Capitolo 1

Introduzione

1.1 Argomenti

- Agenti intelligenti;
- Problemi e algoritmi di ricerca;
- Giochi con problemi di ricerca: siano negli anni '70 dove si ispezionano tutti i casi possibili con un agente intelligente.
- Rappresentazione della conoscenza e ragionamento: dagli anni '70 in poi. Il calcolo proposizionale risulta non sufficientemente espressivo, ma permette di fare inferenza (conoscenza). Si parla di calcolo dei predicati e di programmazione logica. Si ottiene in questo modo una definizione del problema e del dominio che porta a ottenere un risultato del problema.
- Trattazione dell'incertezza: nel mondo reale esiste il problema dell'incertezza, diversi gradi modellati abbastaza bene con la teoria di probabilità. L'incertezza si risolve conoscendo la distribuzione di probabilità congiunta. Ovvero indipendenza condizionale dove, per esempio, se conosco C, allora A è indipendente da B (fattorizzazione della probabilità congiunta). Esperessione nelle reti *bayesiane*.
- Introduzione all'Apprendimento automatico, *Machine Learning* e Apprendimento con rinforzo: va considerato come usare un modello probabilistico ha un costo computazionale enorme. Per questo nasce l'Apprendimento automatico, ovvero quando l'elaboratore acquisisce informazioni per avere conoscenza mediante addestramento. Nelle Reti Neurali si superano le prestazioni dell'essere umano nel riconoscimento di suoni, parlato, elaborazione dati, .. . In ogni caso c'è sempre necessità di una grande potenza computazionale e dati. Per questo i maggiori *leader* in tali ambiti sono Google, IBM e Microsoft.
  Tuttavia l'aumento della potenza dell'elaboratore non porta buoni risultati solo nell'area dell'apprendimento, ma anche della probabilità.
  Si parla di apprendimento con rinforzo quando vengono provate *random* tutte le strade possibili, concetrandosi per avere risultati su quelle migliori.

- Elaborazione linguaggio naturale: è il linguaggio naturale che permette ai *robot* di comprendere e interagire con il mondo esterno;
- Visione aritificiale.

1.2 Agente intelligente

Un agente razionale è un agente intelligente.
Esistono tanti tipi di intelligenza: emotiva, relazionale, ecc.
Un agente agisce nel mondo che lo circonda, ha degli obiettivi fissati che raggiunge per mezzo dell'informazione che ha disponibile.
Incide la potenza di calcolo, percui un agente è intelligente se riesce a fare il suo meglio possibile, anche solo avvicinandosi a dei risultati.
Cerca di massimizzare il soddisfacimento dei propri bisogni sfruttando le informazioni di cui dispone, o che può acquisire per mezzo delle sue azioni.
Dal punto di vista matematico usa tempi discreti come momento di acquisizione delle informazioni, ed è formalizzabile come:

$$f : P^* \longrightarrow A \tag{1.1}$$

dove f rappresenta un obiettivo e $\longrightarrow$ tutte le informazioni possibili per poter raggiungere quell'obiettivo.
Tuttavia le limitazioni computazionali possono impedire la realizzazione di razionalità perfetta, dunque si cerca esempio di usare la memoria a disposizione nel modo migliore.
Gli agenti si distinguono in agenti fisici, come *robot*, attuatori, umani e *soft*, come il contenuto di una pagina *web*.
Il dominio di f rappresenta tutte le possibili percezioni da quando l'agente ha iniziato a funzionare.

Esempio dell'aspirapolvere: deve sapere dove si trova e se la stanza è sporca.
Percezioni: stanza, pulito/sporco.
Azioni: sinistra, destra, aspira.

L'algoritmo non è intelligentissimo, perchè se le stanza sono tutte pulite continua ad andare avanti e indietro, però tale algoritmo fa uso di pochissima memoria.
Importante è fissare la misura di prestazione che valuta la sequenza di percezioni, ovvero quanto bene ha lavorato il robottino con un certo numero di risorse.
Inoltre non è detto che il robottino aspiri sempre tutto, per questo la misura dipende anche dalle caratteristiche dell'ambiente. Lo scopo è individuare la misura di prestazione massimizzata. Per farlo non è necessario che l'agente sia omniescente, basta che sia razionale e faccia del suo meglio con le informazioni.
Razionalità non è sempre sinonimo di successo (per mancanza di risorse, esempio precisione non disponibile).
Inoltre un agente razionale deve saper anche risolvere situazioni che il progettista non ha pensato, grazie all'addestramento.

PEAS identifica 4 componenti da caratterizzare, le caratteristiche dell'ambiente dove l'agente è chiamato a operare: la misura delle prestazioni, l'ambiente operativo, attuatori (le percezioni) e i sensori (le azioni).

1.2.1 Pilota automatico per taxi

- **Misura di prestazione**: obiettivi che si intendendono realizzare, arrivare a destinazione corretta, profitto massimo, veloce, ligio alla legge, viaggio confortevole;
- **L'ambiente**: in cui l'agente deve operare, strade, altri veicoli, pedoni, clienti;
- **Attuatori**: reagiscono verso l'ambiente, acceleratore, sterzo, freno, frecce, *clacson*, schermo di interfaccia per comunicare con i passageri.
- **Sensori**: recepiscono informazioni dall'ambiente, permettendo all'agente di raggiungere i suoi obiettivi, telecamere, *sonar*, tachimetro, GPS contachilomeri, accelerometro, sensori sul motore, tastiera.

Per progettare un agente devono essere fatte delle scelte:

- **Osservabile**: quando i sensori hanno accesso allo stato completo dell'ambiente in ogni momento. I sensori devono essere in grado di misurare gli aspetti rilevanti per le scelte dell'azione. Altrimenti si parla di osservabilità parziale o non osservabile.
- **Deterministico**: lo stato successivo è completamente determinato dallo stato corrente dell'agente, in caso contratio si parla di stocasticità;
- **Episodico**: esperienza dell'agente che non deve dipendere da episodi precedenti, altrimenti si parla di sequenziale. In ambienti episodici la scelta dell'azione dipende esclusivamente dall'episodio corrente.
- **Statico**: l'ambiente è stazionario nel tempo, altrimenti parliamo di ambiente dinamico;
- **Discreto**: applicato allo stato dell'ambiente, il modo in cui è gestito il tempo, alle percezioni e azioni dell'agente. È discreto quanto si ha un numero finito e distinto. Continuo quando i punti sono successivi l'uno all'altro.
- **Agente**: numero di agenti presenti nell'ambiente, con il vincolo che l'agente B ha un comportaento che tenta di massimizzare una misura di prestazione da cui dipende il comportamento di A.

1.2.2 Tipi di agenti

Un agente ha sensori e attuatori.

Agenti a riflesso semplice

Si dimentica la storia, si ricorda solo l'ultima percezione percepita. Prende la decisione sull'azione da compiere solo sulla base di quello percepito al tempo T. Per realizzare l'agete basta soddisfare la relazione condizione soddisfatta $\rightarrow$ azione conseguente. Un esempio è quello dell'aspirapolvere. Sulla base delle informazioni percepite dai sensori, si seleziona la regola, dunque l'azione da svolgere con gli attuatori.

Tuttavia vi è un problema. Se l'ambiente è parzialmente osservabile, l'agente può fallire. Se, difatti, cambia qualche situazione di contorno si può ritrovare all'interno di un *loop* (come nel caso di un ragnetto in un labirinto portato sempre a sinistra che poi per qualche motivo si trova a dover svoltare a destra). A tale inconveniente si può reagire con una scelta *random*.
In conclusione con un agente a riflesso semplice si hanno delle regole fissate e un unico *goal*.

	Solitario	Backgammon	Internet shopping	Taxi
Osservabile??	Si	Si	In parte	In parte
Deterministico??	Si	No	In parte	No
Episodico??	No	No	No	No
Statico??	Si	Si	Semi	No
Discreto??	Si	Si	Si	No
Agente singolo??	Si	No	Si (eccetto aste)	No

Figura 1.1: Esempi di classificazione degli ambienti.

Agenti a riflesso con stato

Nasce come miglioramento alla tipologia di agente precedente. Si tiene traccia, in memoria, delle percezioni passate. È però osservabile solo se l'ambiente è statico. In questo modo si crea una mappa di come il mondo evolve e l'effetto delle singole azioni.
In questo caso nella sequenza condizione → azione è incluso anche lo stato che permette di lavorare anche in ambienti parzialmente osservabili.
Tuttavia tale approccio è ancora poco flessibile, perchè il comportamento degli attuatori rimane nelle regole che permettono di raggiungere un certo *goal*. Tale tipologia ha portato il vantaggio di togliere le limitazioni alle regole.

Agenti basati su goal

Definisco degli obiettivi internamente che permettono la selezione dell'azione da parte degli attuatori. Viene svolta una sorta di simulazione di cosa dovrebbe succedere per uno specifico *goal* in modo da selezionare le azioni che portano al suo soddisfacimento.

Agenti basati su una misura di utilità

In questo caso l'agente ha una misura di utilità. Tale funzione serve quando un'agente ha più obiettivi che non possono convivere e dunque deve seleziona quale privilegiare o a valutarne la probabilità di successo. Nel concreto la funzione utilità assegna a uno stato un numero reale che è la contentezza a esso associato. Questo porta un agente basato sugli obiettivi a assumere comportamenti di alta qualità. Per esempio nel caso del taxi ci sono molte strade che portano alla stessa destinazione ma alcune sono più sicure, più veloci, affidabili e economiche di altre.

Ogni architettura risponde ad agenti diversi a seconda del compito da svolgere.

Gli agenti possono anche apprendere, questo è possibile innestarlo su ogni tipo di agente. L'apprendimento suggerisce delle modifiche alle componenti interne delle architetture, in base a quello appreso fino a quel momento. In questo modo si ha un aggiornamento costante di regole e azioni. Per esempio sulla base del *goal* l'agente può apprendere se il mondo è continuo o stocastico, e così fare delle stime sulle azioni.
All'interno di un'architettura di apprendimento si trovano due componenti importanti:

- ***Critic***: che si occupa di valutare il risultato ottenuto da un'azione, e nel caso non sia buono continuare;
- ***Problem generator***: che mitiga il rischio di sovraspecializzazione, ovvero l'apprendimento solo su una certa parte del dominio. Il *Problem generator* forza l'agente a esplorare tutte le possibilità dell'ambiente.

Capitolo 2

Risoluzione di problemi

Inizialemente l'AI si era focalizzata nel risolvere problemi che erano già stati formalizzati. Non trattava né il rumore nelle informazioni, né l'osservazione parziale o quant'altro.
Si parla così di *problem-solving agent*, particolare agente basato sugli obiettivi, che si occupa di trattare solo argomenti già trattati.
Un esempio di argomento non trattato sono le cifre monoscritte in quanto si possono presentare con colori differenti di volta in volta, numeri con mancanza di tratto, ecc.

2.1 Formulazione del problema

La formulazione del problema è il processo che porta a decidere dato un obiettivo, quali azioni e stati considerare.
Si si deve astrarre dai dettagli e da questo:

- **Formulare il *goal***: l'obiettivo da raggiungere;
- **Formulare il problema**: descrivere gli stati e le azioni possibili;
- **Trovare una soluzione**: sequenza di azioni/stati desiderabili per il problema.

Facciamo un esempio: siamo in vacanza in Romania, si è ad Arad, e si deve prendere un volo domani per Bucharest. Sulla base del modello descritto sopra:

- **Formulare il *goal***: essere a Bucharest;
- **Formulare il problema**:
  stati: le varie citta
  azioni: spostamenti tra le città
- **Trovare una soluzione**: un percorso di città che porta da Arad a Bucharest, con l'obiettivo di minimizzare il costo (la distanza della strada da percorrere).

L'esempio si dimostra come osservabile, determinisitico e discreto.

Un problema è definito da quattro componenti:

- **Stato iniziale**: in cui si trova l'agente, dunque Arad;
- **Funzione successore**: che descrive tutte le coppie azioni possibili dall'agente e stati raggiungibili, dallo stato dove si trova l'agente. Le azioni devono essere atomiche.
- **Test per il *goal***: riconosce se si è raggiunto l'obiettivo. Non può esclusivamente trattarsi del nome di uno stato (esplicito), ma anche gli stati che soddisfano una certa proprietà (implicito).
- **Costo di un cammino addittivo**: il costo numerico di un cammino (numero di distanza, numero di azioni eseguite). Si valuta sempre il costo di ogni singolo passo che deve sempre essere ≥ 0. L'agente sceglierà il costo che rispecchia la misura di prestazione.

Una soluzione è una sequenza di azioni, da stato iniziale a stato di *goal*.

La funzione successore può non solo essere definita come la coppia (azione, stato), ma anche come le azioni eseguibili di uno stato e il *transition model* ovvero l'effetto ottenuto da ogni singola azione.

Essendo che il mondo reale è complesso per risolvere il problema lo spazio degli stati deve essere astratto.
Uno stato astratto è un insieme di stati reali, un'azione astratta è una combinazione complessa di azioni reali, per garantire la realizzabilità del modello, qualsiasi stato reale deve condurre a qualche stato reale. Anche una soluzione deve essere astratta e rappresenta un'insieme di cammini reali che sono sluzioni nel mondo reale. Inoltre ogni azione astratta deve essere più semplice del problema originario.

Esistono diverse strategie di ricerca:

- Ricerca su grafo;
- Ricerca su albero per profondità o per ampiezza.

Le ricerche su albero in profondità assumono una coda con priorità LIFO, ove l'ultimo nodo a essere visitato viene posto in cima allo *stack*, invece le ricerche per ampiezza assumono una coda con priorità FIFO, ove l'ultimo nodo visitato viene messo a termine dello *stack*.

```
function GRAPH-SEARCH(problem) returns a solution, or failure
  initialize the frontier using the initial state of problem
  initialize the explored set to be empty
  loop do
    if the frontier is empty then return failure
    choose a leaf node and remove it from the frontier
    if the node contains a goal state then return the corresponding solution
    add the node to the explored set
    expand the chosen node, adding the resulting nodes to the frontier
      only if not in the frontier or explored set
```

Figura 2.1: Esempio di ricerca del cammino minimo per Bucharest con grafo.

```
function TREE-SEARCH(problem) returns a solution, or failure
  initialize the frontier using the initial state of problem
  loop do
    if the frontier is empty then return failure
    choose a leaf node and remove it from the frontier
    if the node contains a goal state then return the corresponding solution
    expand the chosen node, adding the resulting nodes to the frontier
```

Figura 2.2: Esempio di ricerca del cammino minimo per Bucharest con albero.