Appunti Fondamenti di Intelligenza artificiale

Lezione 01

L’intelligenza artificiale mira a costruire delle macchine intelligenti che possano capire il contesto, fare predizioni e adattarsi all’ambiente.

Definizione intelligenza artificiale:

1) Pensare umanamente replicare il pensiero umano su una macchina, processo decisione apprendimento e altro , cosa impossibile. L’idea è determinare come l'uomo crea pensieri (meccanismi).
2) Pensare razionalmente, replicare le facoltà mentali attraverso moduli computazionali , processi che rendono possibile percepire, ragionare e agire
3) Agire umanamente eseguire attività che vengono svolte dall'uomo con intelligenza. L'intelligenza può essere misurata guardando il comportamento esterno e non il ragionamento che ha portato a quel comportamento.
4) Agire razionalmente , intelligenza computazionale progettazione agenti intelligenti. Un agente razionale agisce in modo da ottenere il miglior risultato o quasi.

Lezione 02

Agente: Sistema che percepisce il suo ambiente attraverso dei sensori ed agisce su di essi tramite degli attuatori.

Sequenza percettiva: Storia completa di quello che l'agente ha percepito nella sua esistenza, dove per percezione si intende un input avuto in un dato istante. Questa sequenza può influenzare una sua azione, ciò che invece non ha percepito non può farlo.

Misura di prestazione: Progettata sulla base dell'effetto che si desidera osservare e non su come si dovrebbe comportare l'agente.

La razionalità di un agente dipende:

- Dalle sue prestazioni.
- Dalla conoscenza pregressa che ha.
- Dalle azioni che può compiere.
- Dalla sequenza percettiva fino all'istante corrente.

Con questa razionalità dovrebbe massimizzare il valore atteso della sua misura di prestazione.

Funzione agente: Descrive la corrispondenza tra una sequenza percettiva ed una specifica azione. Mappa quello che l'agente sa e può fare.

Un agente razionale per massimizzare il risultato atteso potrebbe fare dell' "information gathering", operazioni che hanno come unico scopo quello recuperare delle informazioni sull'ambiente.

Ambiente: Istanza di un problema di cui gli agenti razionali rappresentano le soluzioni.

Un ambiente si descrive da:

P: Misura di prestazione adottata per valutare l'operato di un agente
E: Descrizione degli elementi che formano l'ambiente
A: Attuatori disponibili all'agente per intraprendere le azioni

S: I sensori attraverso i quali riceve gli input percettivi

Caratterizziamo gli ambienti in:

- Completamente osservabili: I sensori danno continuo acceso allo stato completo dell'ambiente.

- Parzialmente osservabili: L'agente ha acceso a solo parte delle info.

- Deterministico: Quando lo stato successivo dell'ambiente è determina dallo stato corrente e dall'azione eseguita dall'agente.

- Stocastico: Se lo stato successivo può dipendere da elementi casuali.

- Episodico: Ogni interazione dell'agente con l'ambiente è separata e indipendente dalle altre. L'agente prende decisioni basate esclusivamente sull'episodio in corso.

- Sequenziale: Ogni decisione può influenzare le successive.

- Statico: L'ambiente è invariato mentre un agente sta deliberando.

- Dinamico: L'ambiente cambia mentre un agente sta deliberando.

- Discreto: L'ambiente fornisce numero limitato di percezioni di azioni distinte.

- Singolo: L'ambiente consente la presenza di un singolo agente.

- Competitivo: In un ambiente multi-agente il comportamento di uno è volto massimizzare o minimizzare la prestazione dell'altro.

- Cooperativo: I due agenti puntano a massimizzare o minimizzare la stessa misura di prestazione

In un ambiente deterministico e osservabile non è detto che esiste un numero finito di azioni tra cui l'agente può scegliere, inoltre non è detto che l'agente sappia sempre quali stati saranno raggiunti da ciascuna azione.

Il compito dell'intelligenza artificiale è progettare il programma agente che implementa la funziona agente. Il programma agente prende in input solo la percezione corrente mentre la funzione l'intera storia percettiva.

Tipi di agenti:

- Agenti reattivi semplici: Fanno azioni sulla base della percezione corrente ignorando la storia percettiva pregressa. Funzionano bene quando l'ambiente è completamente osservabile, altrimenti potrebbero portare a cicli infiniti. Per risolvere questo problema si definisce una componente casuale che fa compiere un'azione casuale.

- Agente reattivi basati sul modello: Un agente con due tipi di conoscenza, come evolve il mondo indipendentemente dal suo stato e informazioni sull'impatto delle sue azioni sull'ambiente.

- Agenti basati sugli obiettivi: Agente che aggiunge ,al modello, informazioni sugli obiettivi che si desiderano raggiungere. Conoscere lo stato corrente dell'ambiente non sempre basta a decidere cosa fare, gli obiettivi aiutano l'agente a compiere azioni più intelligenti. Un agente basato su obiettivi si basa su una rappresentazione atomica degli stati del mondo e non conosce la rappresentazione interna di uno stato.

- Agenti basati sull'utilità: Gli obiettivi consento di esprimere i risultati in buoni e cattivi, nella realtà abbiamo anche quelli "desiderabili". Assegniamo quindi un grado di desiderabilità.

- Agenti capaci di apprendere: L'apprendimento permette agli agenti di operare in ambienti sconosciuti per poi diventare competenti.

Lezione 03

La conoscenza può essere modificata quindi è considerabile come un agente flessibile.

Gli agenti basati sugli obiettivi si chiedono quali sono le conseguenze dell'azione e se quell'azione li "soddisfa".

Agenti Risolutori di problemi: Si pongono uno o più obiettivi da raggiungere ed eseguono azioni che massimizzano il loro guadagno.

Il primo passo consiste nella formulazione dell'obiettivo, basata sulla situazione corrente e sulla misura di prestazione dell'agente.

Definizione obiettivo: Insieme ammissibile di stati del mondo, tutti gli stati in cui l'obiettivo è soddisfatto.

L'obiettivo influenza la specificità delle scelte ad esempio se deve arrivare in un aeroporto non decido ogni minimo passo altrimenti si arriverebbe ad una soluzione enorme.

Si procede poi con la formulazione del problema, il processo che porta a decidere quali azioni e stati considerare dato un obiettivo.

Un agente che ha a disposizione diverse opzioni immediate di valore sconosciuto può decidere cosa fare esaminando le azioni future che alla fine porteranno a stati di valore conosciuto.

Ricerca: Processo che cerca una sequenza di azioni che raggiunge l'obiettivo.

La progettazione di un agente è divisa nelle fasi di:

1) Formulazione
2) Ricerca
3) Esecuzione

Mentre l'agente esegue le azioni che ha deciso di fare ignora le proprie percezioni. Questo comportamento è detto a ciclo aperto perché l'agente non aggiorna il suo stato, invece a ciclo chiuso invece l'agente aggiorna costantemente la sua conoscenza.

Un algoritmo segue sommariamente queste azioni:

1) Prende in input la percezione corrente dell'agente.
2) Aggiornato il proprio stato e l'agente formula un obiettivo e un problema.
3) Avvia l'algoritmo di ricerca ed esegue le azioni raccomandate.
4) Se l'algoritmo fallisce non compie nessuna azione.

Problema può essere definito da cinque componenti:

1) Stato iniziale, punto di partenza dell'agente.

2) Azioni, descrizioni delle possibili azioni attuabili dall'agente.
3) Modello di transizione, descrive il risultato di ogni azione attuabile dall'agente nello stato attuale. Con il termine successore si indica uno stato raggiungibile dopo una azione.
4) Test obiettivo, determina se un particolare stato è un obiettivo
5) Costo cammino, determina il costo numerico ad ogni cammino.

Le prime tre definiscono lo stato degli spazi del problema ovvero l'insieme di tutti gli stati raggiungibili da quello iniziale grazie ad una sequenza di azioni.

Soluzione: Sequenza di azioni che portano da uno stato iniziale ad uno stato obiettivo. La qualità è misurata dalla funzione di costo di cammino. La soluzione ottima ha il costo minore.

Il processo di rimozione dei dettagli da una rappresentazione è detta astrazione.

Un'astrazione è:

- Valida se possiamo espandere ogni soluzione astratta in una dettagliata.
- Utile se eseguire ogni azione nella soluzione è più facile che nel problema originale.

Lezione 04

Una soluzione è una sequenza di azioni che formano un albero di ricerca con lo stato iniziale alla radice, i rami rappresentano le azioni e i nodi corrispondono a stati nello spazio degli stati del problema.

Ogni volta si cerca di capire se l'obiettivo è stato raggiunto o meno, poi si procede a considerare le possibili soluzioni generando cosi un nuovo insieme di stati.

Se uno stato può essere ripetuto parliamo di un cammino ciclico.

L'insieme di tutte le foglie che possono essere espanse in un dato punto è detto frontiera.

Cammino ridondate: Un cammino in cui esistono due o più modi per passare da uno stato all'altro. Per evitare cammini ridondati, conserviamo semplicemente i nodi esplorati. In questo modo la frontiera separa il grafo degli stati nella regione esplorata e in quella inesplorata, questo permette che ogni cammino che va dallo stato iniziale ad uno inesplorato debba passare per uno stato della frontiera.

Gli algoritmi di ricerca richiedono una struttura dati per tenere traccia dell'albero costruito.

Per ogni nodo N abbiamo:
- N.stato: Stato dello spazio degli stati a cui corrisponde il nodo.
- N.padre: Nodo dell'albero di ricerca che ha generato il nodo corrente.
- N.azione: L'azione applicata al padre per generare il nodo.
- N.costo-cammino: Costo del cammino che va dallo stato iniziale al nodo.

Una strategia di ricerca è definita secondo la modalità di espansione dei nodi, queste strategie vengono valutate in base a:

- Completezza: Se riesce a trovare una soluzione, se esiste.
- Ottimalità: Se riesce a trovare la soluzione ottima.
- Complessità temporale: Quanto tempo impiega per trovate una soluzione
- Complessità spaziale: Quanta memoria è richiesta per trovare una soluzione.

Dato che il grafo è spesso indefinito per delineare la complessità utilizziamo il fattore di ramificazione, profondità della soluzione a costo minimo e la massima profondità.

Gli algoritmi di ricerca non informata fanno riferimento alle strategie di ricerca che non dispongono di informazioni aggiuntive sugli stati oltre a quello della definizione del problema, possono solo generare successori e distinguere gli stati obiettivo dagli altri.

La principale differenza tra le varie strategie di ricerca consiste nell'ordine in cui vengono espansi i nodi. Per natura gli algoritmi di ricerca non informata non sanno stimare quanto un nodo sia promettente per la risoluzione di un problema.

- Ricerca in ampiezza (BFS): Trova sempre una soluzione se esiste, questa è sempre quella con il cammino più breve. Si espande il nodo più vicino alla radice. Complessità esponenziale in base alla profondità. Utili per problemi in cui la soluzione è vicina al nodo iniziale.

- Ricerca a costo uniforme: Invece del nodo meno profondo si espande il nodo con costo minimo da quello iniziale. Si fa questo usando delle code a priorità.

- Ricerca in profondità (DFS): Si espande il primo nodo più profondo nella frontiera corrente. Non è completo perché con profondità infinta non si ferma mai inoltre non è ottimo perché può ritornare un nodo lontano dalla radice. Possiamo migliorarlo impostando una profondità massima, ma poi potrebbe non trovare una soluzione.

- Ricerca ad approfondimento iterativo: Il limite viene incrementano fino a quando un nodo obiettivo non viene identificato. Simile alla ricerca in ampiezza perché controlla tutto il livello prima di scendere. Ma in realtà esegue una ricerca in profondità ad ogni ciclo.

- Ricerca bidirezionale: Viene fatta una ricerca in avanti dallo stato iniziale e poi una indietro dallo stato obiettivo.

Lezione 05

Gli algoritmi di ricerca non informata trovano soluzioni senza nessuna informazione aggiuntiva, mentre quelli di ricerca informata sfruttano conoscenze specifiche del problema. Ad esempio, se in un problema di cammino conosciamo in anticipo tutte le distanze.

Funzioni euristiche: Funzione che indica il costo stimato del cammino più conveniente dallo stato di un nodo ad uno stato obiettivo, sono il modo migliore per aggiungere conoscenza ad un algoritmo. Associano al costo accumulato una stima del costo futuro.

- Best-First Greedy: Ad ogni passo cerca sempre di arrivare più vicino all'obiettivo espandendo un nodo grazie alla funzione euristica. Nel fare questo non tiene conto del costo accumulato, per cui potrebbe non considerare cammini alternativi con stima costo maggiore.

- Ricerca A*: Combina una funzione euristica che calcola il costo per arrivare all'obiettivo con il costo accumulato fino a quel momento. Questo garantisce che vengano esplorati prima i nodi che sembrano portare alla soluzione più economica. Espande più nodi e non uno solo come nella Best-First Greedy, grazie ad una coda a priorità. L'ottimalità di questo algoritmo dipende dall'usare un euristica ammissibile che non sbagli la stima del costo per arrivare all'obiettivo. Questa euristica deve essere anche consistente ovvero il costo per raggiungere l'obiettivo da un certo nodo non deve essere superiore al costo per il passo successivo sommato con la stima di andare dal quel nodo all'obiettivo. Questa soluzione richiede l'esistenza di un numero finito di nodi di costo minore o uguale alla soluzione ottima. Questo è un algoritmo ottimamente efficiente. L'algoritmo è completo se la soluzione è raggiungibile ed è ottima.

- Beam Search: Variante della best-first che ad ogni passo mantiene solo i più promettenti. Il numero di nodi che conferma viene detto ampiezza del raggio ed è definita in base ad un'euristica, solo i nodi conservati vengono valutati per la soluzione. Non completo perché potrebbe non trovare una soluzione sopratutto quando il numero dei nodi che mantiene è limitato. Inoltre non risulta essere nemmeno ottimale perché potrebbe tagliare dei cammini che potrebbe essere migliori..

- IDA*: Utilizza l'approfondimento iterativo ma il limite è dato dalla stessa funzione euristica utilizzata in A*. Questo ha un vantaggio nell'occupazione della memoria, memorizza infatti un solo cammino dalla radice ad un nodo foglia. Algoritmi come questi dimenticano ciò che hanno fatto in precedenza con il rischio di espandere più e più volte gli stessi stai visitati in precedenza.

- RBFS*: Cerca di usare al meglio la memoria che si ha liberando i nodi peggiori quando la memoria si è riempita. Memorizza però nel nodo padre il valore del nodo dimenticato. Rigenera inoltre un sotto-albero dimenticato solo quando tutti gli altri cammini promettono di comportarsi peggio. Risulta completo se la soluzione è raggiungibile ed è ottimale se c'è una soluzione ottima raggiungibile. Il problema di questa è che per problemi difficili passa continuamente tra i vari cammini candidati.

Lezione 06

In molti problemi lo stato obiettivo è esso stesso la soluzione del problema indipendentemente da come ci si è arrivati. Questi algoritmi detti di miglioramento iterativo mantengono in memoria solo lo stato corrente e tentano di migliorarlo.

Algoritmi di ricerca "tradizionali": Considerano interi cammini dallo stato iniziale a quello obiettivo ed hanno un complessità temporale e spaziale esponenziale. Inoltre non possono essere sempre applicati.

Algoritmi di ricerca locale: Considerano lo stato corrente con l'obiettivo di migliorarlo e usano poca memoria con una complessità costante e trovano soluzioni sub-ottimali. Non è importante il percorso con cui ci si arriva.

Funzione obiettivo: Gli algoritmi di ricerca locale non hanno un test obiettivo ma affidano le loro azioni a questa funzione che indica quanto la ricerca sta migliorando. Questa funzione è fondamentale per l'efficienza e definisce le soluzioni che possono essere raggiunte da una soluzione in un singolo passo di ricerca dell'algoritmo. La funzione obiettivo è definita dalle possibili mosse che l'algoritmo può effettuare.

Struttura dei vicini: Funzione F che assegna a ogni soluzione dell'insieme di soluzioni un suo sottoinsieme.

La soluzione trovata da un algoritmo di ricerca locale non è detto che sia globalmente ottima.

Caratteristiche dello spazio degli stati:

- Massimo locale: Un picco che rappresenta una soluzione sub-ottimale
- Piatto (plateau): Una regione dello spazio con stati vicini che hanno tutti lo stesso valore
- Spalla: Un plateau che presenta una leggera salita offrendo la possibilità di miglioramento
- Massimo globale: Soluzione ottima al problema
- Ridge: Porzione dello spazio di ricerca che presenta una brusca variazione.

Un algoritmo di ricerca locale completo trova sempre un obiettivo se questo esiste.

Un algoritmo di ricerca locale ottimo trova sempre un minimo/massimo locale.

Hill-Climbing: Ha l'obiettivo di scalare lo spazio di ricerca per trovare un massimo/minimo globale. Ad ogni

passo, il nodo corrente viene rimpiazzato dal vicino migliore, termina quando non ha vicini di valore più alto. Non mantiene un albero di ricerca quindi la struttura dati per il nodo corrente deve solo memorizzare lo stato e il valore della sua funzione obiettivo. Non "sbircia" negli stati immediatamente vicini.

Miglioramenti Hill-Climbing:

- Per migliorarlo si consente all'algoritmo di fare un preciso numero di mosse laterali ad esempio su un plateau per "convincersi" davvero che la soluzione non migliora.

- Stocastico: Sceglie a caso tra tutte le mosse che vanno verso l'alto e non sceglie immediatamente quella più conveniente.

- Riavvio casuale: L'algoritmo può essere eseguito più volte con stati iniziali generati casualmente, aumentando le probabilità di trovare una soluzione globale.

Simulated Annealing: Nello stato iniziale varia molto nel panorama degli stati per poi concentrarsi su un punto specifico del panorama cercando una soluzione ottima in quel contesto. Simile ad Hill climbing ma decide di spostarsi anche nei nodi che peggiorano con la speranza di trovare una soluzione migliore. Si sposta in base ad una probabilità definita con una Temperature che è all'inizio è alta ma che poi scende.

Local Beam Search: Tiene traccia di k stati anziché uno, inizialmente comincia con k stati generati e ad ogni passo genera i loro successori. Si scelgono poi o l'obiettivo o il migliore tra i loro successori. Nelle stocastica non si sceglie tra i migliori k ma a caso.

Lezione 07 e 08

Algoritmi genetici: Definisce gli algoritmi di ricerca. Sono ispirati ai meccanismi evolutivi osservati in natura. Utili in spazi di ricerca grandi.

Ottimizzazione: Farsi guidare da una funzione obiettivo navigando tra le varie possibilità per trovare la soluzione migliore.

Meta-euristica: Non è un algoritmo ma una strategia che permette di fare algoritmi.

Euristica: Funzione che guida lo spazio di ricerca, qui c'è un approccio euristico con cui generiamo l'algoritmo.

Non sono "problem-specific", sono capaci di adattarsi a più problemi, non risolvono una singola classe di problema.
Sono molto flessibili perché sono una famiglia di algoritmi guidata da un approccio euristico che in base a come la definisco cambia.
Non garantisce l'ottimalità dei risultati ma sono soluzioni sub-ottimali.

Un algoritmo genetico evolve una popolazione di individui, non ha una soluzione e la migliora. Ha una serie di soluzioni non ottimali ma che sono soluzioni. L'algoritmo cerca soluzioni sempre migliori guidati da una funzione obiettivo che migliora la famiglia di soluzioni. Queste soluzioni prendono i tratti migliori dei genitori per crearne di migliori.

Gli algoritmi genetici:

1) Codificano gli individui in stringhe, possono non avere una codifica finita, questo aumenta la flessibilità.

2) Non fa evolvere una singola istanza.

3) La funzione obiettivo non ha bisogno di informazioni di basso livello, ma può essere rappresentata con il comportamento esterno dell'algoritmo.

4) Ha elementi di casualità che guidano la ricerca, piccoli elementi per garantire che le popolazioni generate siano sempre più efficaci.

La soluzione migliore ha una probabilità maggiore di essere scelta e contribuire all'evoluzione ma non è detto che venga scelto (Differenza Climbing)

Convergenza: Andare verso l'ottimo, indirizzare soluzioni candidate verso la soluzione ottima. Parliamo di convergenza prematura quando l'algoritmo da risultati troppo presto.

Funzione fitness: Indica per definizione adattabilità quindi si adatta all'ambiente, simile a funzione obiettivo.

Dimensioni della popolazione (insieme di soluzioni candidate), se la dimensione è fissa bisogna scegliere un numero che si adatti al problema, anche se è variabile va comunque indicato il numero massimo.

Dimensioni Mating Pool (insieme di individui ammessi alla fase di produzione che verranno poi riconsiderati), più è alto più esploriamo le soluzioni nell'ambito di ricerca ma se è più grande sarà più lento.

Inizializzazione: Quanta conoscenza abbiamo del contesto in cui stimo operando se non conosciamo molto una scelta casuale è l'unica scelta. Ma qui possiamo generare degli individui che rispettano già delle proprietà del problema d'esame. Se lo guidiamo è più efficiente e ottimo. Fare attenzione nell'inizializzazione perché l'algoritmo non può evolvere soluzioni inammissibili.

- Selezione: Copia di alcuni individui da portare nella successiva generazione.

Algoritmo di selezione: Come scegliamo gli individui che vengono poi evoluti.

1) Metodo casuale ma è sconsigliato perché la casualità può non guidare l'algoritmo ma fargli esplorare a caso lo spazio di ricerca.

2) Un altra scelta è la roulette Wheel dove gli individui vengono scelti in modo proporzionale alla loro fitness.

3) Rank i più forti passano alla fase successiva ma questa non è calcolata con la funzione di fitness ma con un rango e sempre tramite una classifica fatta con la funzione di fitness alla quale viene assegnato un rango, la probabilità viene assegnata dal rango. Si riduce la differenza tra uno molto forte ed uno debole.

4) Troncamento si ordinano gli individui sulla base di fitness e si selezionano i migliori.

5) K-way tournament creare delle coppie che dovranno affrontarsi per rimanere in vita

6) Selezione stocastica, variante di roulette wheel vengono selezionati più individui contemporaneamente.

- Crossover: Accoppiamento il patrimonio genetico dei genitori per creare nuovi da aggiungere alla nuova generazione.

Algoritmo di cross-over:

1) Single point da un punto in poi invertiamo tutti i geni dei cromosomi.

2) Two-point due punti di incrocio, k-point soglia che scegliamo per combinare gli individui.

3) Uniforme ogni gene del figlio viene scelto casualmente tra i geni corrispondenti dei due genitori.

4) Aritmetico combina i valori dei genitori attraverso operazioni matematiche.

5) Ordinato, preserva l'ordine realistico dei geni di un genitore e riempie i resti geni con quelli dell'altro genitore.

6) Ciclo, costruisce il giglio identificando cicli tra i genitori , il primo è preso da un genitore mentre l'altro dall'altro preservando sempre l'ordine.

7) Segmento casuale, si sceglie un segmento casuale della stringa genetica dei genitori e viene scambiato per produrre il figlio

- Mutazione: Modifica casuale di alcune parti del corredo genetico. Importante per mantenere la diversità del patrimonio genetico.

Mutazione:

1) Bitflip (Singolo , Multi), uno o più bit a seconda del caso vengono selezionati e invertiti.

2) Gaussiana, altera i valori dei geni aggiungendo un valore casuale derivato da una distribuzione gaussiana. Variazione "leggera".

3) Swap, dei geni vengono selezionati casualmente e scambiati di posizione.

4) Inversion, viene selezionato un segmento casuale e l'ordine dei geni del segmento viene invertito.

5) Random, un gene selezionato casualmente viene sostituito da un nuovo valore casuale all'interno del dominio permesse dal gene.

6) Permutazione, un sottoinsieme di geni viene selezionato casualmente e il loro ordine viene permutato.

7) Uniformemente distribuita, un gene viene alterato di una quantità casuale derivata da una distribuzione uniforme.

8) Inserimento, un gene viene rimosso dalla sua posizione reinserito casualmente in un'altra.

Frequenza mutazioni:

- Mutazione a scala, man mano che l'algoritmo avanza verso la soluzione finale introduce delle variazioni.

- Mutazione adattiva, qui la mutazione varia dinamicamente in base alla diversità della popolazione o dal progresso dell'algoritmo.

Un algoritmo genetico termina quando abbiamo raggiunto un numero fisso di generazioni, abbiamo raggiunto una soglia di fitness oppure non abbiamo ottenuto nessun miglioramento significativo nelle ultime generazioni.

Un'algoritmo genetico va quindi utilizzato per la sua flessibilità dato che possono essere usati su vaste gamme di problemi, inoltre sono robusti perché sono in grado di gestire problemi complessi con molti parametri dando soluzioni accettabili , inoltre la combinazione di selezione , cross-over e mutazione permettono di esplorare ampiamente lo spazio delle soluzioni.

Un'algoritmo genetico non va invece utilizzato perché potrebbe avere una convergenza lenta o troppo veloce, inoltre ogni generazione nuova deve essere valutata dalla fitness.

Fronte di Pareto: Ogni soluzione rappresenta un compromesso ottimale tra gli obiettivo in caso di algoritmi genetici multi-obiettivo.

Una soluzione A è detta dominata da B se B è meglio di A in un obiettivo e mai peggiore negli altri, si cerca quindi di eliminare le soluzioni dominate.

Le strategie per equilibrare gli obiettivi sono:

- NSGA-II: Classifica le soluzioni basandosi sulla dominanza di Pareto mantenendo un equilibro tra esplorazione dello spazio di ricerca e diversità delle soluzioni.

- SPEA2: Introduce meccanismi per mantenere la diversità della popolazione durante la ricerca delle soluzioni ottimali.

- Strategia dell'archivio: Vengono salvate delle soluzioni Pareto ottimali fino a quando non se ne trovano di migliori.

Modello ad isole: Divide la popolazione in isole ovvero delle sotto-popolazioni dove ciascuna evolve in modo indipendente per un certo numero di generazioni. I migliori individui migrano poi da un isola ad un'altra scambiando informazioni genetiche.

Algoritmi memetcui: Algoritmi ibridi tra GA e altre tecniche di ottimizzazione.

Lezione 09 -> esercitazione

Lezione 10

Nel multi agente l'agente deve selezionare le proprie azioni in relazione a quelle degli altri agenti che sono in gioco e questo può dare luogo a contingenze ovvero scegliere la propria mossa in base a quello che potrebbe fare l'altro.

Gli agenti esaminati sono avversari e mirano a ottimizzare il guadagno personale, chiamiamo queste situazioni giochi.

Teoria dei giochi: Considera ogni agente multi-agente un gioco a prescindere che sia competitivo o meno. Questo perché in ambo i due casi l'influenza è significativa.

Tipologie di giochi (economie): Dipendono dal grado di osservabilità e determinismo dell'ambiente.

I giochi possono essere

- Giochi con informazione perfetta: Gli stati del gioco sono espliciti agli agenti. (contrario imperfette)

- Giochi deterministici: Ogni stato del gioco è determinato unicamente dalle azioni che l'agente compie. (contrario stocastici)

Giochi a somma zero con informazione perfetta: Quando ci sono dei turni, questi turni influenzano la scelta dell'altro agente e sono quei giochi in cui il risultato finale dei due giocatori è uguale di segno opposto. Tutti i giochi in cui c'è uno che vince o pareggio.

Fasi di algoritmo di ricerca con avversari:

- Stallo iniziale (stato iniziale).
- Il giocatore (agente).
- Insieme di azioni (insieme di mosse lecite applicabili).
- Risultato , modello di transizione , in che stato ci troviamo se applichiamo una certa mossa.
- Test di terminazione (stati terminali, uno dei due vince o c'è un pareggio).
- Funzione utilità, definisce il punteggio finale ai giocatori.

Se partiamo dallo stato iniziale e sappiamo le mosse possibili e la transazione sappiamo tutti i possibili stati in cui ci possiamo trovare, tutte queste cose formano l'albero di gioco = albero di ricerca.

Equilibrio di Nash: Prendere la decisione migliore per il caso peggiore. Durante questo equilibrio nessun agente può migliorare il proprio guadagno cambiando la propria strategia se gli altri mantengono la loro. Si cerca di minimizzare la perdita nel caso peggiore.

Ottimo paretiano: Caso in cui per migliorare la propria soluzione si finisce per peggiorare quella dell'altro. Una soluzione è detta Pareto-ottimale se non è possibile migliorarne una senza peggiorarne un'altra.

Dinamiche dominanti: Quando un giocatore fa il meglio che può indipendente da quello fa l'altro.

Una strategia ottima è una strategia che porta ad un risultato almeno pari a quello di qualsiasi altra strategia.

Valore minimax: Il valore minimax di un nodo corrisponde all'utilità di trovarsi nello stato corrispondente, assumendo che entrambi gli agenti giochino in modo ottimo da li alla fine della partita. In sostanza, il valore minimax ci dice qual è il miglior punteggio che un giocatore può aspettarsi di ottenere partendo da uno specifico stato di gioco, supponendo che nessuno dei due commetta errori strategici fino alla fine della partita.

La ricerca minimax non è altro che una ricerca in profondità.

Performance di minimax: Risulta essere un algoritmo completo se l'albero è finito, ed è ottimo se l'avversario è ottimo. Risulta essere esponenziale alla profondità in complessità temporale e polinomiale nella sua complessità spaziale.

Lezioni 11

É possibile dimezzare l'esponente per ridurre la complessità temporale di questo algoritmo, si fa questo attraverso una tecnica detta potatura, tecnica che consente di prendere in considerazione grandi porzioni di albero.

Potatura alfa-beta: Restituisce lo stesso risultato di minimax con l'unica differenza che pota i rami che non possono influenzare la decisione finale.

Alfa: Il valore della scelta migliore (massimo) per MAX che abbiamo trovato sin qui in un qualsiasi punto di scelta lungo il cammino. Durante la ricerca viene aggiornato per assicurasi che non ci siano migliori mosse disponibili.

Beta: Il valore della scelta migliore (minimo) per MIN che abbiamo trovato sin qui in un qualsiasi punto di scelta lungo il cammino. Viene aggiornato per verificare se l'opzione corrente non rende una mossa peggiore.

Buona idea potrebbe essere quindi quella di esaminare per prima i successori più promettenti utilizzando una strategia best-first.

La ricerca ad approfondimento iterativo prevede di esplorare inizialmente la profondità di un livello alla volta. A ogni livello, si aggiorna l'ordine delle mosse per esplorare prima quelle che si ritiene siano più promettenti (strategie best-first). Questo metodo può rivelarsi efficace, specialmente in giochi complessi, dove andare in profondità senza esplorare percorsi più vantaggiosi potrebbe essere inefficiente.

Un altro modo è quello di considerare le trasposizioni che portano l'albero di gioco a contenere cammini ridondanti, ovvero degli stati che si ripetono che portano ad aumentare il costo della ricerca. Delle serie di scelte con ordine diverso portano spesso ad una configurazione comune, conviene quindi memorizzare la valutazione di una configurazione ogni volta che una nuova viene incontrata.

Questa memorizzazione avviene tramite una tabella che è chiamata tabella delle trasposizioni (come dire esplorato su un nodo per esempio). Grazie a questa tabella possiamo decidere di ordinare le mosse successive sulla base della conoscenza già acquisita esplorando una trasposizione.

Per minimax con l'aggiunta di potatura non risulta gestibile arrivare ai nodi terminali dello spazio di ricerca. Per evitare di doverli raggiungere si decide di applicare una funzione di valutazione degli stati. Questa funzione restituisce una stima del guadagno atteso in una determinata posizione. Se si trova in uno stato non terminale dovrebbe calcolare la probabilità di vincita attuale.

Stato non quiescenti: Stati che portano ad una variazione repentina del valore delle mosse.

Miglioramenti per alfa-beta:

- La potatura in avanti limita la ricerca solo ai rami considerati più promettenti, basandosi su euristiche. Tecniche come la beam search aiutano a focalizzarsi sulle migliori k mosse a ogni passo, mantenendo alta l'efficienza.

- Il database di mosse di apertura e chiusura consente di memorizzare mosse ottimali per l'inizio e la fine del gioco. Questo accelera il processo e riduce la necessità di esplorare interamente l'albero di gioco nei turni iniziali.

FINE PRIMA INTERCORSO

Lezione 13

Agente capace di apprendere: Permette agli agenti di operare in ambienti inizialmente sconosciuti diventando nel tempo più competenti.

Questo tipo di agente è composto da:

- Elemento di apprendimento: Elemento responsabile del miglioramento interno.
- Elemento esecutivo: Elemento responsabile della selezione delle azioni esterne.
- Elemento critico: Elemento responsabile di dare feedback sulle prestazioni correnti dell'agente.
- Generatore di problemi: Elemento responsabile di suggerire azioni che portino a esperienze nuove e significative.

Apprendere: Un processo tramite il quale un sistema migliora le prestazione sulla base dell'esperienza. Miglioramento nel tempo dell'esecuzione di un dato task rispetto ad una misura di prestazione su una base di (esperienza).

Machine learning: Studio dei dati per fare previsioni. Componente diversa che gli permette di migliorare algoritmi. Algoritmi basati sui dati, migliorando i dati miglioreranno gli algoritmi.

Apprendimento supervisionato: C'è la mano del progettista in quello che sarà compiuto. Dati etichettati (esempio mail spam no spam) porta l'algoritmo ad essere supervisionato. Le etichette sono variabili dipendenti (label o etichette) ovvero variabili che vogliamo classificare nel futuro. Il progettista deve fornire le etichette e il database di partenza.

Apprendimento non supervisionato: Avrà i dati ma non saranno etichettati l'agente dovrà imparare senza conoscere l'oracolo di partenza. Generalmente i non supervisionati sono usati per problemi complessi quelli per cui è costoso etichettare dati e in quel caso è non supervisionato poiché il compito è solo di recuperare dati e pre-processarili.

Modello: Realizza il problema in scala e quello che deve fare è prendere un istanza e "azzeccare" l'etichetta nel supervisionato, nel non supervisionato cerca di raggrupparli in classi omogenee e deve assegnare l'istanza al raggruppamento più coerente.

Esistono anche semi-supervisionato e per rinforzo. Nel semi-supervisionato alcuni dati sono etichettati mentre altri no. In quello per rinforzo si risolve un problema dei supervisionati ovvero che abbiamo un modello statico su cui fare previsioni per il futuro ma si basa sempre sugli stessi dati di partenza, invece con quello di rinforzo aggiustiamo il modello dinamicamente con bonus se azzecca e malus se sbaglia l'etichetta.

Output: Che valore può assumere quello che vogliamo predire se sono numerici parliamo di problemi di regressione. Se invece dobbiamo classificare come spam e non spam parliamo di classificazione. Se abbiamo problemi non supervisionati e dobbiamo raggruppare abbiamo il clustering.

Misura di prestazione: Quanto sbaglia e capire chi è meglio di un altro.

Errore: Differenza tra il valore stimato della variabile da predire e il suo valore attuale.

Errore irriducibile: Errore che non può essere ridotto che dipende dal problema, un errore che avremo sempre e comunque. L'errore non dipende però solo dai dati visto che anche i modelli di machine learning possono produrre predizioni non reali.

Predizione + errore irriducibile + errore modello = Valore reale. Tutti i modelli cercano di minimizzare l'errore del modello.

Bias: Si verifica quando un modello, addestrato su un dataset, fornisce output errati e non riesce a generalizzare per risolvere problemi in diverse circostanze. Questo accade perché le assunzioni alla base del modello non sono adeguate al problema, impedendogli di cogliere le caratteristiche essenziali per formulare previsioni corrette.

Varianza: Si manifesta quando un modello produce output sistematicamente variabili, mostrando instabilità. Un'elevata varianza indica che il modello si adatta eccessivamente ai dati di addestramento (overfitting), compromettendo la sua capacità di generalizzare correttamente a nuovi dati.

Si cerca di minimizzare sia bias che varianza per ottenere un output il più possibile accurato e stabile. Tuttavia, bias e varianza sono inversamente proporzionali, quindi è necessario trovare un compromesso tra la flessibilità del modello e la sua capacità di generalizzare su dati mai visti. L'obiettivo è costruire un

modello che mantenga buone prestazioni anche su dati nuovi. Questo compromesso influenza non solo il comportamento del modello, ma anche la valutazione della sua qualità complessiva.

Under-fitting: Bias non riesce a capire bene i dati e tende a sottoadattarsi. Non riesce a capire gli schemi nei dati e non riesce poi a dare output sensati. Ha i dati ma non riesce a capirli. Un modello che underfitta non può risolvere problemi complessi. Non riesce a predire in maniera accurata.

Over-fitting: Modello a varianza elevata, il modello si adatta eccessivamente ai dati appena però vede qualcosa di diverso sbaglia. Le possibili conseguenze sono che non riesce ad apprendere dati leggermente diversi oppure si comporta in maniera troppo complessa che peggiora le prestazioni. Dà predizioni sistematicamente diverse, non capisce le caratteristiche di dati non viste ma se i dati sono simili a quelli visti potrebbe fare dei ragionamenti interni che peggiorano le prestazioni e quindi tende ad avere una complessità maggiore.

Alcune operazioni comuni in machine learning includono:

- Selezione delle caratteristiche rilevanti: Permette al modello di concentrarsi solo sui dati utili, migliorando l'apprendimento.
- Convalida incrociata: Crea diversi insiemi di test per perfezionare l'apprendimento e migliorare la generalizzazione.
- Configurazione dei parametri: Ottimizza i parametri degli algoritmi parametrici per ridurre il rischio di underfitting o overfitting.
- Aumento della dimensione dei dati: Fornisce al modello più osservazioni per migliorare l'apprendimento.

Lezione 14

CRISP-DM: Rappresenta il ciclo di vita di progetti basati sull'intelligenza artificiale e data science. Modello non sequenziale dove le diverse fasi possono essere eseguite un numero illimitato di volte.

Le fasi sono:

- Business Understanding: Raccolta dei requisiti e definizione obiettivi. Questa fase produce anche i business success criteria ovvero i criteri con cui ci accertiamo di essere in linea con gli obiettivi. Ha come output il piano del progetto, ovvero un documento che spiega l'esecuzione del progetto da una vista di gestione tecnica.

- Data Understanding: Identificazione , collezione , e analisi dei dati. Serve a capire i dati, capire se i dati che vogliamo collezionare sono sufficienti al nostro obiettivo. I quattro passi sono acquisizione , documentazione (cercare di spiegare in che modo quei dati sono stati recuperati e tutte le informazioni che possono validare gli obietti tecnici), esplorazione e valutazione della qualità. L'output un documento di analisi dei dati che riporta i metodi di astrazione e analisi e le relazioni tra i dati e i problemi di qualità.

- Data Preparation: Preparare i dati. Feature engineering selezione caratteristiche più rilevanti che supportano la soluzione del problema. Abbiamo una fase di pulizia dei dati in base ai problemi trovati nella fase precedente. Infine i dati vengono formattati. Output insieme di dati che verranno considerati in fase di modellazione dell'algoritmo.

- Data Modeling: In questa fase si seleziona la tecnica o l'algoritmo da usare. Poi si passa alla fase di addestramento dove si configurano i parametri del modello selezionato e si vedono i risultati del modello addestrato. Output il modello di machine learning ovvero l'algoritmo addestrato e configurato sui dati a disposizione.

- Evaluation: Fase di validazione dei risultati per vedere se sono chiari e in linea con gli obiettivi. Output il risultato della validazione del modello che rivela il grado di attendibilità e conformità dell'algoritmo rispetto agli obiettivi di business.

- Deployment: Ha come obiettivo quello di mettere in funzione l'approccio definito e di renderlo usabile. Output report finale di progetto che descrive tutte le fasi condotte oltre che al piano di manutenzione e monitoraggio.

SCRUM: Modello di ciclo di vita che prevede la divisione dei blocchi di lavoro in sprint, ovvero brevi periodi di tempo in cui determinati requisiti vengono definiti, analizzati, progettati e sviluppati. Qui abbiamo uno sviluppo incrementale del sistema.

SCRUM ha tre ruoli fondamentali:

- Product owner: Lo stakeholder, definisce e prioritizza i requisiti.
- SCRUM Master: Agevola il lavoro e coordina le attività di sviluppo, non ha responsabilità di gestione del personale.
- Team di sviluppo: Responsabile consegna del prodotto. Il team è composto da 3-9 persone con competenze trasversali.

Sprint planning: Periodo in cui le attività da compiere vengono definite per il periodo temporale successivo.
Daily scrum: Riunione quotidiana in cui vengono riportati i progressi.
Sprint review: Fase dove si mostrano i progressi effettuati.

Combinando SCRUM e CRISP-DM otteniamo il TDSP, qui ad intervalli regolari c'è un fase di validazione di come il sistema implementa i requisiti di business.

Lezione 15

Ingegneria dei dati: Insieme delle tecniche e degli algoritmi che consentono l'estrazione, l'analisi e la preparazione di dati che siano fruibili da altre tecniche o algoritmi di analisi dei dati.

Qualità dei dati: Descrive l'accuratezza, completezza e consistenza dei dati.

Data governance: Gestisce la disponibilità, usabilità, integrità e sicurezza dei dati. Basata su standard interni ad un organizzazione o politiche che ne regolano l'utilizzo.

Variabili indipendenti: Caratteristiche che usiamo per la predizione
Variabili dipendenti: Quello che vogliamo predire.

Data Leakage: Utilizzo di una caratteristica che non è utilizzabile in fase reale ma usiamo in fase di addestramento. In pratica un modello è capace di lavorare bene in fase di addestramento ma non in fase di rilascio.

Leaky predict: Caratteristica che aiuta a caratterizzare il problema, ma che nella pratica non sarà quasi mai disponibile.

Dato: Un qualsiasi elemento di cui si dispone per un giudizio o per risolvere un problema.

Dati strutturati: Organizzati in righe e colonne
Dati non strutturati: Testi , immagini o video, qualsiasi cosa che non sia strutturata.
Dati semi-strutturati: Struttura parzialmente definita come i file JSON, il formato è fissato ma la struttura non ha una definizione stretta.

Nella data preparation facciamo:

- Data cleaning, per esempio se ci troviamo ad avere dati mancanti. Usiamo la tecnica della data

imputation ovvero un insieme di tecniche che possono stimare il valore dei dati mancanti sulla base dei dati disponibili. Le due alternative sono scartare le righe del dataset che presentano dati mancanti oppure scartare le colonne. Contraction expansion, disambigua il significato delle frasi. Stemming, sostituzione di una parola nella sua radice.

- Feature scaling: Insieme di tecniche che consentono di normalizzare o scalare l'insieme di valori di una caratteristica, visto che spesso hanno scale di valori diverse. La tecnica più comune è la min-max normalization che normalizza i valori e li fa ricadere in un intervallo a nostro piacimento. Un altra tecnica è quella della z-score normalization.

- Feature selection: In questa fase è importante la feature engineering ovvero il processo nel quale il progettista utilizza la conoscenza del dominio per determinare le caratteristiche dai dati grezzi estraibili tramite data-mining, serve a dare un senso ai dati. Ha l'obiettivo di definire le caratteristiche che possano caratterizzare gli aspetti principali del problema in esame. Differenza tra dato e caratteristica , il dato è un'informazione grezza, raccolta senza alcuna trasformazione, la caratteristica è un dato selezionato o trasformato per rappresentare meglio il problema, spesso usato per l'addestramento dei modelli di machine learning. Feature extraction generare le feature dai dati, feature construction punta a generare feature dal progettista. Una tecnica da seguire potrebbe essere quella di eliminare le feature con bassa varianza, quindi caratteristiche che hanno valori simili.

- Data balancing: Insieme di tecniche per convertire un dataset sbilanciato in un dataset bilanciato. Le tecniche sono l'undersampling ovvero eliminare casualmente un numero di istanze (righe) dal dataset. Il problema di questa tecnica è che se abbiamo pochi dati ci leva esperienza, inoltre la casualità potrebbe farmi perdere dei dati che erano particolarmente rilevanti. L'altra tecnica è l'oversampling ovvero vengono casualmente aggiunte un numero di istanze del dataset della classe di minoranza. Questa potrebbe portare all'overfitting ovvero imporre "a memoria" le istanze della classe.

Lezione 16

Classificazione: Classificare e quindi fare predizione di un valore di una variabile categorica, tramite l'utilizzo dell'esperienza, dove l'esperienza è il mining di dati, la pulizia e la strutturazione di questi dati affinché i modelli possano apprendere. Fa parte dell'apprendimento supervisionato.

Un problema di classificazione porta alla costruzione di una modello che da in output delle predizioni. Un modello è uno strumento che è l'agglomerato dei dati di partenza che con un algoritmo (classificatore) classifica nuove istanze.

Ensamble combinazione tra diversi classificatori perché abbiamo bisogno che si adattino a diverse istanze. La combinazione ci permette di sfruttare le caratteristiche complementari dei classificatori per avere una migliore classificazione.

Naive Bayes: Considera le caratteristiche della nuova istanza da classificare e calcola la probabilità che queste facciano parte di una classe tramite l'applicazione del teorema di Bayes. L'algoritmo assume che le caratteristiche non siano correlate l'una all'altra. Non valuta la relazione tra le varie caratteristiche.

Per utilizzarlo

1) Calcolo della probabilità della classe, probabilità dell'avverarsi di una certa classe calcolata come numero di istanza diviso tutte le istanze

2) Calcolo della probabilità condizionata, applicazione del teorema di Bayes per determinare le probabilità condizionate dalle caratteristiche del problema.

3) Decisione chi ha il valore di probabilità più elevato

Gli algoritmi probabilistici applicano solo probabilità e per quanto complesso sia il calcolo il meccanismo è identico e inoltre per come è definito Naive Bayes non va a considerare l'istanza nel suo complesso ma si basa su eventi indipendenti che possono accadere, ma magari sbaglia perché una variabile non dipende dalle altre.

Albero decisionale: L'albero mira a creare un albero i cui nodi rappresentano un sottoinsieme di caratteristiche del problema e i cui archi rappresentano delle decisioni. L'obiettivo è predire il valore di una variabile target grazie a delle semplici regole di decisione. Utili per la loro facilità di lettura e possono essere usati sia per problemi di classificazione che regressione.

Passi per comporre l'albero decisionale:

1) Identifica la migliora caratteristica e mettila come radice

2) Dividi il training set in sottoinsiemi. Sottoinsieme puro (contiene tutti e soli gli elementi di una classe) e impuro (ambiamo incertezze e quindi quella caratteristica non è discriminante).

3) Fai 1 e 2

Information Gain: Determina il grado di purezza di un attributo ovvero quanto quell'attributo riesce ad esprimere il dataset e in conseguenza quanto riesce a dividerlo adeguatamente.

Entropia: Maggiore è l'entropia minore e la quantità di informazioni di un messaggio. Variabile più entropica ovvero meno dispersa e più caratterizzante. L'entropia ci indica in che misura un messaggio è ambiguo e difficile da capire.

Negli alberi decisionali l'entropia è la base per l'information gain, con questo concetto i passi da seguire sono ora

1) Calcolo dell'entropia per ogni attributo del dataset
2) Dividi il training set in sotto-insiemi con l'attributo per cui l'entropia è minimizzata o il gain massimizzato
3) Crea un nodo dell'albero decisionale contenente quell'attributo.
4) Ripeti i passi fino a a quando tutti i sottoinsiemi sono puri

Per validare un modello dividiamo il data set in:

- Training set: Insieme per l'apprendimento
- Test set: Insieme di dati per la validazione

Training e test sono insiemi disgiunti, per scegliere come dividere le risorse tra i due insiemi non c'è una regola specifica ma si usa solitamente 67% e 33%. Così dividiamo senza fare nessun tipo di ragionamento.

Un altro modo per validare è convalida incrociata: Metodo statistico nella partizione e valutazione dell'insieme dei dati di partenza. Divide più volte i dati in training set e test set , questo fa si che testando più volte e su training set diversi posso migliorare la robustezza.

Passi:

Mischiare i dati in modo casuale
Divide in K gruppi
Per ogni gruppo considerare
 considerarlo come test set e i restanti come training set
 Addestrare il modello con i dati

Valutare le prestazioni del modello ed eliminarlo

Il mischiare i dati può portare a dati che sono migliori per il classificatore oppure divide in un modo irrealistico, per risolvere facciamo la validazione più volte. Oppure cerchiamo di mischiare le cose in maniera più realistica, ovvero trovare dei sottogruppi che mantengono le stesse caratteristiche della popolazione generale.

Un altro problema è che se i dati seguono una linea temporale potremmo quindi usare dei dati futuri sui dati che non sono noti a tempo di training. Dobbiamo assicurarci di mantenere la linea temporale. Non possiamo più mischiare ma dobbiamo rispettare i parametri temporali. Simulare il modello in tempi diversi della storia.

Per vedere se il modello funziona o meno dovremmo usare delle metriche di valutazioni .

Matrice di confusione: Matrice in cui vengono indicati in quanti casi il classificatore si è comportato bene o male. Sulla base della matrice di confusione possiamo calcolare una serie di metriche.

Alcune metriche sono:

- Precision: Indica quanti errori ci saranno nella lista delle predizioni fatte dal classificatore.

- Recall: Indica quanto è bravo il classificatore a recuperare istanze di un dato tipo, quante istanze positive il classificatore può determinare.

- Accuratezza: Considera anche quando un classificare ha individuato la classe negativa. È largamente usata ma il suo problema è che usassi solo l'accuratezza senza considerare altre metriche, rischio di dire che il modello accurato anche se alla fine predice ad esempi sempre la classe negativa, in pratica azzecca molto bene i veri negativi.

Lezione 17

Regressione: Task in cui l'obiettivo è predire il valore di una variabile numerica.

I problemi di regressione sono problemi di apprendimento supervisionato. Qui non avremo più una classe ma un numero e abbiamo la costruzione di un modello che si più chiamare regressore. I regressori sono delle funzioni matematiche che cercano di descrivere i dati.

Regressione lineare singola/multipla: La differenza sta nel numero di predittori (variabili indipendenti). Nella regressione singola usiamo una retta che meglio descrive il nostro problema.

Residuo: Differenza tra il valore reale ed il valore predetto dal modello di regressione.

In termini di valutazione della regressione parliamo di metriche numeriche.

Metriche

- Mean Absolute Error (MAE): Differenza media osservata tra i valori predetti e i valori reali del test set, facile da capire ma il suo limite è che non penalizza fortemente gli errori grandi.

- Mean Squared Error (MSE): Indica l'errore quadratico medio commesso sui dati presenti nel test set, risolve il problema di MAE penalizzando maggiormente gli errori grandi ma il suo limite è che la sua unità di misura è il quadrato del target il che lo rende più complicato da capire.

- Root Mean Squared Error (RMSE): Indica la radice quadrata dell'errore quadratico medio commesso sui

dati presenti nel test set. Corregge la problematica di MSE visto che con la radice quadrata abbiamo la stessa unità di misura, rimane comunque sensibile agli errori grandi e la sua interpretazione è diretta.

- R^2: Misura la percentuale della varianza, fornisce un valore tra 0 e 1 che indica quanto il modello riesce a spiegare i dati, non dà quindi un interpretazione degli errori ma solo sulla bontà del modello.

I modelli di regressione puntano a trovare la migliore retta che si adatta ai dati. Molte delle tecniche di regressione puntano a minimizzare la perdita ovvero la somma dei residui. Il metodo più conosciuto per la minimizzazione è il metodo dei minimi quadrati.

L'utilizzo della regressione lineare e assume:

- La linearità dei dati, ovvero la relazione tra variabile indipendente X e variabile dipendente Y deve essere lineare , ovvero esprimibile con una funzione lineare.

- Gli errori residui devono essere normalmente distribuiti, avere una varianza costante e devono essere indipendenti per ogni valore di X.

Se abbiamo più di un unica variabile indipendente parliamo di regressione lineare multipla, anche in questo caso valgono gli stessi vincoli della regressione lineare singola. Inoltre è importante non avere variabili ridondanti per evitare il problema della multicollinearità.

Alberi decisionali di tipo regressivo: Variante di albero decisionale per problemi di regressione. Qui piuttosto che fare predizione di una classe di un valore predominante dobbiamo modificare l'algoritmo tenendo conto che dobbiamo predire numeri.

I passi da seguire sono:

1) Individuare regioni dello spazio dove i valori interni sono simili.
2) Calcolare un valore rappresentativo di queste regioni
3) Assegnare i nuovi dati alla regione più plausibile.

Albero decisionale: Mira a creare un albero i cui nodi rappresentano un sottoinsieme di caratteristiche del problema e i cui archi rappresentano delle decisioni.

Come funziona:

1) Posiziona la migliore caratteristica del training set come radice dell'albero. Questa è rappresentata dalla caratteristica che suddividerà il set in modo che la somiglianza dei valori target (valori che vogliamo predire) sia massimizzata . Questa variabile è chiamata split che permette di ridurre al massimo la varianza dei valori target nei sottoinsiemi che si creano.
Prima usavamo l'entropia qui utilizziamo la varianza , parliamo sempre di entropia ma parliamo di distribuzioni numeriche e per descrivere questa variabilità usiamo la varianza.

2) Calcolare il miglior punto di split questo punto massimizza la riduzione della varianza. Il punto decisionale mi fa discriminare classi diverse.

3) Dividiamo il dataset in base alla variabile di split e al punto di split calcolato.

4) Ripetere i passi fin quando non si raggiunge un criterio di fermata (profondità massima, numero di dati nella foglia...)

Il valore che associamo poi alle foglie è la media dei valori target. Quando abbiamo outlier però questi potrebbero modificare la media , quindi possiamo optare per un raggruppamento sulla mediana. Una volta assegnati i valori di output alle foglie la regressione è completata. Una volta che una nuova istanza

viene presentata al modello le sue caratteristiche la faranno arrivare ad un foglia.

Anche qui vale che se non riusciamo a dividere bene i dati di training siamo costretti a finire l'albero prematuramente e questo impatta sull'accuratezza.

Lezione 18

Problemi di raggruppamento.

Clustering (raggruppamento non supervisionato): Gruppi che abbiano un certo grado di omogeneità al loro interno e eterogeneità con gli altri. Istanze di problemi non supervisionato (no etichette) mira a creare classi e etichette, senza info sulle etichette. Dobbiamo costruire un numero sufficiente di cluster.

La qualità di un algoritmo dipende dalla misura di similarità utilizzata e dalla capacità di massimizzare questa qualità scoprendo pattern nascosti e cose che legano un dataset. Questa ci servirà a valutare quanto buono è un raggruppamento nel dataset. Altre cose che rendono buono un algoritmo di clustering, sono lo scalabilità, la robustezza e la interpretabilità dei risultati.

Il concetto di similarità è cruciale all'interno del clustering, e la misura più ovvia di similarità è la distanza tra due pattern, non sempre però questa misura è significativa per la diversità.

Una misura metrica è una quantità calcolabile di distanze tra più elementi di una popolazione. Una misura metrica rispetta le proprietà di identità, positività , simmetria e disuguaglianza triangolare. Se la disuguaglianza triangolare non è soddisfatta parliamo di semi-metrica, mentre se è la simmetria a non essere soddisfatta parliamo di pseudo-metrica. Un esempio di metrica è la distanza Euclidea, altre sono la distanza di Manhattan (la distanza è data dalla somma del valore assoluto delle differenze delle due coordinate) e di Mahalanobis (si tiene conto della dispersione delle varibili e della loro correlazione), distanze di Jaccard, Hamming, Levenshtein .

Soglia: Come faccio a dire chi è vicino e lontano , somma degli errori quadrati usata per vedere la similarità di un clustering

Tipologie di clustering:

- Esclusivi e non esclusivi: Se è esclusivo ogni pattern farà parte di una singolo cluster mentre non esclusivo farà parte di più cluster.

- Gerarchico e partizionale: Un algoritmo è gerarchico se vuole costruire delle gerarchie di cluster anche dette sequenze innestate di partizioni. Mentre un algoritmo partizionale da solo delle partizioni dei pattern.

Categorie di clustering:

- Agglomerativi e divisivi: Agglomerativo se parte da cluster atomici e punta ad unire iterativamente in cluster più grandi. Divisivo se invece parte da ampi cluster per dividerli poi in cluster più piccoli.

- Seriale e simultanei: Seriale se elabora i pattern uno alla volta simultaneo se invece elabora i pattern insieme.

- Graph-theoretic e algebrico: Graph-theoretic se elabora pattern sulla base della loro collegabilità, mentre algebrico se elabora i pattern sulla base di criteri di errore.

L'algoritmo più usato è quello del K-mean che è a partizionamento iterativo con errore quadratico.

I passi sono:

1) Selezioniamo k rappresentati degli elementi dove questi rappresentanti possono essere scelti in maniera pseudo casuale.
2) Generiamo un partizionamento assegnando ad ognuno dei punto un cluster.
3) Calcola nuovi centroidi del cluster considerando la media dei valori del cluster generati al punto 2
4) Ripete lo step 2 fino a quando i centroidi non cambiano più.

Ad ogni iterazione si cerca di migliora il cluster e genera sempre una sola partizione e usa errore quadratico perché mira a minimizzare l'errore rispetto ai centroidi. Si arresta quando generando i centroidi i cluster non cambiano.

Uno dei problemi è che dobbiamo andare a dettare il numero di cluster e per farlo , se già sappiamo quanti cluster vogliamo lo sappiamo altrimenti utilizziamo algoritmi di ricerca che in base alla bontà dei cluster possono farci ottenere il numero di cluster migliore da utilizzare per k-means. Una scelta errata può portarci a influenzare in modo negativo il risultato senza farlo mai convergere ad una soluzione. Se ne abbiamo tanti i cluster sono simili tra loro e facciamo tanti cicli.

Quando lanciamo un algoritmo di clustering dobbiamo andare ad analizzarlo e nel caso fare unione, scissione o mischiare alcuni cluster.

K-means non può essere usato per dati categorici, si usa una cosa simile detta k-medoids che al posto dei centroidi ha un melodie.

Clustering gerarchico andiamo a raggruppare a multi-livello. Differenza con k-means qui non dobbiamo decidere il numero di cluster in anticipo, ma possiamo determinare il numero di cluster finali in base alle misure di distanza e all'analisi del dendrogramma. Il suo output è un dendrogramma di dati che vengono messi insieme a un certo livello. A differenza di k-means che restituisce delle partizioni disgiunte alcuni gruppi di pattern potrebbero avere caratteristiche simili se osservati ad un certo livello. Va avanti fino a quando tutti gli elementi non formano un unico cluster.

Può essere di tipo divisivo: Inizia con tutti i punti in un unico cluster e li suddivide progressivamente in cluster più piccoli. Mentre agglomerativo ognuno degli elementi crea un cluster che poi vengono uniti progressivamente. Per creare il dendrogramma, possiamo calcolare la distanza tra i cluster in due modi:
- Distanza minima: La distanza più piccola tra due punti di cluster diversi.
- Distanza massima: La distanza più grande tra due punti di cluster diversi.

Density-based clustering: Raggruppa i pattern in base alla densità della loro distribuzione. Si basa su due parametri principali, il numero minimo di punti necessari per considerare una regione "densa" e un raggio che definisce l'intorno di ciascun punto. Questo metodo è in grado di rilevare cluster con forme arbitrarie, non necessariamente sferiche o compatte.

Per validare i risultati alcune tecniche sono:

Elbow Point: Permette di determinare il numero ottimale di cluster da generare. Si rappresentano graficamente i candidati e, al variare del numero di cluster, si osserva l'errore quadratico. Non è sempre consigliabile creare troppi cluster.

Coefficiente di Forma: Valuta la consistenza dei cluster. Analizza quanto bene i punti appartengono al loro cluster di riferimento e quanto distano dagli altri cluster. Un valore di +1 indica alta coesione, mentre -1 indica il contrario.

Mojo: Misura il numero minimo di operazioni necessarie per spostare o raggruppare i dati. Richiede la presenza di etichette e può essere utile per validare clustering non supervisionati. Oltre alle distanze aritmetiche, si possono considerare altre metriche di distanza.

Lezione prof. Catolino

Collaborative Filtring: Tecnica di machine learning usata per predire la preferenza di un utente sfruttando le preferenze di altri utenti. È solitamente non supervisionata o semi-supervisionata.

User-Based Collaborative Filtering: Se A e B hanno comprato prodotti simili e B compra un prodotto X l'algoritmo potrebbe raccomandare il prodotto X a A.

Item-Based Collaborative Filtering: Se Y e Z sono spesso comprati assieme e A compra Y, ad A viene proposto Z.

Lezione 19

Large Language Model: Un modello di IA basato su reti neurali, addestrato su grandi quantità di testo per comprendere e generare linguaggio naturale, utile per compiti come traduzioni, risposte e programmazione.

Riuso: La pratica di utilizzare un modello già addestrato su un determinato problema o dominio, come base per risolvere un altro problema, riducendo il bisogno di ricostruire un modello da zero. Questa tecnica consente di sfruttare conoscenze preesistenti apprese dal modello per risparmiare tempo, risorse computazionali e dati.

Dominio della sorgente dei dati: Insieme di dati utilizzato per costruire la conoscenza di base per il task sorgente.
Dominio dei dati target: Insieme che il modello utilizza per adattarsi al test target a partire dal task sorgente.

Transfer Learning: Tecnica di machine learning in cui un modello pre-addestrato nel dominio della sorgente di dati su un task sorgente viene riutilizzato, adattato o perfezionato per risolvere un nuovo task target nel domino dei dati target.

Sequential transfer learning: Variante del transfer learning, variante del modello in cui viene addestrato progressivamente in più fasi invece di esser completamente riaddestrato da zero.

1) Il modello viene addestrato su un dataset sorgente di grandi dimensioni e generico che aiuta ad apprendere caratteristiche di base

2) Il modello pre addestrato viene ulteriormente addestrato sul dataset target o su un dataset intermedio, che può essere più piccolo e specifico rispetto al dataset sorgente. Questo avviene facendo prima un congelamento di Layer ovvero i primi layer del modello (che catturano le caratteristiche principali) vengono congelati per preservare le conoscenze già acquisite. Poi c'è un fase di fine-tuning dove gli ultimi strati vengono addestrati con i dati target per apprendere caratteristiche più specifiche.

3) I dati del dominio target vengono utilizzati per perfezionare ulteriormente il modello, a questo punto solo una parte degli strati del modello vengono aggiornati.

Transformer: Architettura di rete neurale basata sul meccanismo di self-attention che riesce a modellare relazioni complesse tra elementi di una sequenza, indipendentemente dalla loro posizione relativa.

Self-attention: Analizza ogni elemento della sequenza in relazione a tutti gli altri.

Encoder: Modulo che prende in input una sequenza e la trasforma in una rappresentazione comprensibile per il modello.
Decoder: Modulo che genera una sequenza di output basandosi sulla rappresentazione creata dall' encoder e sugli elementi precedentemente generati.

Alcuni problemi tipici dell'interazione con gli LLM sono l'ambiguità , la sensibilità al contesto, le limitazioni del modello , la conoscenza dinamica e la complessità.

Flakiness: Variabilità delle risposte date dagli LLM.

Allucinazione: Quando un modello genera informazioni false, inesistenti o non supportate dai dati disponibili.

Prompt Engineering: Arte e tecnica di progettare prompt ottimizzati per guidare un modello di linguaggio naturale verso risposte accurate, pertinenti e utili. Few-shot learning, chain of thoughts e emotion prompting.

FINE CORSO
