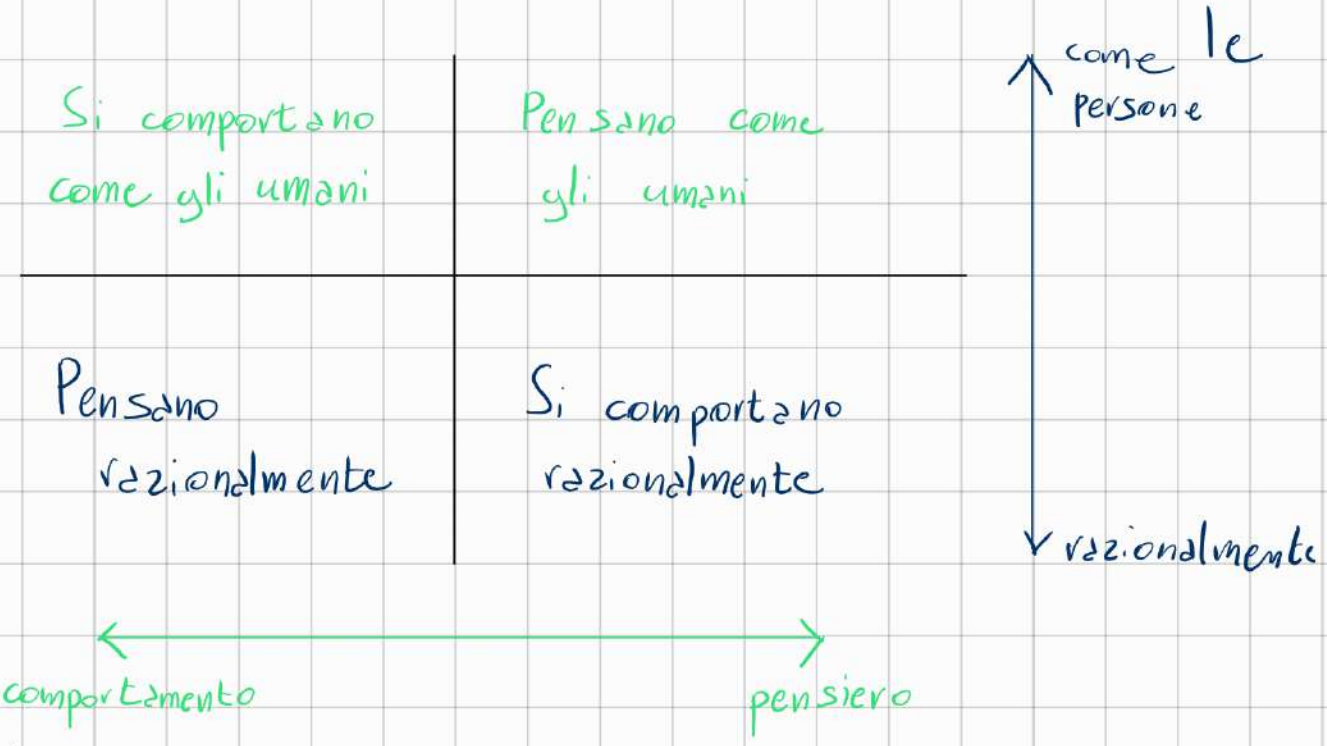
INTELLIGENZA ARTIFICIALE

Disciplina che costruisce macchine che:



2 dimensioni:

- comportamento esterno vs processo di pensiero
- imitazione umana vs razionalità

Si comportano umanamente:

macchine che superano il Test di Turing: un interlocutore umano non è in grado di distinguere tra macchina e uomo

Devono essere in grado di processare l'informazione in input e di interagire con l'ambiente esterno

Pensano come le persone:

tentativo di costruire una mente umana: scienze cognitive

Pensano razionalmente:

pensare in modo corretto: logica, fare inferenze corrette

Nel mondo reale c'è incertezza: probabilità

Comportarsi razionalmente:

creazione di un agente razionale

- autonomo
- percepisce l'ambiente
- percepisce il tempo
- si adatta ai cambiamenti

- si adatta ai cambiamenti
- crea e persegue un obiettivo

Agisce per ottenere il risultato ottimale:
- un obiettivo
- una misura di successo

Costruzione di agenti che fanno la cosa giusta

WEAK AI

Macchine che si comportano in modo intelligente, ma in realtà lo fanno solo simulando

Non possiede intelligenza, coscienza e comprensione.

Eccelle in compiti precisi e limitati

STRONG AI (GENERAL AI, HUMAN-LEVEL AI)

Macchine che possiede capacità cognitive umane, come ragione, coscienza e capacità di ragionare su una vasta gamma di argomenti

STORIA DELL'AI

THE GOLDEN AGE 56-74

Dartmouth Summer meeting → nascita ufficiale dell'AI

Rapida crescita e eccitazione

Argomenti:
- task che dimostrano intelligenza (puzles, giochi...)
- problem solving
- NL
- ragionamento automatico

Applicate a dei mondi giocattolo

PRIMA AI WINTER 70-80

Le soluzioni nei mondi giocattolo non scalano

Alchemy and AI -> l'AI ha successo su problemi semplici e fallisce in quelli complessi
Lighthill report -> problema dell'esplosione combinatoria. Seguirono tagli ai fondi

SISTEMI ESPERTI 69 - 86

Lo sviluppo continua anche durante l'inverno
Knowledge-based AI, basata su regole che catturano la conoscenza umana e inferenza basata sulla logica
The Cyc project: si tenta di creare un AGI, con KB ~ 500K regole codificate manualmente

SECOND AI WINTER 85 - 90

Finisce il boom dei sistemi esperti:
- mancanza di senso comune
- incapacità della logica di gestire l'incertezza
- mancanza di apprendimento

ROBOT, AGENTI, RAZIONALITÁ 90 - OGGI

Non si usano più Filosofia, scienze cognitive o logica, ma probabilità, statistica e economia

INCERTEZZA E RAGIONAMENTO PROBABILISTICO

Reti Bayesiane e tecniche per gestire l'incertezza

RETI NEURALI

Proposte da Rosenblatt nel 1960 -> percettrone
Vengono introdotte le reti multilayer
Avvento dei big data:
- benchmark migliori
- aumentare i dati > migliorare l'algoritmo
- internet = enormi quantità di dati in NL

DEEP LEARNING 2011 - OGGI

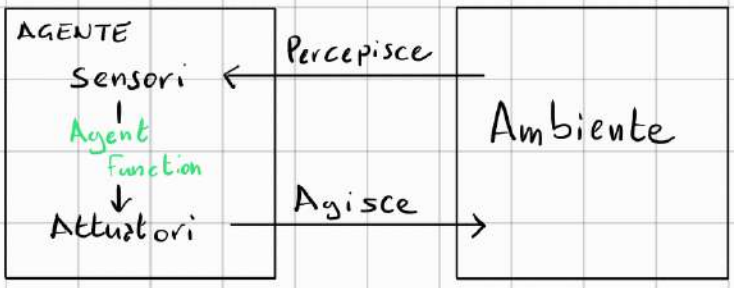
Algoritmi migliori e hardware più potente
Performance sovrumane nei task di visione
- ImageNet
- DeepMind (atari)
- AlphaZero e AlphaGo Zero

AGENTI RAZIONALI

AGENTI RAZIONALI

Un agente è un' entità in grado di percepire e agire

Un agente razionale è un agente che massimizza la sua utilità attesa



L' agent function mappa una sequenza di percezioni in azioni e viene implementata dall' agent program

Per ogni sequenza di percezioni, un'agente razionale seleziona le azioni che massimizzano una misura di performance

TABLE DRIVEN AGENT

Usa una lookup table per associare le percezioni alle azioni

P = insieme di possibili percezioni T = # totale di percezioni

Entry della tabella = $\sum_{t=1}^{T} |P|^t$ → impossibile da memorizzare

TASK ENVIROMENT

Il problema per il quale l' agente trova una soluzione. Formato da:

- Performance mesaure
- Enviroment
- Actuators
- Sensors

Proprietà del task enviroment OMDESCC

1) completamente vs parzialmente osservabile

   I sensori danno accesso allo stato completo dell'ambiente, non serve memorizzare lo stato dell'ambiente

   Parte dello stato non è osservabile

2) single vs multi-agent

   Quando l'agente A deve vedere l'oggetto B come un altro agente? quando il comportamento di B massimizza una performance che dipende dal comportamento di A (es. scacchi: B massimizza una performance, che corrisponde a minimizzare quella di A)

3) deterministico vs non deterministico

STATO CORRENTE -> AZIONE -> STATO SUCCESSIVO

Le situazioni reali sono stocastiche:

- non deterministiche
- probabilistiche

4) episodico vs sequenziale

Le azioni correnti non sono influenzate dalle precedenti, l'agente non deve pensare alle conseguenze delle sue azioni

5) statico vs dinamico

L'ambiente può evolvere mentre l'agente prende decisioni

semi dinamico: l'ambiente non cambia, ma misura della performance sì

6) discreto vs continuo

Riguarda lo stato, il tempo, le azioni e le percezioni

7) conosciuto vs sconosciuto

le regole dell'environment e la misura della performance potrebbero essere sconosciute all'agente

TIPI DI AGENTE

SIMPLE REFLEX AGENT

Seleziona l'azione sulla base delle percezioni attuali ignorandone la storia

situation - action rules ≡ if - then

L'ambiente deve essere completamente osservabile

MODEL-BASED REFLEX AGENT

Hanno uno stato interno che dipende dalla storia delle percezioni e che modella l'enviroment

Hanno un transition model che modella come l'enviroment cambia nel tempo e come si deve riflettere sulle azioni dell'agente

GOAL-BASED AGENT

L'agente punta al raggiungimento di una situazione desiderata

UTILITY-BASED AGENT

L'obiettivo é binario

utility function: misura quanto lo stato attuale é desiderabile

RAPPRESENTAZIONI DELLO STATO

ATOMICO

Ogni stato dell'enviroment é indivisibile

FACTORED

Ogni stato ha un numero finito di attributi con un valore

STRUCTURED

L'enviroment é fatto di elementi relazionati tra loro

PROBLEM SOLVING AGENT

Quando l'azione corretta non é immediata, bisogna guardare avanti una sequenza di azioni, pianificando l'azione utilizzando un algoritmo di search

ALGORITMI DI SEARCH

Assumono:

- stati e azioni discrete
- enviroment osservabile
- transition model deterministico e conosciuto
- stati atomici

Un problema di search consiste in:

- uno spazio degli stati S (insieme di stati possibili)
- uno stato iniziale s_0
- azioni $A(s)$ $\forall s \in S$
- un transition model $Result(a, s)$, $a \in A(s)$, $s \in S$
- un stato obiettivo $G(s)$, raggiunto allo stato s
- il costo delle azioni

Una soluzione é una sequenza di azioni che raggiungono lo stato obj

Il grafo dello spazio degli stati é una rappresentazione di un search problem:

- i nodi sono gli stati
- gli archi diretti sono le transizioni, pesati con l'azione e il costo

Ogni stato appare una sola volta. Una soluzione é un cammino

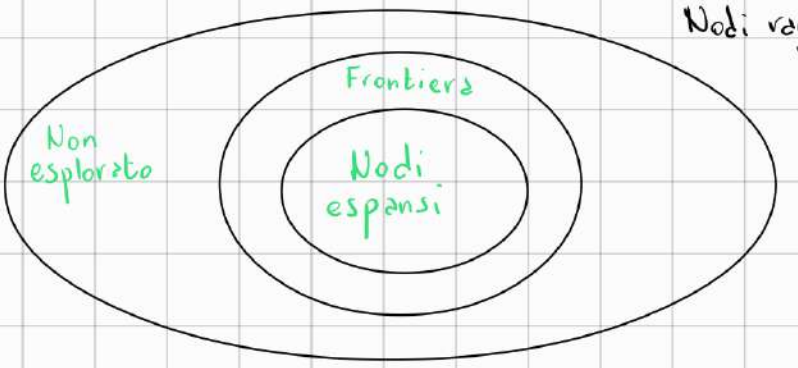
Gli algoritmi di search analizzano un search tree che ha come radice lo stato iniziale e cercano di trovare un cammino che raggiunga l'obiettivo

Se un grafo dello spazio degli stati é finito non é detto che il search

tree associato lo sia

RICERCA SISTEMATICA

Un algoritmo di search deve essere sistematico, garantendo il raggiungimento di tutti i nodi connessi a S_0



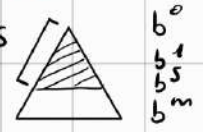
DEPTH-FIRST SEARCH

Espandere i nodi in profondità, la frontiera è uno stack LIFO

- completo: no, m può essere infinito
- ottimo: no, trova la soluzione migliore a sinistra
- tempo: se m è finito $O(b^m)$
- spazio: la frontiera occupa $O(bm)$, lunghezza del cammino da root

BREADTH-FIRST SEARCH

Espandere il nodo meno profondo, la frontiera è una coda FIFO



- completo: s è finito se esiste una soluzione
- ottimo: se i costi sono uguali
- tempo: se la soluzione è a profondità s $O(b^s)$
- spazio: la frontiera contiene l'ultimo tier $O(b^s)$

UNIFORM COST SEARCH (ALG. DIJKSTRA)

$g(n)$ è il costo del cammino da root a n, si espande il nodo con $g(n)$ minore, la frontiera è una coda prioritaria ordinata per $g(n)$

BIDIRECTIONA SEARCH

Inizia dal root e dall'obiettivo, quando le frontiere si intersecano

$b^{m/2} + b^{m/2} << b^m$

INFORMED SEARCH

Si usa un euristica $h(n)$ per stimare la distanza di n dall'obiettivo

$h(n)$ si può solo ottenere dalla conoscenza del mondo

GREEDY BEST-FIRST

Espandere il nodo nella frontiera con $h(n)$ minore

Non è garantita la soluzione ottima

A* SEARCH

Espande il nodo n con più probabilità di essere sul cammino ottimo ovvero il nodo con $g(n) + h^*(n)$ minore, $h^*(n)$ è il costo ottimo da n all'obiettivo e viene stimato da $h(n)$

La frontiera è una coda prioritaria ordinata per $f(n) = g(n) + h(n)$

È importante che $0 \leq h(n) \leq h^*(n)$, $h(n)$ sia ammissibile

AND-OR TREES

Si usa quando le azioni non sono deterministiche

- nodo or: viene scelta una delle possibili azioni
- nodo and: l'ambiente sceglie il risultato dell'azione, l'algoritmo deve tenere conto di tutte

Una soluzione è un sotto-albero che ha un obiettivo ad ogni foglia e include tutte le diramazioni and

GIOCHI

Task environment multi-agenti

Un algoritmo calcola una strategia che determina la mossa migliore per ogni stato

MINIMAX

Due agenti in competizione. Si assume che si giochi in modo ottimo

Il gioco è un grafo dove ogni nodo è uno stato e gli archi sono una mossa

A nodo foglia (stato finale del gioco) è associata una funzione di utilità

$$\text{MINIMAX}(s) = \begin{cases} \text{UTILITY}(s, \text{MAX}) & \text{se } s \text{ è uno stato finale} \\ \max_{a \in \text{Actions}(s)} \text{MINIMAX}(\text{RESULT}(s)) & \text{se deve muovere MAX} \\ \min_{a \in \text{Actions}(s)} \text{MINIMAX}(\text{RESULT}(s)) & \text{se deve muovere MIN} \end{cases}$$

È esaustivo come una DFS, tempo $O(b^m)$

α-β PRUNING

Ottimizzazione del MINIMAX che riduce il numero di nodi

α è il valore massimo per MAX garantito fino ad ora

β è il valore minimo per MIN garantito fino ad ora

P è il valore minimo per min
Quando $\beta \leq \alpha$ il ramo viene potato
Può ridurre la complessità fino a $O(b^{m/2})$

KNOWLEDGE-BASED SYSTEMS

Usare rappresentazioni esplicite della conoscenza per ragionare e prendere decisioni

KNOWLEDGE BASE

Insieme di Frasi in linguaggio Formale di un dominio specifico apprese autonomamente dalle percezioni o impartite a priori

INFERENCE ENGINE

Algoritmo che risponde alle domande, non è un semplice task di retrieval

Knowledge level: metodo per descrivere un'agente in base a quanto sa

LOGICA

Strumento utilizzato per:

- memorizzare le Frasi della KB
- derivare nuove frasi che seguono logicamente quelle già presenti

LOGICA PROPOSIZIONALE

Analizza la verità delle proposizioni e delle relazioni logiche tra di esse
Una proposizione è una dichiarazione che può essere o vera o Falsa

INFERENZA

- IMPLICAZIONE $\alpha \models \beta$: β deriva da α, in ogni possibile mondo se α è vero lo è anche β
- PROVE: dimostrazione di implicazione tra α e β
  - model checking: $\forall$ possibile mondo se α è vero lo deve essere β
  - theorem proving: trovare una sequenza di passaggi che portano da α a β

FORWARD CHAINING

Utilizza Modus Ponens per generare nuovi Fatti:
$X_1 \wedge X_2 \wedge \ldots \wedge X_n \Rightarrow Y$, dati $X_1, X_2, \ldots, X_n$ inferisci Y
Questa regola viene applicata aggiungendo Fatti finché ce ne sono
La KB deve contenere conoscenze definite:

- congiunzione di simboli => simbolo, o
- un solo simbolo x vero

Viene eseguito in O(n)

AGENTI BASATI SULLA LOGICA PROPOSIZIONALE

Agente che rappresenta la propria conoscenza con una KB, dalla quale può derivare o inferire nuovi fatti

FRAME PROBLEM

Gli assiomi specificano solo cosa cambia, ma non cosa non cambia

Si potrebbe risolvere con regole del genere:

$Forward^t \Rightarrow (HaveArrow^t \Leftrightarrow HaveArrow^{t+1})$

Se ho la freccia al tempo t, ce l'ho anche al tempo t+1

- $\forall$ azione, $\forall$ t, se ho m azioni e n fluenti ho $O(mn)$ fluenti

La soluzione è scrivere assiomi dello stato successore, che descrivono come cambiano i fluenti:

$F^{t+1} \Leftrightarrow ActionCausesF^t \vee (F^t \wedge \neg ActionCausesNotF^t)$

- F^{t+1}: fluente
- $ActionCausesF^t$: azione che causa il fluente
- F^t: il fluente valeva al tempo t
- $ActionCausesNotF^t$: azione che causa il non valere più del fluente

QUALIFICATION PROBLEM

Nel mondo reale le cose non sono così precise (eccezioni ∞)

La logica non ha una soluzione a questo problema

LOGICA DEL PRIMO ORDINE

Predicati: simboli che rappresentano proprietà o relazioni tra oggetti es. P(x)

Funzioni: simboli che rappresentano operazioni su oggetti

Variabili: rappresentano gli oggetti del dominio

Quantificatore universale e esistenziale

Esempio:

$B_{1,1} \Leftrightarrow P_{1,2} \vee P_{2,1}$

$B_{1,2} \Leftrightarrow P_{1,1} \vee P_{2,2} \vee P_{1,3} \dots$

$\longrightarrow \forall x,y\ B(x,y) \Leftrightarrow P(x+1,y) \vee P(x-1,y) \vee P(x,y+1) \vee P(x,y-1)$

KNOWLEDGE ENGINEERING

Il processo di costruzione di un KB

1) identificare il problema
   a quali domande la KB deve rispondere e quali fatti disponibili
2) acquisizione della conoscenza
3) decidere il vocabolario di predicati / costanti / funzioni
   si crea un' ontologia specifica
4) codificare la conoscenza del dominio
   scrittura degli assiomi
5) codificare l'istanza del problema
   scrivere i fatti
6) Inferenza
   partendo dai fatti e usando gli assiomi, derivare nuovi fatti
7) debug e valutazione
   incorrettezza, incompletezza o mancanza di assiomi

RETI SEMANTICHE

Fa uso di oggetti e categorie:
- archi etichettati indicano relazioni e categorie
- i nodi membri di una categoria

Permette il ragionamento per ereditarietà e eccezione
È traducibile in frasi logiche

CLIPS

Linguaggio di programmazione rule-based per la creazione di sistemi esperti:
- assert permette di asserire un fatto
- facts permette di vedere i fatti asseriti fino a quel momento
- retract permette di eliminare un fatto
- clear cancella tutti i fatti (riparte da 0)

esempi:
```
(assert (coord 1 2))
(assert (sum (+ 1 2)))
```
- retract * come clear ma riparte da n
- defrule definisce una regola

es. condizione risultato

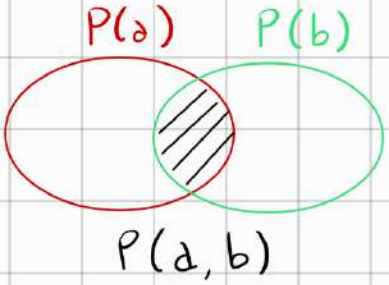
(defrule duck (animal-is duck) => (assert (sound quack)))

- agenda mostra le regole attive (non le esegue)
- run esegue le regole attive (asserisce nuovi fatti)
- variabili ?var $?var → variabili multiple
- deffunction

INCERTEZZA

La logica non basta → qualification problem

PROBABILITÀ CONDIZIONATA



$$P(a \mid b) = \frac{P(a,b)}{P(b)}$$

probabilità che si verifichino entrambi

INFERENZA PROBABILISTICA

Calcolare una probabilità a partire da un modello probabilistico

calcolare la probabilità di una variabile data un'evidenza $P(Q \mid e)$

la probabilità cambia all'aggiunta di nuova evidenza

Esempio:

$P(\text{airport on time} \mid \text{no accidents}) = 0.9$

$P(\text{airport on time} \mid \text{no accidents}, 5\ a.m.) = 0.95$

$P(\text{airport on time} \mid \text{no accidents}, 5\ a.m., \text{rain}) = 0.8$

TEOREMA DI BAYES

$$P(Q \mid e) = \frac{P(e \mid Q)\,P(Q)}{P(e)}$$

- costruire una condizione dal suo opposto
- descrive un aggiornamento $P(Q) \rightarrow P(Q \mid e)$

$$P(\text{causa} \mid \text{effetto}) = \frac{P(\text{effetto} \mid \text{causa})\,P(\text{causa})}{P(\text{effetto})}$$

INFERENZA PER ENUMERAZIONE

Dato $P(X_1 \dots X_n)$, vogliamo calcolare $P(Q \mid e)$:

- selezionare le righe consistenti rispetto l'evidenza e
- sommare le righe per ottenere la probabilità congiunta
- normalizzare $P(Q \mid e) = \alpha\, P(Q \mid e)$

INDIPENDENZA

Due variabili X e Y sono indipendenti se

$\forall x, y \quad P(x, y) = P(x) P(y)$

$P(x|y) = P(x) \qquad P(y|x) = P(y)$

L'indipendenza è rara, è più comune l'indipendenza condizionata

X è cond. ind. da Y dato Z <=>

$P(x|y,z) = P(x|z) \quad P(y|x,z) = P(y|z)$

$P(x, y|z) = P(x|z) P(y|z)$

RETI BAYESIANE

Tecnica per rappresentare distribuzioni congiunte complesse con semplici distribuzioni condizionate

Sono dei DAG:

- nodi variabili
- archi interazioni

Per ogni nodo è data una distribuzione di probabilità condizionata CPT (Conditional Prob. Table) dove ogni riga è una distrib. del figlio dato il valore dei genitori

Ogni variabile è condizionalmente indipendente dei suoi non-discendenti

MACHINE LEARNING

SUPERVISED LEARNING: il dataset contiene sia le feature che le label

- classificazione
- regressione

UNSUPERVISED LEARNING: il dataset non contiene label

- clustering

NEURAL NETWORKS

I pesi sono inizializzati casualmente

- FORWARD STEP: la rete viene attivata su un'esempio e viene calcolato l'errore per ogni neurone
- BACKWARD STEP: l'errore viene usato per aggiornare i pesi. L'errore è propagato all'indietro e viene calcolata la derivata parziale di ogni neurone

GENERATIVE AI

GENERATIVE AI

GAN

Due reti neurali competono tra loro, dove il guadagno di una corrisponde alla perdita dell'altro

LLMs

Addestramento:

- SSL: il modello non usa etichette fornite dall'utente, ma le genera autonomamente.
  Il modello cerca di prevedere la parola successiva mascherata
  word embedding: rappresentare le parole in modo che parole simili abbiano valori simili
- SL: il modello prevede una parola mascherata data una serie di parole. L'errore viene utilizzato per aggiornare i pesi
- RL: il modello viene raffinato utilizzando il feedback umano

DECISION TREES

Serie di decisioni binarie che portano ad una foglia

Come dividere? Massimizzare la purità del nodo

ENTROPIA -> misura della casualità (impurità) dei dati

$$H(p_1) = -p_1 \log_2(p_1) - (1-p_1)\log_2(1-p_1)$$

INFORMATION GAIN -> misura l'efficacia di un attributo nel dividere i dati in sottoinsiemi omogenei

$$IG = H(p_1) - \left(\frac{\#left}{n} H(p_{1left}) + \frac{\#right}{n} H(p_{1right})\right)$$

Si sceglie la feature con l'information gain più alto

Quando si ferma lo split?

- quando un nodo è puro al 100%
- dopo un certo numero di split
- quando IG è < a un certo threshold
- quando il numero di campioni in un nodo < a un certo threshold

Se le features non sono binarie si usa one hot encoding

Es. ear shape $\in$ {pointy, Floppy oval} = pointy $\in \{0,1\}$, oval $\in \{0,1\}$...

Con feature continue vengono scelti i threshold che massimizzano il gain

REGRESSIONE CON DECISION TREE

Le foglie sono variabili continue
L'impurità viene calcolata utilizzando la varianza

$$IG = \sigma^2_{root} - \left(\frac{\#left}{n} \sigma^2_{left} + \frac{\#right}{n} \sigma^2_{right} \right)$$

TREE ENSEMBLE

Gli alberi sono sensibili ai cambiamenti nel dataset
Metodi per combinare insieme più alberi per migliorare le prestazioni

RANDOM FOREST

Sample with replacement -> campionamento con reinserimento

Dato un dataset $|m|$:

da $b = 1$ a B:

usando il sampling with replacement creo un nuovo dataset di m elementi.
Se i dati hanno n feature, ne scelgo un sottoinsieme di $K < n$ elementi e addestro un albero

XGBOOST

Dato un dataset $|m|$:

da $b = 1$ a B:

usando il sampling with replacement creo un nuovo dataset di m elementi. La probabilità per ogni sample di essere estratto è proporzionale all'errore di classificazione dell'albero precedente

DECISION TREE VS NN

- Decision tree:
  - funzionano solo con dati strutturati
  - veloci da addestrare
  - interpretabili
- Neural network:
  - funziona su tutti i tipi di dati
  - lenti da addestrare
  - transfer learning

RECOMENDATION SYSTEMS

Si usa una matrice di valutazione con gli utenti sulle colonne e gli elementi sulle righe

CONTENT BASED RS

Si utilizzano le caratteristiche degli elementi per predirre le preferenze

x^i è il vettore delle caratteristiche dell'elemento i

θ^j è il vettore delle preferenze dell'utente j

La valutazione dell'utente i per l'elemento j si calcola $(\theta^j)^T x^i$

L'obiettivo è minimizzare la differenza tra i rating previsti e reali

$$\min_{\theta^j} \frac{1}{2m^j} \sum_i r(i,j)\left((\theta^j)^T x^i - y(i,j)\right)^2$$

- m^j: # elementi valutati da j
- $r(i,j)$: 1 = elemento i valutato da j
- $y(i,j)$: rating target

FILTRAGGIO COLLABORATIVO

Utenti simili tendono ad apprezzare elementi simili e viceversa

Non sempre è facile estrarre informazioni sul contenuto degli elementi

θ^j e x^i sono sconosciuti, ma uno può essere calcolato dall'altro

Processo iterativo:

Stimo $\theta \rightarrow$ calcolo $x \rightarrow$ calcolo $\theta \rightarrow$ calcolo $x \rightarrow \dots$

Questo processo converge a θ e x

LATENT FACTOR MODEL

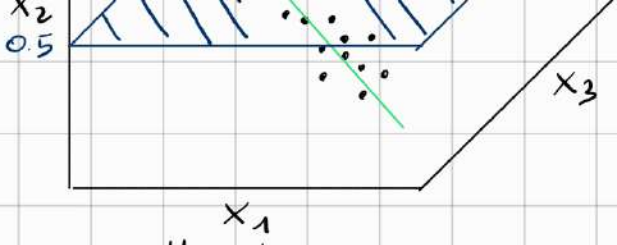
Sfrutta il fatto che righe e colonne della matrice sono correlate

L'obiettivo è stimare la matrice dei dati utilizzando tecniche di riduzione della dimensionalità.

Se i rating sono correlati, si troveranno lungo una retta (latent vector)

Dato che i punti possono essere rappresentati da una sola retta, la matrice avrà $\sim$ rango 1 $(\min(n,m))$

Basta specificare un solo rating (x_2) per stimare gli altri ovvero l'intersezione del piano con il latent vector



La fattorizzazione permette di approssimare una matrice

Ogni matrice R $m \times n$ di rango K si può esprimere come:

$$R = UV^T \quad U\ m \times K \quad V\ n \times K$$

L' i-esima riga di U contiene K elementi che rappresentano l'affinità con il K-esimo concetto, detto anche user factor

Similmente per la matrice V, detta anche item factor

Il rating stimato dell'utente i per l'item j è: $\hat{r}_{ij} = \sum_{K} u_{iK} \cdot v_{jK}$

La fattorizzazione di R può essere formulata come un problema di minimizzazione:

$$\min_{U,V} \frac{1}{2} \|R - UV^T\|^2$$

REINFORCEMENT LEARNING

È una tecnica di ML dove un' agente impara a comportarsi in un' ambiente eseguendo azioni e vedendone il risultato

Reward hypothesis: ogni obiettivo può essere formulato come il risultato della massimizzazione cumulata di ricompense

Ad ogni step t l' agente:

- riceve dalle osservazioni O_t e reward R_t
- esegue un' azione A_t

L' ambiente:

- riceve un' azione A_t
- emette osservazioni O_{t+1} e reward R_{t+1}

Stato dell' environment = osservazione

- completamente osservabile
- parzialmente osservabile

Action space

Insieme delle possibili azioni
- continuo
- discreto

REWARD

Un reward è uno scalare che indica quanto bene l'agente va in t
L'agente massimizza il reward cumulato

$$G_t = \sum_{k}^{T} R_{t+k+1}$$

Nella realtà i reward non sono lineari

Discount cumulative reward $G_t = \sum_{k}^{\infty} \gamma^k R_{t+k+1}$, $\gamma \in [0, 1)$
- più alto è γ, più l'agente dà importanza ai reward a lungo termine
- più basso è γ, più l'agente dà importanza ai reward a breve termine

EXPLORATION VS EXPLOITATION

Trade-off tra trovare informazioni sull'ambiente e massimizzare il reward

POLICY

Rappresenta la funzione che, dato uno stato, dice che azioni intraprendere

$$\text{STATO} \rightarrow \pi(\text{STATO}) \rightarrow \text{AZIONE}$$

Il RL allena modelli per trovare π^*, la policy che massimizza l'outcome
Si può trovare:
- direttamente, insegnando all'agente quale azione fare in base allo stato (policy-based method)
- indirettamente, insegnando all'agente quali sono gli stati ottimali e eseguire le azioni per raggiungerli (value-based method)

POLICY-BASED METHOD

La funzione mappa per ogni stato la migliore azione (deterministica) o una distribuzione di probabilità per ogni azione (stocastica)

VALUE-BASED METHOD

La funzione mappa per ogni stato il valore atteso di essere in quello stato dicendo il reward massimo ottenibile per ogni stato
Questa funzione viene utilizzata per scegliere l'azione che porta ad uno stato con un reward maggiore

Q-LEARNING

Q-table: lookup table che associa il reward futuro massimo di ogni possibile

azione in ogni possibile stato, permettendo all'agente di scegliere la migliore azione in ogni stato

Algoritmo:

1) La Q-table (m stati x n azioni) viene inizializzata a 0
2) Ripetere per x step o quando si termina l'addestramento:
   3) scegliere un'azione a nello stato corrente s in base al Q-value attuale. All'inizio:
      - se si sceglie un'azione casuale, si fa exploration
      - se si sceglie un'azione in base al Q-value, si fa exploitation

   All'inizio del training è meglio fare più exploration
   4) si ottiene un nuovo stato s' e un reward r.

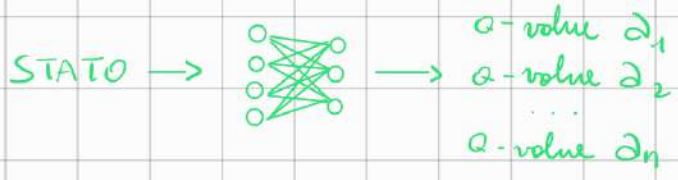
   Si aggiorna il valore della funzione Eq di Bellman

$$Q(s,a) = Q(s,a) + \alpha[R(s,a) + \gamma \max(Q(s',a')) - Q(s,a)]$$

- Q-value corrente
- reward per aver preso quell'azione in quello stato
- reward massimo atteso dato il nuovo stato s' e tutte le possibili nuove azioni

DEEP Q-LEARNING

La Q-table può essere enorme richiedendo molta memoria
Una NN approssima la funzione Q-value



È un problema di regressione: MSE tra Q-value predetta e target

$$Loss = \frac{1}{2}\left[r + \max_{a_{t+1}}(Q(s_t, a_{t+1}, \theta_{t-1})) - Q(s,a,\theta)\right]^2$$

Dato che la rete calcola sia il target che la predizione ci può essere divergenza, quindi si usa una target network che ha la stessa struttura della rete primaria ma con i pesi freezzati. Ogni n iterazioni i parametri vengono copiati dalla rete principale

ESPERIMENTO MENTALE STANZA CINESE

Una persona che non sa il cinese è dentro una stanza con un manuale in inglese che descrive come tradurre. Delle persone cinesi possono mandare delle domande all'interno della stanza. La persona all'interno, grazie al manuale, può rispondere alle domande e produrre risposte in cinese

Contesta il concetto di strong AI:

- sintassi vs semantica: seguendo regole sintattiche è possibile manipolare simboli senza comprenderne la semantica
- un computer può comportarsi come se capisse ma non c'è vera conoscenza, criticando il test di Turing
- l'IA non raggiungerà una vera coscienza umana e intelligenza