Artificial Intelligence

Giuseppe Magazzù

2021 - 2022

Contents

Chapter 1

Introduction

1.1 Acting Humanly (Turing Test Approach)

Alan Turing ha progettato il **Turing Test** per fornire una definizione operazionale di intelligenza.

Per passare il test un computer deve possedere le seguenti capacità:

- NLP: comunicare in una lingua comprensibile all'uomo
- Knowledge Representation: memorizzare quello che sa e che ascolta
- Automated Reasoning: usare le informazioni raccolte per formulare risposte e trarre conclusioni
- Machine Learning: adattarsi ed estrapolare pattern

In aggiunta per avere anche una simulazione fisica di intelligenza sono necessari:

- Computer Vision: percepire oggetti
- Robotics: manipolare oggetti e muoversi

1.2 Thinking Humanly (Cognitive Approach)

Per poter dire che un computer pensa come un umano è necessario prima determinare il funzionamento della mente umana. Da questo è nato il campo della **cognitive science** che unisce i modelli di intelligenza artificiale agli esperimenti psicologici per costruire e testare teorie su come pensa un umano.

1.3 Acting Rationally ("laws of thought" Approach)

Questo approccio nasce dall'idea che delle leggi di pensiero (laws of thought) governino la mente umana. Si è cercato quindi di tradurre la conoscenza in linguaggio logico con l'obiettivo di costruire sistemi intelligenti basati sulla logica.

1.4 Thinking Rationally

Questo approccio si basa sul concetto di agente, ovvero qualsiasi cosa che agisce. Un **agente razionale** è un agente che agisce per raggiungere il miglior risultato.

Chapter 2

Agents

Un **agente** è qualsiasi cosa che può essere vista come percepente del suo ambiente attraverso sensori e che agisce sull'ambiente attraverso degli attuatori.

Per **percezione** si intende un'insieme di input percettivi di un agente in un determinato istante. Una **sequenza di percezione** è l'intera storia delle percezioni che l'agente ha percepito fino ad oggi.

La scelta dell'azione da compiere di un agente può dipendere dall'intera sequenza di percezione, ma non da qualcosa che non ha percepito.

Il comportamento di un agente è descritto da una **agent function** che mappa una sequenza di percezioni ad azioni. L'agent function (astrazione) viene implementata da un **agent program** (implementazione).

2.1 Rational Agent

Quando un agente viene messo in un ambiente genera una sequenza di azioni in base alle percezioni che riceve. Questa sequenza causa l'ambiente ad andare attraverso una sequenza di stati.

Se la sequenza di stati è **desiderabile**, allora l'agente ha performato bene. La desiderabilità è data da una **performance measure** che valuta una sequenza di stati dell'ambiente.

2.1.1 Razionalità

La **razionalità** di un agente dipende da quattro cose:

- la performance measure che definisce criterio di successo

- la conoscenza a priori dell'ambiente da parte dell'agente
- le azioni che l'agente può eseguire
- la sequenza di percezione dell'agente

Per ogni possibile sequenza di percezione, un **agente razionale** dovrebbe scegliere un'azione che ci si aspetta massimizzi la performance measure, date le evidenze fornite dalla sequenza di percezioni e da una qualsiasi conoscenza a priori.

2.1.2 Onniscienza e Apprendimento

Un agente **onnisciente** conosce i risultato attuale e agisce di conseguenza.

La razionalità massimizza la performance attesa, mentre la perfezione massimizza la performance attuale.

Una parte importante della razionalità è l'**information gathering**, ovvero fare delle azioni al fine di modificare le percezioni future.

Un agente razionale deve, oltre a raccogliere informazioni, **imparare** il più possibile da quello che percepisce. La configurazione iniziale di un agente può contenere della conoscenza a priori dell'ambiente che con l'esperienza può essere modificata e aumentata.

2.1.3 Autonomia

Un agente razionale dovrebbe anche essere **autonomo**, ovvero affidarsi unicamente alle sue percezioni e compensare ad una conoscenza a priori parziale o scorretta fornita dal suo progettatore. Dopo una sufficiente esperienza dell'ambiente il comportamento può diventare indipendente dalla conoscenza a priori.

2.2 Task Environment

Per definire un agente è necessario determinare il suo **task environment** che è descritto da performance measure, ambiente, attuatori e sensori. Questi insieme di fattori può essere riassunto con l'acronimo **PEAS** (**P**erformance, **E**nvironment, **A**ctuators, **S**ensors).

Fully Observable vs Partially Observable

Quando i sensori di un agente gli danno accesso allo stato completo dell'ambiente in ogni momento, allora diciamo che il task environment è **completamente osservabile**. Se l'agente non ha sensori, l'ambiente è **osservabile**.

Competitive vs Cooperative

Un ambiente multi-agente è detto **competitivo** se massimizzare la performance measure di un agente minimizza la performance measure di un altro agente. Se invece tutte le performance measure vengono massimizzate, allora l'ambiente è detto **cooperativo**.

Deterministic vs Stochastic

Se lo stato successivo dell'ambiente è completamente determinato dallo stato corrente e dall'azione eseguita dall'agente, allora diciamo che l'ambiente è **deterministico**. Altrimenti è detto **stocastico**.

Uncertain vs Certain

Diciamo che un ambiente è **incerto** se non è completamente osservabile o non deterministico

Episodic vs Sequential

In un ambiente **episodico** l'esperienza dell'agente è divisa in più episodi. In ogni episodio l'agente riceve delle percezioni ed esegue una singola azione. L'episodio successivo non dipende dalle azioni prese negli episodi precedenti.

In un ambiente **sequenziale**, invece, la decisione attuale può incidere sulle decisioni future.

Static vs Dynamic

Se l'ambiente si aggiorna alle decisioni un agente, allora diciamo che l'ambiente è **dinamico** per quell'agente. In caso contrario, è **statico**.

In un ambiente statico l'agente non deve osservare lo stato dell'ambiente per decidere l'azione da eseguire.

Discrete vs Continuos

Un ambiente è **discreto** se c'è un numero fissato di azioni e percezioni al suo interno. In caso contrario, è **continuo**.

Known vs Unknown

In un ambiente **conosciuto** l'agente conosce il risultato di tutte le sue azioni sull'ambiente. Altrimenti dovrà imparare come funziona per fare delle buone scelte.

2.3 Agent Structure

L'obiettivo dell'AI è progettare un agent program che implementa la agent function che mappa le percezioni ad azioni. Il programma viene eseguito su un calcolatore con sensori e attuatori che costituiscono l'architettura.

$$\text{Agente} = \text{Programma} + \text{Architettura}$$

2.3.1 Simple Reflex Agents

Questo tipo di agente seleziona le azioni sulla base della percezione corrente, ignorando quelle passate. L'ambiente deve essere completamente osservabile (fully-observable).

2.3.2 Model-based Reflex Agents

L'agente mantiene uno stato interno che dipende dallo storico delle percezioni. Questo riflette in qualche modo gli aspetti non osservabili nello stato corrente.

Per aggiornare lo stato interno è necessario conoscere come l'ambiente evolve indipendentemente dall'agente e come l'azione dell'agente influenza l'ambiente.

2.3.3 Goal-based agents

Non sempre lo stato corrente dell'ambiente è sufficiente per decidere la prossima azione. L'agente ha bisogno di una informazione sull'obiettivo che descrive la situazione desiderabile.

Quando per raggiungere l'obiettivo, l'agente deve considerare una lunga sequenza di azioni sono necessarie delle fasi di ricerca e pianificazione.

2.3.4 Utility-based agents

L'obiettivo fornisce solo un'informazione binaria che nella maggior parte degli ambiente non basta per avere comportamenti complessi.

Si utilizza quindi una misura di performance più generale che permette di valutare gli stati dell'ambiente sulla base di quanto è utile.

Se un agente ha più obiettivi, con la funzione di utilità è possibile gestire conflitti tra essi e pesarli in base all'importanza.

Un agente razionale utility-based deve modellare e tenere traccia del suo ambiente e dei suoi task.

2.3.5 Learning agents

Questa architettura permette all'agente di operare in ambienti sconosciuti e di diventare più competente. Inoltre non è necessario definire manualmente l'agent program.

Chapter 3

RDF