Università degli Studi di
Napoli Federico II

Dipartimento di Ingegneria Elettrica e delle
Tecnologie dell'Informazione

Corso di Intelligenza Artificiale

Elaborato Esercitativo

Work Projects

Milo Saverio
Pommella Michele
Previtera Gabriele

Prof. Neri Filippo

Anno 2016-2017

Work Project 3

3.1 Esercizio 1: Logica Proposizionale

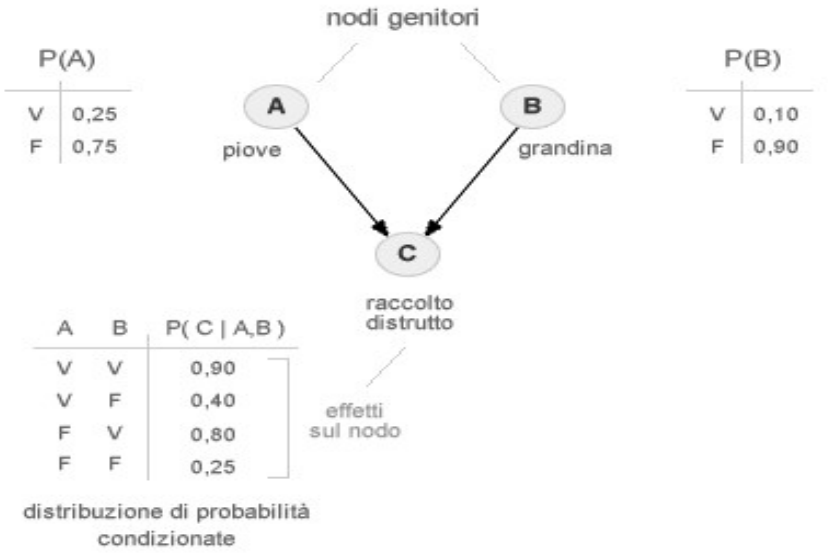
La logica proposizionale è un linguaggio formale utilizzato dagli agenti logici per la rappresentazione della conoscenza. Un agente intelligente che sfrutta tale approccio si ispira al processo umano di deduzione a partire da avvenimenti noti che ha precedentemente appreso. Quindi l'agente avrà bisogno di una base di conoscenza (`KB: knowledge base`), in cui archiviare le informazioni utilizzate nella deduzione, e di un meccanismo di inferenza (`inference engine`) per effettuare le deduzioni.
L'agente, basato sulla conoscenza, ogni volta che innesca il processo deduttivo, enuncia alla base di conoscenza la percezione corrente, dopodiché chiede ad essa l'azione da eseguire, la scelta avverrà in base alla informazioni archiviate,comunica alla fine di aver eseguito l'operazione,così da garantire che l'evento scatenante il processo di ragionamento possa ampliare la propria base di conoscenza. Inizialmente l'agente può non essere a conoscenza di tutto ciò di cui necessita per completare la sua operazione, quindi sarà necessario una fase di raccolta delle informazioni iniziali. Possiamo fornire questa conoscenza al nostro agente in due modi, mediante un approccio dichiarativo, vi è proprio una fase di raccolta di informazioni da parte dell'agente, oppure nel caso di un approccio procedurale possiamo fornire noi una conoscenza iniziale all'agente. Solitamente i due approcci vengono usati insieme in modo tale da avere una buona base di conoscenza.
Le informazioni non possono essere salvate nella base di conoscenza dell'agente in un linguaggio naturale, o comunque in un qualsiasi modo da cui non sia possibile trarre delle conclusioni. Per tale motivo le frasi, o formule, che costituiscono il KB, sono espresse in un linguaggio formale di rappresentazione della conoscenza. Esso prevede una sintassi, per esprimerle in maniera corretta, ed una semantica, che gli attribuisce un significato. La stessa identica frase espressa naturalmente può essere tradotto in modi differenti in base alla diversa sintassi della logica che andiamo ad utilizzare. Infatti il risultato da ottenere, determinare se delle evidenze possono

implicare dei fatti (`entailment`) è indipendente dalla logica utilizzata per la traduzione. Utilizzeremo la logica proposizionale, in cui la sintassi è composta da frasi atomiche, legate tra loro tramite connettivi logici per ottenere frasi più complesse, il singolo elemento della frase può assumere valore vero o falso,infatti la semantica di tale logica è quella di trovare i mondi in cui la frase risulta vera. Le operazioni tra i letterali sono: la **negazione** che muta il valore della nostro letterale; l' operazione di **and** che ci da vero solo se i due connettivi sono veri;l'operazione di **or** restituisce vero se uno dei due è vero;l' **implicazione** ci restituisce falso se dal vero deduciamo il falso e infine quella **bicondizionale** vera se e solo se entrambi i letterali hanno lo stesso significato. Per verificare la relazione di implicazione possiamo ricorrere a due strade, la prima è quella in cui enunciamo tutti i possibili mondi del nostro modello, per esempio tramite la tabella di verità, tra questi individuiamo quelli che rendono veri la nostra base di conoscenza e la deduzione che stiamo facendo e se i mondi che rispecchiano la nostra percezione sono contenuti all' interno del concetto che vogliamo dedurre, allora possiamo dire che la nostra conoscenza implica quel fatto. Sulla tabella possiamo vederlo facilmente basta verificare che per ogni riga in cui la nostra evidenza è vera anche la frase che vogliamo implicare lo sia.
Tale metodo è molto oneroso quando i possibili mondi sono tanti, enumerarli tutti sarebbe molto dispendioso sia in termini di spazio che di tempo, per tale ragione di preferiscono altre strade.
Una possibile è scrivere le frasi in forma CNF e da li applicare metodi inferenziali per capire se la nostra conoscenza implica il fatto. Prima di tutto però dobbiamo trasformare le frasi che abbiamo nella forma CNF, cioè or di congiunzioni atomiche, per farlo sfruttiamo le regole di equivalenza logica. Purtroppo anche con questo metodo c'è un grande sforzo dell' essere umano che deve applicare tali regole affinché sia possibile dedurre qualcosa. Quando la conoscenza, è nella forma da noi voluta, possiamo provare a dimostrare che questa, renda non soddisfacibile il negato della frase che vogliamo dedurre, cioè non esista nessun modello che implica quella frase. Se ciò accade significa che la frase non negata è valida, cioè in tutti i mondi in cui è vera la nostra base di conoscenza anche la frase lo è. Per verificare che not A sia insoddisfacibile, basta prendere una coppia di clausole(congiunzione di letterali), presenti nella nostra base di conoscenza ed iniziare a derivare nuove informazioni, ciò avviene quando nella frase che si viene a creare si presentano due letterali complementari (A e not A). Applichiamo questo ragionamento ricorsivamente fino a quando non riusciamo più a determinare nuove clausole e quindi quel fatto non è deducibile dalla nostra conoscenza, oppure si arriva alla clausola nulla, potendo affermare che la not A non è soddisfacibile. Tali situazioni implicano la validità della clausola di partenza

A.
Tutti e due algoritmi sono completi e soundness. Un algoritmo è completo se é capace di enumerare tutte le relazioni di implicazione possibili dalla base di conoscenza. Un algoritmo è soundness quando ogni implicazione dello stesso, rispecchia quelle della base di conoscenza. Per entrambe le procedure enunciate vale il discorso che l' agente non conosce il significato della frase, quello è dato dall' essere umano, per l' agente una frase vale un' altra, esso è solo capace di applicare le regole inferenziali, non riuscendone infatti a discernere il significato.

3.2 Esercizio 2: Reti Bayesiane

L' approccio probabilistico, rispetto alle altre tipologie tiene conto di un grado di incertezza. Infatti a differenza di un approccio logico, in cui un fatto può essere vero o falso, nell'approccio probabilistico si considera la probabilità di quell' avvenimento. Tale livello d'incertezza non ci dice a priori se sia vero oppure no, per esempio affermando che oggi pioverà con una probabilità di 0.99 potrebbe capitare che non piova, proprio perché la probabilità definisce il livello di incertezza rispetto ad un conoscenza da noi acquisita che varia da persona a persona (agente ad agente), dipendete da esperienze diverse che determinano valori di incertezza diversi. Le reti Bayesiane si rifanno al concetto di probabilità, più propriamente alla regola di Bayes così definita: $P(causa|effetto) = \frac{P(effetto|causa)*P(causa)}{p(effetto)}$. Una rete Bayesiana di solito è fatta in questo modo:



come osserviamo ci sono dei nodi, che determinano la variabile aleatoria da tenere in considerazione, a cui si assegnano anche i relativi valori di probabilità a priori e dei collegamenti che indicano la relazione di causalità tra i nodi, determinata dalla freccia che va da un nodo verso un altro.
Per realizzare tale rete dobbiamo tenere conto delle relazioni di causalità delle nostre variabili, così facendo otteniamo una rete che riesce ad esplicare bene le nostre conoscenze. Una peculiarità di questa rete è la compattezza, infatti qualsiasi altro modo di realizzare la rete ne aumenterebbe la complessità.
Una rete così realizzata permette di effettuare calcolo delle probabilità in modo semplice, perché in una rete siffatta vale l' indipendenza condizionata, una volta definito un valore per una variabile padre, le variabili figlie sono indipendenti tra di loro. Per esempio se la variabile A è padre di B e C la probabilità $P(BandC) = P(B|A) * P(C|A)$, se non avessimo l' indipendenza non potremmo scrivere i due domini separati, ciò implicherebbe che dovremmo tener conto anche della probabilità di B dato C. Per queste reti si descrive anche il concetto di semantica locale, cioè ogni nodo è indipendente dai suoi non discendenti dato il nodo padre; e della coperta di Markov, un nodo è indipendente da tutti dato il nodo padre, i figli e i padri dei figli.
Per ricavare dati da queste reti date le probabilità a priori, possiamo semplicemente calcolare tutti i valori di probabilità a posteriori che ci occorrono, metodo complesso nel caso in cui la rete sia grande e il numero di possibili valori delle variabili è elevato. Di solito si può ricorrere all' approccio frequentistico, che consiste nel generare un token, questo avrà una probabilità di ottenere un valore del nodo pari proprio all' incertezza legata

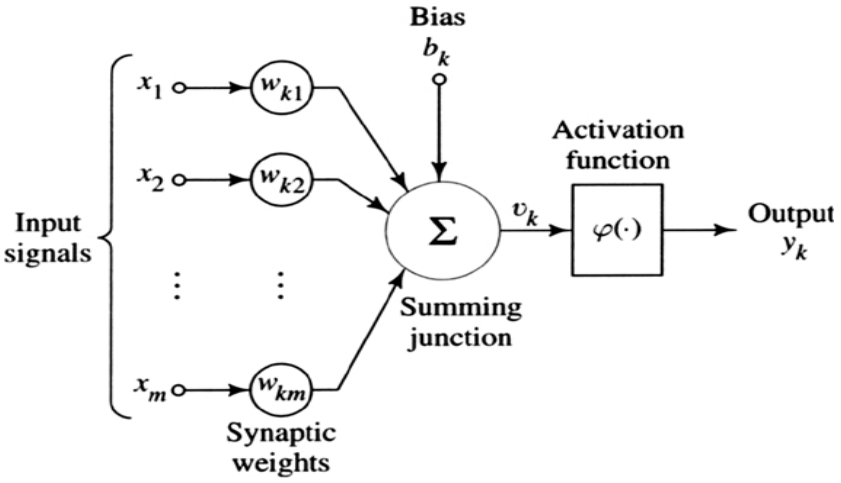
ad esso, dopodiché si inizia a campionare la rete facendo scorrere il token su di essa, fissandone i valori con quelli risultanti dal campionamento. Le configurazioni ricavate, una volta raggiunti la quantità di risultati voluti, vengono divise per il numero dei campioni valutati, tenendo conto della loro molteplicità. Tale metodo per un alto numero di eventi determinati deve tendere all' approccio di calcolare le probabilità a posteriori. Le reti bayesiane così costruite sono statiche perché non tengono conto di un riferimento temporale, infatti esistono le reti bayesiane dinamiche che tengono conto anche di un valore temporale, suddiviso in istanti. Quindi la variabile casuale non ha solo relazioni con altre variabili, ma anche con se stessa però valutata in istanti diversi di tempo. I valori di probabilità di solito che vengono ricavati su tali reti sono: il `filtering` cioè la probabilità che avvenga un determinato evento all' istante t dati gli eventi precedenti e l' evidenza; lo `smoothing` la probabilità che un evento in un istante t-k con k che va da 0 a t data l' evidenza fino all' istante t, utilizzato per correggere la probabilità con cui si va da un fatto in un istante ad un altro; la `previsione`, la probabilità di un evento all' istante t+k con k>1 dati gli eventi e l' evidenza fino a t; la `spiegazione migliore` che data la probabilità all' istante t di eventi, evidenza quali sono quei determinati valori che ci permettono di avere quella probabilità.
Per esempio il `filtering` può essere facilmente calcolato ricorsivamente, difatti conoscendo il fatto iniziale ed essendo i processi stazionari, possiamo successivamente determinare quello all' istante successivo e poi pesare la probabilità dello stato data l' evidenza. Questo è possibile per la stazionarietà dei processi, cioè per passare da un fatto ad un altro la tabella della probabilità non cambia rendendo così tale relazione identica ad ogni avanzamento. Esiste anche una tecnica detta di `particle filtering` in cui faccio un campionamento sul fatto iniziale, propago tali esempi sul campione successivo li peso per l' evidenza e rifaccio il campionamento per lo stato successivo tenendo conto del nuovo peso dei campioni. Ovviamente per ottenere il valore di probabilità di una determinata configurazione, dobbiamo dividere il numero di casi uguali per il totale di campionamenti effettuati e per un alto valore di quest' ultimi la mia probabilità tende a quella reale che avrei ottenuto con il filtering. Tali reti possono essere usate anche nel machine learning. Dato un insieme di ipotesi possiamo vedere qual' è la probabilità associate ad esse data un' evidenza, cioè vogliamo calcolare la probabilità di $P(hi|d)$ dove hi è l' ipotesi e d è l' evidenza, ovviamente ad ogni nuovo valore potrebbe variare la hi che massimizza la probabilità. Non è molto agevole portare avanti i calcoli per tutte le ipotesi, per tale ragione si sceglie quella che massimizza il valore a posteriori (`MAP`), cioè quella ipotesi che massimizza $\alpha * P(d|hi) * p(hi)$ dove α è il valore di normalizzazione. Nel

caso in cui le ipotesi hanno probabilità a priori (stessa complessità) uguali si può usare la `maximum-likelihood (ML)`, in cui basta solo massimizzare $p(d|hi)$. Potremmo pensare all' uso dei logaritmi nel caso in cui compaiano esponenziali rendendo la massimizzazione della derivata molto più semplice.

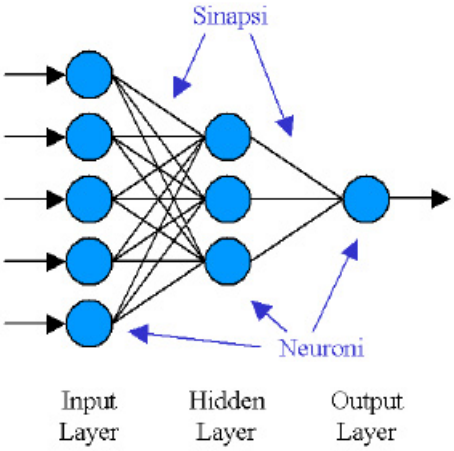
3.3 Esercizio 3: Reti Neurali

Le reti neurali sono un meccanismo computazionale che si ispira al funzionamento del cervello umano. Sono rappresentabili tramite un grafo orientato, i cui nodi costituiscono un modello matematico del neurone, e agli archi orientati è associato un peso. Ogni nodo i calcola la sua uscita a_i applicando una funzione di attivazione g, chiamata `activation function`, alla somma pesata degli ingressi.

$$a_i = g(in_i) = g(\sum_{j=0}^{n} W_{j,i} a_j) \tag{3.1}$$



La funzione di attivazione storicamente utilizzata era quella di *soglia*; in seguito si è diffusa la funzione *sigmoide*, continua e derivabile rispetto alla precedente.
Possiamo classificare strutturalmente le reti neurali distinguendo le `Recurrent Networks` e le `Feed-Forward Networks`. Le ultime non si fondano sul concetto di stato interno, a differenza delle prime,queste possono essere suddivise in più livelli, o *layers*. Le reti neurali a singolo layer riescono a rappresentare solo operazioni *linearmente separabili*, a due layer tutte le funzioni continue e quelle a tre discontinue.



Una difficoltà, non trascurabile, risiede nella costruzione della struttura della rete neurale. Esse infatti sono sistemi `black-box`, il cui funzionamento interno è di difficile interpretazione logica, anche se è possibile averne una matematica che non aiuta nella sua costruzione.
Per determinare il peso degli archi si utilizza l'algoritmo di `Back-Propagation`.Inizialmente si costituisce una rete con pesi casuali, o uniformi per abolire l' onere computazionale. Consideriamo poi una coppia ingressi-uscita del data-set. L'errore tra l'uscita ottenuta e quella desiderata verrà propagato all'indietro,per ogni livello, così da modificare i pesi ed ottenere una migliore approssimazione della funzione.Tale propagazione fa uso del `learning rate` per far si che un singolo esempio non influisca interamente sui valori dei pesi, altrimenti questi tenderebbero ad oscillare di molto nel caso in cui ogni esempio fornisca un errore relativamente grande. La procedura verrà iterata per ogni esempio, ciascuno dei quali consentirà una correzione dei pesi, delineando una struttura della rete coerente con la funzione da approssimare. I problemi principali che riscontriamo in questo algoritmo sono l'eccessiva complessità temporale e la possibilità di arresto su di un minimo locale.
Per quanto riguarda, invece,la determinazione del numero di neuroni, si procede tipicamente con un approccio `trial and error`, perché come già detto non è chiaro qual' è il significato della rete, per tale motivo si parte da reti semplici, se queste dopo la `Back-Propagation` funzionano bene, le si usa, altrimenti si iniziano a inserire nuovi neuroni o layer di questi e si continua a testare la validità delle nuove reti create.
In definitiva, le reti neurali si sono rivelate empiricamente un buon approssimatore di funzioni, ed oggi il loro utilizzo ha ripreso vigore nella

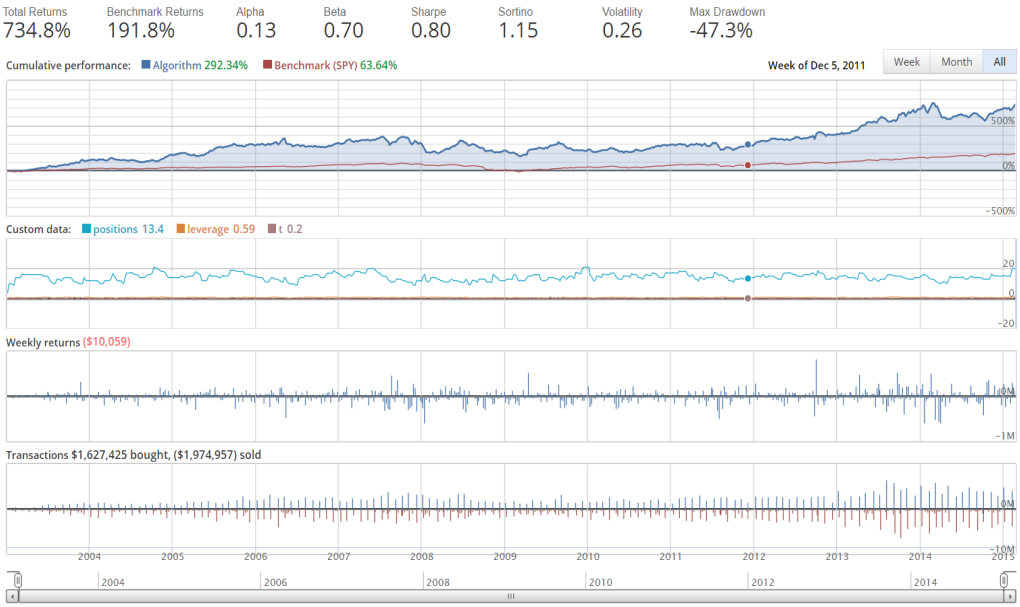
forma di `Deep Learning`.Tali tecniche ottengono un basso *error rate* nei problemi relativi al riconoscimento di caratteri e sono largamente diffuse nei sistemi multimediali per quanto concerne il riconoscimento di immagini, audio o nella `Computer Vision` in generale. Seppur presentando buoni risultati, bisogna ricordare che *non esiste un algoritmo di apprendimento migliore in assoluto, ma bisogna trovare quello più adatto al problema dato.*

3.4 Esercizio 4: Cloud e Crowdsourcing

Il `Cloud Computing` ed il `Crowdsourcing` sono paradigmi affermatisi nel nuovo millennio, figli della dirompente rivoluzione della rete non solo al livello culturale, ma anche economico, fornendo nuovi modelli per sviluppare ed offrire servizi al cliente.
Il cloud prevede l'erogazione di risorse informatiche *on demand* attraverso Internet. Tipicamente, in linea del tutto generale, un `Cloud Provider` fornisce un *pool* condiviso di risorse; esse possono essere configurate da un cliente amministratore che le carica di valore aggiunto; infine l'utilizzatore finale usufruisce delle risorse in tal modo configurate per poi rilasciarle al termine del suo utilizzo.
Il crowdsourcing è lo sviluppo collettivo di un progetto da parte di volontari esterni al suo ideatore. Questo fenomeno deriva dalla nascita delle prime *open community* di condivisione, legate profondamente al concetto di *sharing*. Le *community open source* sono state la prima a sfruttare i benefici del crowdsourcing. Questo paradigma si è diffuso anche in ambito aziendale con il modello di business di *open enterprise*.
Quantopian applica questi concetti fornendo i servizi per scrivere algoritmi di trading e rendendoli disponibili all'intera community, nella quale si potranno condividere idee, codice e dati. Esso, inoltre, investe sugli algoritmi con le migliori performance, condividendone i profitti con l'autore. Rappresenta, dunque, un modello di business all'avanguardia, che sfrutta pienamente le potenzialità della community, guidandola tramite i propri strumenti e ricavando introiti dall'impulso creativo a cui il crowdsourcing porta.
Tale piattaforma offre molti dataset direttamente senza importarli da altre fonti, evitando la ricerca compulsiva sul web. Nel nostro esempio vediamo operazioni di trading su azioni americane, di una nostra possibile società che ha un certo numero di azioni iniziali e decide quando vendere o comprare tali titoli per aumentare i suoi introiti partendo da un capitale iniziale ed osservando cosa accade in un determinato lasso di tempo.

Il codice fornisce tale output:

la linea rossa definisce il reale valore che si avrebbe avuto confrontandosi con il mercato, invece quello blu determina la previsione stimata dell' algoritmo , il valore position ci dice il numero di azioni attive, cioè non ancora chiuse, leaverage ci dice quanto siamo disposti ad investire in base agli introiti,il guadagno settimanale è la transazione monetaria che c'è stata. Come si può vedere dall' esempio la nostra scelte di trading secondo il nostro algoritmo sono molto rosee rispetto a ciò che è successo realmente, anche se a grandi linee comunque rispetta l'andamento avuto, dandoci però un guadagno falsato. La modifica dell' algoritmo, oltre ad essere permessa da qualsiasi pc essendo tutto in cloud, viene eseguita da una server farm che sicuramente è più performante del nostro pc, difatti quello che si vede in figura solo le valutazioni del mercato americano in circa 12 anni, un onere di computazione non banale. Una possibile modifica dell' algoritmo è puntare solo sulle azioni che ci danno un valore di utile, maggiore:



Come vediamo l' algoritmo si è comportato meglio, perché ho preso solo le 3000 azioni che mi davano un' utilità sperata maggiore, quindi ho fatto trading su azioni che mi permettono di massimizzare i guadagni, ovviamente è un operazione che di solito premia a lungo termine, però non è detto che quell' azione abbia il comportamento sperato, può capitare che la sua efficienza sia minore, dato che fattori non ancora accaduti potrebbe in realtà portare ad un' utilità diversa. Un altro esperimento è stato quello di aumentare il profitto che si voleva fare con una determinata azione, tendendo quindi a cercare di sfruttare quando più quell' azione per massimizzare i guadagni,il risultato è peggiorato:

Possiamo vedere che il numero di transazioni sono state minori rispetto ai due casi precedenti, appunto perché una volta che si è iniziato a fare un trade su quell' azione fino a che non ci porta al guadagno voluto o scende al di sotto di un valore minimo non la vendiamo, così abbiamo cercato di attuare una politica più conservativa sulla compravendita, che ci ha portato a dei ricavi minori, ma ad un atteggiamento più consono a quello che in realtà è successo.