sum_{s' \in S} \mathcal{P}_{ss'}^a \, v_*(s') \tag{3.17}$$

Lo stesso ragionamento può essere applicato alla *Action-Value Function*, che viene quindi chiamata *Action-Value Function ottimale* (indicata con q_*).

$$q_*(s, a) = \max_\pi q_\pi(s, a) \tag{3.18}$$

L'equazione di *Bellman* ottimale per q_* può essere riscritta come segue:

$$q_*(s, a) = \mathcal{R}_s^a + \gamma \sum_{s' \in S} \mathcal{P}_{ss'}^a \, \max_{a'} q_*(s', a') \tag{3.19}$$

Utilizzando questo approccio per la risoluzione di un problema di RL, appare fondamentale conoscere il modello dell'ambiente, avere a disposizione una potenza computazionale in grado di risolvere un problema di ricerca esaustiva e godere della proprietà di *Markov*.