Riassunto di supporto alle decisioni

# Il supporto alle decisioni nella business intelligence

Dal momento che il mondo attuale produce una grande quantità di dati che continuerà a crescere nei prossimi anni (a causa dell’ IoT), bisogna trovare un modo per gestirli opportunamente e da essi prendere le migliori decisioni possibili.

I cosiddetti Big Data hanno infatti le seguenti proprietà:

* hanno un volume enorme;
* la velocità di produzione è elevata;
* c’è molta varietà di dati;
* la veridicità dei dati è molto importante perché da essi bisogna prendere decisioni, talvolta però l’affidabilità e la qualità sono limitate.

Tutti questi dati possono essere caricati nel datawarehouse dopo averli puliti e organizzati grazie agli ETL, in questo modo è possibile interrogarli per effettuare delle analisi, anche attraverso dei tool appositi.

Si possono comunque memorizzare i dati nel loro formato originale per evitare perdite di informazione, questi possono essere analizzati in diversi modi e ciò permette di evitare i tempi di attesa degli ETL. Non è necessaria una struttura ben fatta per poter analizzare i dati, basta prendere decisioni velocemente dal momento che può esserci concorrenza.

Di conseguenza, il datawarehouse da solo non basta, devono esserci anche tecniche per prendere decisioni anche coi dati grezzi, tenendo conto dei possibili errori.

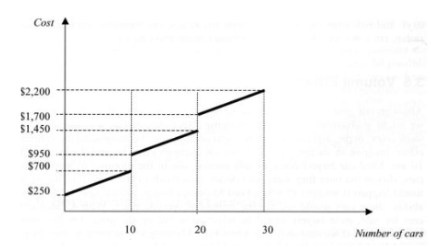
Solitamente la soluzione migliore è dinamica nel tempo e/o dipende da condizioni esterne, a fronte di ciò cosa si fa? Si può utilizzare il process mining la cui analisi di processi permette di cambiare il modello utilizzato con uno più adatto allo scopo.

Ciò significa che è necessario un trade-off tra modelli più facili da trattare ma meno precisi rispetto a modelli più precisi ma anche più complessi.

Le soluzioni devono soddisfare degli obiettivi, alcuni di questi potrebbero essere in contrasto, portando di conseguenza a conflitti. La scelta di un modello o dell’altro dipende dal dominio applicativo, la loro costruzione avviene a partire dall’input e dipende in buona parte dalla sua qualità.

## Esempio

Bisogna trovare un modo per distribuire delle auto negli Stati Uniti, queste sono state comprate nei punti neri e bisogna rivenderle in quelli grigi.

Il primo problema è infatti dove distribuire le auto e ci sono cinquanta possibili soluzioni, ognuna dipendente dall’altra dato che una soluzione con certe caratteristiche implica che bisogna spostare le auto in altri stati per evitare che il prezzo scenda.

Lo spazio delle soluzione è quindi enorme, quello che si può fare è diminuire lo spazio o evitare la soluzione migliore a favore di altre.

Il secondo problema riguarda i trasporti, questi infatti sono costi e bisogna considerarli.

Se ad esempio un camion può portare un massimo di 10 auto e bisogna trasportarne 11, bisogna utilizzarne per forza 2 e quindi il costo raddoppia.

Per rendere il modello più realistico si introduce la dinamicità variando il prezzo di trasporto, eccetera in base al periodo, in casi come questi è opportuno che la soluzione deve essere resistente ai cambiamenti, indipendentemente dal fatto che sia la migliore o meno.

Un problema può inoltre comprendere differenti obiettivi, ognuno con la propria importanza, certi vincoli infatti permettono certe soluzioni anzichè altre.

In alcuni casi è opportuno creare un modello per effettuare delle previsioni.

## Quali sono i tipi di conoscenza?

La conoscenza può essere:

* implicita: è la conoscenza che deriva direttamente dai dati usandoli così come sono;
* esplicita: è la conoscenza generalizzata, ottenuta facendo astrazioni sui dati.

## Conoscenza esplicita

La conoscenza esplicita deriva da linee guida ed è formalizzata in regole e/o modelli, indicando il metodo standard più corretto per trovare una soluzione in un dato contesto.

Per utilizzarla al meglio occorre strutturarla con regole, modelli matematici, eccetera.

Nella conoscenza esplicita, le regole definiscono le conseguenze date le premesse, i modelli invece descrivono le realtà con delle proprietà matematiche.

## Conoscenza implicita

La conoscenza implicita è la conoscenza più operativa dato che esamina il dato utilizzandoli direttamente nel processo di ragionamento. In questo caso non è necessario modellare niente dato che è già a disposizione e potrebbe essere già strutturato e digitalizzato. L’acquisizione di questi dati è più facile rispetto alla conoscenza esplicita proprio per questi motivi, inoltre sono utili per ricavare la soluzione e quindi anche le previsioni.

L’unica pecca di questi dato è che non sono utilizzabili al di fuori del loro contesto.

## Quale delle due conoscenze è meglio utilizzare?

L’ideale sarebbe utilizzarle entrambe, sia considerando un utilizzo combinato, sia utilizzando quella più adatta al momento.

All’inizio veniva utilizzata solamente la conoscenza esplicita, infatti sono stati sviluppati sistemi a regole complessi, questi però furono un insuccesso dato che non c’era interattività ed erano poco applicabili. Bisogna inoltre considerare che una volta i tool non erano veloci come oggi, lo stesso discorso vale anche per l’acquisizione dati, facendo da collo di bottiglia.

Di recente la conoscenza implicita è stata sfruttata grazie all’aumento di dati in circolazione, ciò anche dovuto a nuovi algoritmi di machine learning e alla maggior potenza di calcolo.

La tendenza odierna è utilizzarle entrambe per ottenere più vantaggi possibili, quindi generalizzare soluzioni applicate su casi concreti, ottenere nuova conoscenza implicita e da essa ricavare quella esplicita.

Inoltre è possibile astrarre la conoscenza implicita per ricavare conoscenza esplicita e viceversa.

Attualmente si sta cercando di integrare tutto in vari modi:

* utilizzando sistemi separati scegliendo il più adatto;
* utilizzando la conoscenza esplicita per strutturare le repository dei casi concreti, velocizzando il recupero;
* utilizzando la conoscenza implicita per contestualizzare un problema;
* utilizzando la conoscenza esplicita per rendere più chiari gli algoritmi di intelligenza artificiale, questi infatti sono come delle black box che forniscono una soluzione senza dare spiegazioni.

## Ragionamento per regole

Il metodo classico per ragionare con la conoscenza esplicita è il sistema a regole, l’idea è rappresentare tutta la conoscenza con delle regole semplici, la complessità si ottiene attraverso delle concatenazioni con connettivi, logica temporale, eccetera.

Se una premessa è vera, allora lo è anche la conseguenza, ciò può rendere di conseguenza vere altre regole e così via.

Partendo da una situazione iniziale con dati iniziali, sulla base di questi il motore inferenziale:

* cerca di capire le regole attivabili;
* Alcune di queste regole possono andare tra loro in conflitto, per risolverli si utilizzano delle priorità;
* L’attivazione delle regole comporta nuova conoscenza, da essa si va a vedere se esistono altre regole attivabili e così via.
* Se non ci sono regole attivabili o non ci sono nuovi dati, il sistema si ferma.

Allo stesso modo si può fare il processo inverso: partendo dalla conclusione si cerca di ricostruire la situazione iniziale.

I sistemi a regole sono facili da comprendere e interpretare e la loro modellazione è semplice, tuttavia formalizzare la conoscenza è un processo lento e difficile, tenendo conto anche di possibili eccezione che possono portare a problemi.

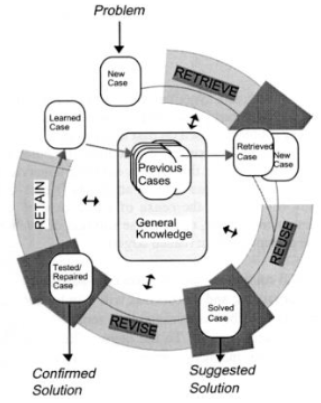
Inoltre, con un grande numero di regole, questi sistemi sono inefficienti.

## Come si acquisisce la conoscenza?

Il problema dell’acquisizione della conoscenza può essere risolto con diversi approcci:

* si estraggono le regole dalla conoscenza degli esperti del dominio, ciò permette di ottenere un sistema facile e leggibile ma costoso nella formazione;
* si imparano le regole in automatico attraverso il data mining, ciò permette di scoprire i legami tra i dati senza l’aiuto di un dominio. Questo approccio è automatizzabile e più veloce del precedente, tuttavia le regole ottenute sono poco leggibili e difficili da interpretare;
* Un approccio misto permette di unire i primi due punti per sfruttarne entrambi i vantaggi, quindi si definisce parte delle regole con l’esperto del dominio, questa viene poi aggiornata e affinata tramite il data mining.

## Ragionamento per casi

Il ragionamento basato su casi permette di gestire la conoscenza implicita, fornisce informazioni e conoscenza una volta interpretata ed è strategica dal momento che è stabile nel tempo.

Questo tipo di ragionamento utilizza i dati a disposizione per ragionare nei processi decisionali, precisamente li usa come esempi per trovare soluzioni a problemi simili, adattandoli quando necessario.

I sistemi a casi sono utilizzabili in contesti in cui la conoscenza a dispozione è incompleta o imprecisa, tuttavia ciò non è sufficiente nei casi in cui bisogna dedurre regole generali o modelli.

Questo sistema è inoltre in grado di aumentare la sua conoscenza in modo automatico, in suo ragionamento infatti si basa sulle 4 R:

* Retrieve: recupero delle situazioni più simili al problema in input;
* Reuse: prova a utilizzare la soluzione così com’è;
* Revise: cerca di adattare la soluzione al problema;
* Retain: il vero passo di apprendimento, verifica infatti se è bene tenere la soluzione modificata.

### Retrieve

La fase di retrieve confronta il problema in input con ogni caso utilizzando i concetti di similarità e distanza, l’obiettivo infatti è quello di recuperare i casi più simili. La nozione di distanza non è univoca dato che viene effettuata per ogni attributo, quindi in questi casi diventa utile la conoscenza esplicita in modo da pesare gli attributi in base a delle priorità.

Con dei modelli piatti, la fase di retrieve è una ricerca esaustiva, tuttavia la conoscenza esplicita può renderla più veloce cercando di indirizzarla il più possibile.

Il passo di retrieval viene implementato in due modi:

* può specificare una soglia massima in cui trovare i casi in un insieme rispetto alla query in input, questo metodo è detto range query;
* può specificare solo i k casi più simili, la cosiddetta k-NN query. Questo approccio è utile per indicare la classe di un gruppo di casi, si può quindi sfruttare nel passo di retain per la classificazione di un nuovo caso.

### Reuse

La fase di reuse prova a riutilizzare le soluzioni dei casi recuperati nel problema in input, almeno per quanto riguarda la tecnica. Anche qui la conoscenza esplicita può essere utile per pesare gli attributi e rendere la soluzione coerente.

### Revise

Questa fase controlla se la nuova soluzione è effettivamente applicabile nella pratica, correggendola e revisionandola dove necessario. In questa fase viene integrata la conoscenza esplicita con quella implicita, tutto ciò per l’adattamento delle soluzioni proposte (utilizzando un sistema a regole).

### Retain

La fase di retain si pone se è necessario o meno memorizzare la nuova soluzione dato che si tratta di nuova conoscenza implicita. Come fa a capirlo? Essenzialmente, la soluzione nuova viene valutata con un concetto di distanza rispetto ai casi già memorizzati. Inoltre viene anche valutato lo sforzo computazionale per la produzione della soluzione

All’inizio è giusto memorizzarne un po’, tuttavia con troppo casi memorizzati si rischia di rallentare tutto, bisogna quindi evitare di inserire soluzioni ridondanti.

La repository di casi può essere arricchita con una base di conoscenza contenente della conoscenza esplicita, essa serve per velocizzare il recupero dei casi più simili.

Il problema dei CBR riguarda la possibile non strutturabilità dei casi, di conseguenza la fase di retrieve potrebbe essere molto complessa così come le altre fasi. Proprio per questo motivo in molte situazioni è implementata solo la fase di retrieve.

## Vantaggi

I sistemi a casi permettono di ridurre il collo di bottiglia causato dall’acquisizione della conoscenza, qui infatti la conoscenza può essere incrementata in automatico senza il bisogno di un esperto del dominio.

Inoltre la costruzione di una soluzione può essere semplificata dal momento che si possono riadattare le soluzioni memorizzate.

Quando si utilizza questa tecnica? Ebbene, il riadattamento di una soluzione viene utilizzato a fronte di problemi simili tra loro, quando ci sono delle eccezioni non catturabili con un sistema a regole e quando il recupero di una soluzione costa meno della sua costruzione.

## Com’è strutturato il un sistema a casi?

Una caso indica la descrizione di un problema aventi delle caratteristiche indicate con una lista di coppie <feature, value>, a ogni caso sono associate la soluzione e l’outcome, cioè la feature indicante il risultato che ha permesso di assegnare la soluzione.

Nei casi più complessi, le feature sono a loro volta strutture dati più complesse, come ad esempio serie temporali o sequenze di tempi.

La base di casi può essere strutturata in vari modi:

* nel caso piatto, ogni caso è una riga della tabella di un database e ogni attributo ha lo stesso peso degli altri. La ricerca di un caso sfocia quindi in una ricerca esaustiva, inoltre è più lenta ma più facile da mantenere.
* nel caso gerarchico, si individuano gli attributi più importanti per definire dei cammini privilegiati durante la selezione dei casi. Ciò permette di velocizzare la ricerca ma rende più complesso il mantenimento dei casi.

## Modelli a memoria dinamica

I modelli a memoria dinamica sono modelli gerarchici che permettono di individuare i casi più promettenti utilizzando una struttura ad albero.

A partire dai casi nella base dei casi, vengono definiti degli episodi generalizzati, dei prototipo che riassumono le caratteristiche dei casi più specifici, essi sono connessi tra loro e ai singoli casi tramite una rete di discriminazione.

Questa rete di discriminazione specifica le cosiddette norme, cioè gli insiemi di tutti gli attributi comuni e con gli stessi valori dei casi che discendono da un’episodio generalizzato.

La rete di discriminazione filtra i vari casi utilizzando gli indici assegnati a ognuno di essi.

Considerando la foto a destra, il primo episodio generalizzato contiene tutte le informazioni comuni a tutti i casi dato che si trova nella radice, il secondo invece è formato da tutti gli attributi comuni ai casi 1, 2 e 4, precisamente contiene index2 e index3.

Ciò significa che nei casi 1, 2 e 4 del secondo episodio, i valori di index2 e index3 saranno gli stessi per tutti, inoltre questi tre casi si distinguono dagli altri perchè hanno degli attributi in più.

La fase di retrieve andrà a fare una ricerca in base agli attributi comuni e discriminanti, un discorso simile vale anche per il retail, qui infatti bisogna trovare la posizione corretta dei nuovi casi.

Questo è un metodo per imparare conoscenza generalizzata inizialmente non nota, da ciò si può ottenere e mantenere conoscenza esplicita.

## Category & exemplar model

Il category & exemplar model è un modello gerarchico che costruisce una rete semantica utilizzando gli attributi dei casi.

Sulla base degli attributi, vengono definite delle categorie e in essa si ordinano gli esempi (i casi) a partire dal più tipico.

In questo modello la rete semantica costruisce una rappresentazione della conoscenza del dominio.

Anche in questo caso, le fasi di retrieve e retain avvengono per mezzo degli attributi, in ciò permette di orientare la ricerca verso le categorie di interesse e quindi dei casi più simili agli input. Nella fase di retain bisogna anche determinare la categoria e il livello di prototipalità del nuovo caso in input, sulla base di quest’ultimo è inoltre possibile definire nuovi attributi.

## Adattamento

L’adattamento indica il riutilizzo generale di una soluzione, questa può subire poche modifiche oppure si può sfruttare il processo di costruzione per crearne un’altra.

Un’altra cosa che si può fare è combinare più soluzioni insieme.

Tutti questi metodi devono però rispecchiare le differenze tra il caso in input e tutti quelli recuperati, inoltre c’è molta dipendenza dal dominio.

Il processo di adattamento è complicato dato che può richiedere a sua volta del ragionamento.

## Learning e mantenimento

La fase di learning è quella che permette di apprendere e mantenere una buona soluzione, è importante però non memorizzarne tanti per non rendere difficile la fase di retrieval.

Durante il mantenumento dei casi, è opportuno effettuare:

* una valutazione del coverage, ciò dell’insieme di problemi risolvibili da un singolo caso;
* una valutazione della reachability, l’insieme dei casi che possono risolvere un problema.

Sulla base di queste analisi, si valuta l’eliminazione di inconsistenze e ridondanze.

Fare troppo learning è dannoso dal momento che rende il retrieval delle soluzioni più oneroso senza aggiungere effettivamente nuova conoscenza

## Curiosità sul retrieval

Esistono molti tipi di retrieval, in alcuni casi è addirittura possibile effettuare più fasi di questo tipo per raffinare il recupero in base ai nuovi attributi aggiunti.

Nel caso dei recommended system, si possono proporre casi simili a quelli scelti, un’applicazione di questi sistemi è nell’e-commerce prendendo come base gli acquisti.

Come si può avere un retrieval flessibile?

Ebbene, si possono utilizzare tecniche fuzzy, consistenti in un’insieme di coppie ordinate (x, m(x)):

* x indica il caso;
* m(x) indica il grado di appartenenza di x a un dato insieme, il valore restituito è un numero compreso tra 0 e 1.

Le coppie fuzzy permettono di capire quali casi sono più simili agli altri, quali sono “uguali” e quali “diversi”, permettendo una sorta di classificazione, utile quando l’utente non ha le idee ben chiare sul da farsi.

Da ciò è possibile ottenere query fuzzy da effettuare su questi insiemi, esse sono convertibili comunque in normali query.

La logica fuzzy è applicabile anche sugli attributi oltre che nei confronti, permettendo flessibilità nella scelta dei casi.

# Cos’è la distanza di edit?

La distanza di edit indica il numero minimo di operazioni da fare per convertire una stringa x in un’altra, come si fa ciò? Ebbene, tutto viene effettuato in base a queste tre azioni:

* insert(a): inserisce il carattere a nella posizione corrente della stringa;
* delete(a): cancella il carattere a dalla posizione corrente della stringa;
* replace(a,b), sostituisce il carattere a con b nella posizione corrente della stringa.

La tabella a destra indica tutte le operazioni da fare per convertire la stringa RISOTTO in PRESTO, in tutto sono state 4 operazioni di editing, la distanza quindi è 4.

Esistono tante soluzione per convertire una stringa in un’altra, un esempio in questo caso è tagliare la prima T e mantenere la seconda.

Qual è la distanza di edit migliore tra due stringhe? Quella minima!

## Approccio per la distanza di edit minima

Per calcolare la distanza di edit minima, si utilizza la programmazione dinamica, precisamente la tecnica della memoizzazione per evitare di ricalcolare le soluzioni più semplici.

Date due stringhe X e Y lunghe m e n rispettivamente, per memorizzare le varie soluzioni, si utilizza una matrice m\*n, in cui, nella cella (i,j), è presente la distanza delle due stringhe Xi e Yj (rispettivamente le stringhe X e Y fino al carattere i e j).

L’algoritmo prevede una fase di inizializzazione della tabella inserendo i sottoproblemi più semplici, precisamente:

* d(X0,Yj) = j, ciò è dovuto al fatto che X0 è una stringa vuota e quindi ci vogliono j inserimenti per ottenere Yj. Queste distanze vengono memorizzate nella prima riga della tabella;
* per lo stesso motivo, d(Xi,Y0) = i, tutte memorizzate nella prima colonna della matrice.

Considerando ora il passo intermedio, ci possono essere due casi differenti:

* se i caratteri delle due stringhe sono uguali, allora si propaga la distanza i-j, ciò equivale al numero di passi per trasformare una stringa in sé stessa;

D[i,j]=D[i-1,j-1]

* se invece i caratteri sono diversi viene memorizzata una distanza diversa in base all’azione eseguita. In generale la distanza è il costo minimo per convertire il prefisso, a ciò si aggiunge 1, il costo dell’operazione.

In caso di caratteri differenti, ci possono essere tre possibili casi:

* se si effettua un inserimento, l’algoritmo ricava la distanza dalla colonna precedente;;

D[i,j] = 1 + D[i,j-1];

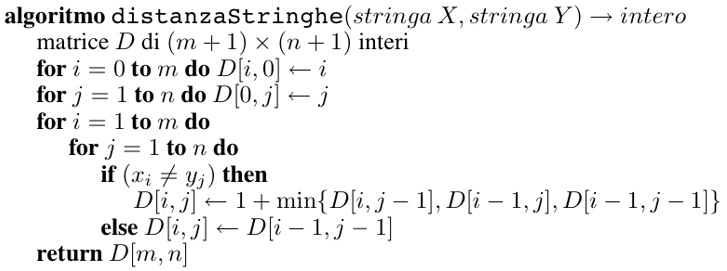
* Nella cancellazione, la distanza viene ricavata dalla precedente riga;D[i,j] = 1 + D[i-1,j];
* Con la sostituzione invece, si prende la cella precedente sulla diagonale.

D[i,j] = 1 + D[i-1,j+1]

Si può quindi concludere che la distanza viene propagata se i caratteri sono uguali, altrimenti si calcola il minimo dall’intorno e si aggiunge 1.



## Algoritmo



## Ricostruzione della soluzione ottimale

La ricostruzione della soluzione ottimale avviene partendo dalla cella in basso a destra della matrice, si controllano le celle adiacenti e si va dove c’è distanza minima.

In base a come ci si muove, si ricava l’operazione corretta e da qui la sequenza di azioni di costo minimo.

# Retrieval di tracce di processo

Una traccia di processo indica una sequenza di azioni eseguite per completare un processo produttivo.Le tracce di processo sono accompagnate da diverse informazioni:

* le informazioni quantitative indicano ad esempio l’inizio e la fine seguendo la metrica del tempo;
* quelle qualitative invece indicano la posizione di un’azione rispetto alle altre, qui entra in gioco l’algebra di Allen. Oltre a ciò, sono presenti anche informazioni aggiuntive come ad esempio il nome dell’azione e le risorse utilizzate.

Queste azioni vengono eseguite in sequenza, tra queste possono anche esserci sovrapposizioni.

Ogni traccia è la storia delle azioni che vengono effettuate e quindi rappresenta un caso.

## Come funziona il retrieval in questo caso?

Si possono recuperare tutte le tracce simili per indicare come procedere nel caso attuale, ciò è detto operational support e cerca di prevedere la prossima azione, il suo tempo rimanente e/o i costi stimati.

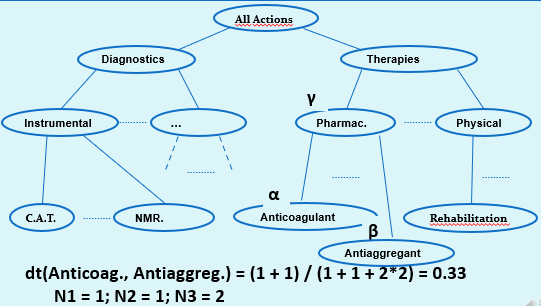
Tutte queste azioni sono rappresentabili con un grafo, ogni azione infatti può essere applicata in diversi ambiti, quindi una traccia di processo è un ramo di questo grafo.

Tutto ciò è il primo passo di un’analisi detta process mining, cioè l’applicazione di date tecniche per ricostruire e imparare il funzionamento di un processo, in esse le tracce rappresentano un ramo dell’esecuzione.

Data la complessità di questi grafi e di conseguenza delle tracce, esistono tecniche di clustering che permettono di ottenere semplificazioni e processi più mirati, oltre che ripulire i processi già imparati..

## Distanza tra tracce

Come si calcola la distanza tra tracce? Partendo dall’edit distance, l’obiettivo è migliorarla tenendo conto delle informazioni temporali.

Le informazioni non temporali della traccia vengono tra loro confrontate, ciò permette di valutare eventuali sostituzioni e di pesarle diversamente in base a ciò. Com’è possibile tutto questo? Grazie alla conoscenza esplicita!

La nuova edit distance verrà calcolata in base alla distanza tra le azioni all’interno di una certa gerarchia, essa però non è definita secondo una metrica delle azioni bensì in base alle scelte svolte con un esperto del dominio.

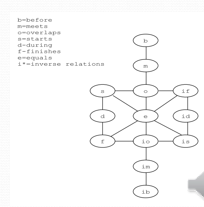
La distanza di edit deve anche fornire informazioni riguardanti l’allineamento ottimo tra le azioni in sequenza, ciò serve per ottenere tracce allineate nel modo migliore.

Un’esempio di distanza di edit è la distanza di Palmer:

Nella formula sopra, N1 e N2 indicano le distanze di alpha e beta dall’antenato comune gamma, N3 invece è la distanza di gamma dalla radice.

## Gestione dei vincoli temporali

La gestione dei vincoli temporali avviene dopo il calcolo dell’edit distance degli attributi non temporali.

Per gestire i vincoli qualitativi, quello che si può fare è cercare di confrontare le informazioni temporali delle tracce, precisamente si comparano le durate delle azioni che occorrono in tracce differenti, calcolandone l’interval distance.

L’interval distance può essere utilizzata per confrontare le durate degli intervalli tra azioni su tracce differenti.

E per quanto riguarda i vincoli qualitativi?

In questo caso l’interval distance viene meno dal momento che non permette di confrontare le durate delle azioni, al suo posto di utilizza la neighbors-graph distance, la quale permette ciò seguendo i cammini di un grafo predefinito:



Nella formula, la distanza è il rapporto tra la distanza tra due nodi e il cammino più lungo percorribile nel grafo, ciò permette di legare le informazioni temporali con quelle non temporali, anche in maniera pesata.

Questi vincoli non richiedono per forza che un’azione debba avvenire prima di un’altra, infatti possono esserci sovrapposizioni di questo tipo:

* meets: un’azione inizia quando finisce quella precedente;
* overlap: le azioni sono sovrapposte almeno in parte.

## Retrieval di modelli di processo

Utilizzando gli algoritmi di process mining, si ha un modello di processo generale, utile per trovare casi simili.

Dando delle tracce di processo in input a un algoritmo di process mining, si ottiene un grafo indicante il modello di processo appreso. In questo grafo i nodi sono le azioni (aventi risorse e tempi) da eseguire mentre gli archi indicano i flussi di controllo, questi forniscono informazioni qualitative e riguardanti le affidabilità.

Il processo imparato può essere confrontato con uno schema già esistente, nel caso delle linee guida, potrebbero esserci differenze causate da adattamenti locali, posticipi di azioni per via di mancanza di risorse, dell’obsolescenza oppure della specializzazione verso casi specifici.

Se questo schema non esiste, è possibile apprenderlo.

## Come si confrontano i processi?

Il confronto tra processi avviene definendo una metrica, ciò è difficile a causa della non linearità dei processi e delle relative informazioni. Tutto questo è difficile perchè i processi vengono rappresentati attraverso grafi, il loro confronto non è banale, tra i tanti cammini bisogna infatti trovare quelli minimi.

La soluzione è quella di avere un mapping per ogni nodo e per ogni arco, qui la programmazione dinamica viene meno a causa dell’elevato costo, di conseguenza si utilizzano approcci greedy.

Gli approcci greedy, nonostante siano subottimali, permettono di ottenere mapping accettabili in tempo polinomiale.

Anche in questo caso si ha un’estensione della edit distance, ciò permette di calcolare la distanza delle singole componenti tra due modelli di processo, inoltre permette di tenere conto se due o più azioni sono sovrapposte o meno.

Il calcolo della edit distance viene definito con quattro operazioni:

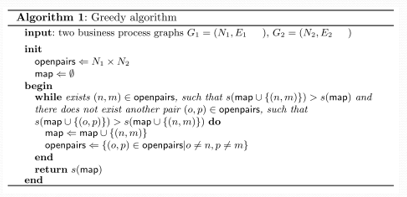
* il rapporto tra i nodi inseriti e quelli cancellati;
* il rapporto tra gli archi inseriti e cancellati;
* la distanza media dei nodi sostituiti, per calcolarla si applica un’edit distance su tutte le coppie di nodi sostituiti;

Il rapporto degli archi sostituiti

Tutte queste formule indicano quant'è la distanza indotta da un mapping, per calcolarla si utilizza la seguente formula:



## Algoritmo



## Astrazione nel confronto tra tracce e processi

Astrarre su tracce e processi è utile per ignorare dettagli irrilevanti per il retrieval, rendendo i grafi più leggibili.

Tutto ciò si basa sul fatto che tracce e processi hanno obiettivi uguali o simili, quindi tralasciare certi dettagli permette di ottenere tracce e processi più simili e leggibili, facendo notare le differenze più importanti e rendendo il retrieval più efficiente.

Tutto questo è possibile applicando la conoscenza implicita ed esplicita per operare su tracce e modelli.

## Confronto tra tracce e modelli di processo

Esistono casi in cui non è necessario ottenere un confronto dettagliato tra tracce e processi nella fase di retrieval, quindi si può evitare di ottenere un’insieme più ampio di elementi simili.

Un’insieme troppo piccolo non permette una valutazione corretta della soluzione, di conseguenza può essere utile un’astrazione per rendere tutto più leggibile, i dettagli infatti rendono i grafi troppo complessi e di difficile interpretazione.

### Modellazione tramite ontologie

L’ontologia è una struttura gerarchia ad albero in cui si indicano tutti i possibili scopi, questi più sono vicini alla radice, più sono generici, definendo anche possibili relazioni tra scopi.

Precisamente gli scopi si trovano sulle foglie, a questi è possibile associare delle azioni che mirano a soddisfarlo.

### Modellazione tramite sistemi a regole

Dal momento che alcune azioni possono essere utilizzate per diversi scopi in base a dove stanno sulla traccia, quindi è necessario disambiguare.

I sistemi a regole serve per esplicitare lo scopo di date azioni, lo scopo dell’azione infatti varia in base al contesto in cui viene eseguita.

### Aggregazione di azioni

Nel caso in cui delle azioni adiacenti condividono uno scopo, è possibile aggregarle insieme, a quali condizioni?

* la somma dei ritardi tra queste azioni non deve superare la soglia della conoscenza del dominio;
* non devono esserci altre azioni frapposte con un obiettivo diverso, è possibile ammetterne solamente nel caso in cui siano poche e di breve durata.

Tutte le azioni vengono quindi inglobate in una macroazione, qui bisogna inoltre ricordare l’eventuale presenza di azioni frapposte in modo da poterla utilizzare nel confronto di tracce astratte, penalizzando macroazioni con delay lunghi.

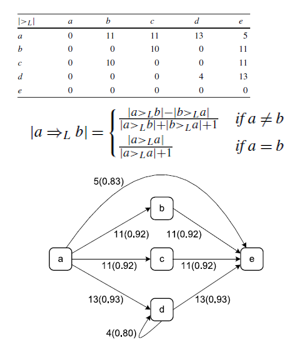
Tutti questi meccanismi permettono di ottenere modelli di processo più semplici e leggibili, in essi sono ben visibili gli scopi delle azioni che permettono di visualizzare i processi svolti.

## ProM

ProM è un tool che mette insieme algoritmi per il process mining, l’obiettivo è quello di utilizzarli per imparare e astrarre modelli di processo per verificare l’andamento e il funzionamento di un particolare caso.

La fase di discovery è quella che, date in input delle tracce di process, apprende il modello,quest’ultimo è di solito rappresentato con una rete di Petri oppure un simil-flowchart.

La fase di conformance parte da un modello di processo e le relative tracce per verificare se queste ultimi sono conformi misurandone l’allineamento, infatti se non è così, potrebbe esserci stata qualche eccezione.

Ultima ma non meno importante, la fase di enhancement mira a completare o migliorare un modello di processo già esistente utilizzando delle tracce date in input.

## Fase di discovery

La fase di discovery cerca di imparare il modello di processo focalizzandosi sulle azioni e il loro ordine, qui sono inoltre presenti degli split che permettono di seguire un percorso rispetto ad un altro.

Una volta imparato il modello, si può utilizzare per verificare l’andamento del processo.

Esistono tanti algoritmi per apprendere il modello di un processo, uno di questo è Heuristic Miner il quale, basandosi sulle frequenze di azioni A che seguono azioni B e viceversa, permette di ricavare il modello anche in presenza di rumori.

Precisamente l’algoritmo va a vedere se è meglio avere A→ B o B→ A, in alcuni casi però l’ordine delle azioni è indifferente.

Utilizzando alcune di queste dipendenze, si costruisce un grafo, da ciò si può capire se sono presenti cicli o meno.

# Retrieval di serie temporali

Le serie temporali sono sequenze di valori che evolvono nel tempo, qui è presente un problema di dimensionalità dei dati e diventa quindi difficile sia memorizzarli, sia rappresentarli.

Tutto ciò porta a un retrieve lento causato dai confronti tra serie temporali, inoltre la quantità dei dati da considerare è molto alta, rallentando il calcolo della distanza punto a punto.

Come si procede in questo contesto? Ebbene, le trasformate matematiche vengono incontro riducendo la dimensionalità.

La trasformata di Fourier permette di passare da tempo a frequenza lasciando invariato il numero dei dati e restituendo i coefficienti della combinazione lineare delle sinusoidi

I coefficienti ottenuti dalla trasformata sono molto significativi nella prima parte rispetto a tutto il resto, quindi si può considerare solo questa questa per rappresentare la serie temporale riassumendone il comportamento.

Un’altra tipo di trasformata è Wavelets, qui i concetti vengono descritti da forme d’onda più complesse anziché delle sinusoidi, il ragionamento però rimane lo stesso di Fourier. Questa trasformata è possibile rappresentare i picchi di segnale in modo più accurato rispetto a Fourier.

Entrambe le trasformate preservano la distanza tra serie temporali, l’utilizzo di pochi coefficienti però potrebbero sottostimarla, ciò genera dei falsi allarmi e falsi dismessi, queste tuttavia non sono un problema.

## Indicizzazione

Grazie alle trasformate è possibile implementare strutture di indicizzazione spaziale multidimensionale, le quali permettono di velocizzare il retrieval sfruttando i coefficienti delle serie temporali. Queste tecniche si basano su:

* R-tree, strutture ad albero basate sulla distribuzione spaziale degli oggetti da indicizzare;
* TV-tree, una struttura ad albero che utilizza un numero variabile di coefficienti per l’indicizzazione, alcuni di essi vengono utilizzati solo quando è necessario per discriminare di più (solitamente ai livelli inferiori dell’albero).

## Piecewise constant approximation

La piecewise constant approximation consiste nel dividere la serie temporale in k segmenti, dove n è il numero di punti della serie temporale. Per ogni segmento ottenuto, si prende il valore medio. Valori piccoli di k permettono un’approssimazione più grezza e trasforma la funzione originale in una lineare a tratti.

Questo approccio permette di ricostruire la serie originale rispetto a ciò che viene fatto con Wavelet a onde quadre, permettendo l’adozione di funzioni di distanza più complesse al costo di una mancata indicizzazione.

## Tecniche qualitative

Le tecniche qualitative permettono di individuare il legame tra i dati grezzi e quelli trasformati in modo semplice, come si fa? si effettua un mapping dagli uni verso gli altri. Precisamente queste tecniche si basano su astrazioni temporali, le quali permettono di trasformare dati puntuali in intervalli.

Come si individuano questi intervalli? Ebbene, è necessario individuare una serie di parametri e delle definizioni basate sulla conoscenza del dominio.

Le astrazioni temporali semplici possono essere:

* di stato, cioè quando mirano a rivelare intervalli i cui punto hanno mantenuto lo stesso valore qualitativo. Qui i parametri da impostare riguardano i limiti superiori e inferiori per la definizione dei livelli qualitativi, le durate minime e massime e il numero di outliers, cioè tutti i punti che non rispettano la definizione qualitativa dell’intervallo.
* di trend, queste definiscono l’andamento della serie temporale su un certo numero di punti contigui. Qui i parametri sono i range di pendenza minimo e massimo, utili per definire l’andamento di un intervallo, e il numero di outliers.

## Tassonomie di simboli

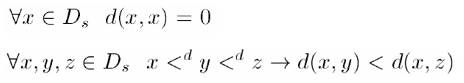
Le tassonomie di simboli vengono utilizzate per descrivere l’andamento di un’astrazione temporale nel dettaglio, etichettando gli intervalli in vari livelli.

Queste tassonomie permettono di formare astrazioni multilivello, la cosa importante è quella di rispettare gli assiomi di:

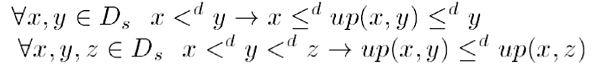
* ordine;



* distanza.



E’ inoltre possibile introdurre una tassonomia di granularità temporale per descrivere gli aspetti qualitativi con differenti livelli di precisione, in tal caso bisogna essere in grado di scalare da granularità più fini a quelle più grosse tramite lo scaling:



Deve inoltre essere possibile combinare le due tassonimie citate attraverso delle strutture a indice.

## Query Answering

Utilizzando le strutture a indice, è possibile velocizzare il query answering per indicare il livello di dettaglio richiesto:

* prima si converte la query in una sequenza di simboli;

dopo si va ad individuare il nodo corretto nell’indice, specificando il livello di dettaglio richiesto e risalendo le tassonomie.

# Modelli di predizione

La predizione nella business intelligence ha come obiettivo quello di analizzare i dati per stimare un valore, ciò equivale alla probabilità che un evento accada.

Come si fa una predizione? Esistono diverse tecniche per farlo, le più comuni sono la classificazione, la regressione o l’analisi di serie temporali.

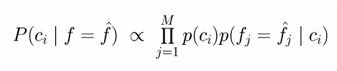
## Classificazione

La classificazione è un metodo in cui si intende etichettare un esempio con una classe in modo da comprenderne l’evoluzione, ciò è possibile solo quando il target è nominale.

In caso di target numerico e continuo, la regressione è la tecnica più adatta, essa consiste nel capire l’evoluzione in base all’andamento di una funzione.

### Naive Bayes

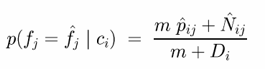
Naive Bayes è un metodo di classificazione che permette di classificare un nuovo caso stimando la classe più probabile, come fa? Partendo da un insieme di attributi f, verifica se il caso appartiene a una classe C.



L’algoritmo sfrutta il teorema di Bayes per fare la classificazione, il valore di ogni feature fj diventa indipendente una volta osservato, di conseguenza non influenza tutte le altre feature (indipendenza condizionale), ciò riduce tutto nella formula sopra.

Nel caso reale potrebbero esserci influenze tra attributi, tuttavia ciò non interessa.

Come fa l’algoritmo a stimare lla seconda com? Esistono diversi metodi per fare ciò, uno di questi è m-estimate:



M-estimation permette di calcolare la probabilità basandosi solo sugli m dati osservati che ne riassumono l’attendibilità.

### Reti bayesiane

Anche se l’indipendenza condizionata semplifica i calcoli, questo approccio risulta poco realistico e improbabile in un contesto reale. Per tenere conto delle dipendenze tra attributi si utilizzano quindi le reti bayesiane per la classificazione.

Una rete bayesiana è una grafo diretto e aciclico in cui:

* i nodi sono variabili casuali, contenenti tante probabilità quante sono le possibili combinazioni con gli attributi da cui dipende;
* Gli archi indicano le dipendenze tra nodi.

La quantità di cose da apprendere nelle reti bayesiane aumenta enormemente, di conseguenza è possibile incorrere in alte complessità in caso di reti troppo connesse. A fronte di ciò, esistono algoritmo approssimati che permettono di ottenere valori approssimati in tempi ragionevoli.

Nelle reti bayesiane è possibile andare a specificare osservazioni su alcune variabili, inoltre è possibile effettuare il ragionamento backward.

### K-NN

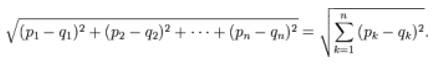
K-NN è una tecnica che si basa sul calcolo delle distanze, precisamente presente i k casi più vicini e considera questi come appartenenti alla classe più significativa (quella più frequente).

Tanto più i casi sono simili e tanto più sarà affidabile la predizione sarà alta.

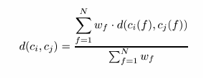
La scelta dei k casi è importante, un numero troppo alto infatti potrebbe non essere un bene. Altri fattori importanti sono la metrica utilizzata per la distanza e i pesi, la gestione dei dati mancanti e la scelta delle feature da considerare.

### Metriche di distanza

* Nel caso in cui le feature sono numeriche, si utilizza la distanza euclidea dal momento che presentano un concetto intrinseco di ordinamento e distanza;



* Per le feature simboliche invece,dato che non hanno i concetti citati prima, si utilizza una matrice di similarità in cui si indicano dei valori per ogni possibile combinazione. Tutti questi valori vengono poi combinati con la seguente formula:



### Retrieval multistep

Nel caso in cui la libreria di casi risulta divisa in classi, è possibile sfruttarla per velocizzare la fase di retrieval, ciò permette di concentrarsi solo su una parte della libreria oppure di implementare tecniche non esaustive.

In mancanza di classi, si può effettuare del clustering per costruirle utilizzando sempre il concetto di distanza.

# Clustering

Il clustering è una tecnica utile quando non si hanno a disposizione delle etichette, cosa si fa? Ebbene, si possono raggruppare i dati in base alla similarità dei loro attributi, tuttavia bisogna ancora imparare cosa li accomuna.

Il clustering può essere di due tipi:

* partitivo, cioè quando si controlla quanto è distante un caso rispetto al punto più rappresentativo;
* gerarchico, qui si costruisce una gerarchia di cluster, si parte da una situazione in cui ogni elemento è un cluster e si fanno aggregazioni tra loro.

In generale le tecniche di clustering possono essere:

* bottom up quando ogni dato è un cluster e si uniscono tutti quelli più vicini, fermandosi se la distanza minima è maggiore della soglia stabilita o se il numero di cluster è minore di quest’ultima;
* top down: si parte da un unico cluster e si divide, cercando di separare gli elementi meno omogenei, in questo caso ci si ferma quando è stato raggiunto il numero di cluster desiderato.

## K-mean

Il k-mean funziona decidendo k punti come centroidi, da essi si calcolano le distanze verso gli altri punti, il risultato minore indica che l’elemento appartiene al cluster del centroide più vicino.

La distanza utilizzata è quella euclidea, ciò però dipende dalle caratteristiche del problema.

Una volta uniti tutti i punti, si ricalcolano i centroidi e si ricontrolla ogni elemento per vedere se il cluster scelto è quello giusto.

Si continua così finchè non ci sono più elementi che si spostano da un cluster all’altro.

## UPGMA

UPGMA è un metodo gerarchico bottom up che utilizza la media aritmetica come distanza per la creazione dei cluster, precisamente:

* si calcolano le distanze tra i punti e si inseriscono in una matrice;
* si trovano gli elementi tra loro più vicini (i più piccoli della matrice) e si clusterizzano le rispettive righe-colonne;
* si calcola la distanza tra il nuovo cluster rispetto a tutti gli altri, precisamente si calcolano le medie delle distanze dei vecchi cluster;
* si prosegue in questo modo finchè non si ottiene un cluster unico.

La gerarchia di cluster ottenuta permette di comprendere i cluster tra loro distanti e quindi il punto in cui tagliare l’insieme.

# Approcci non esaustivi

Gli approcci non esaustivi vengono utilizzati solitamente in database grandi, un tecnica esaustiva in questi casi porterebbe a un costo eccessivo sia sul retrieval, sia sul calcolo delle distanze.

Una cosa che si può fare è sfruttare l’organizzazione dei casi in classi, in modo da velocizzare il calcolo delle distanze.

## Pivoting Based Retrieval

Il Pivoting based retrieval consiste nel prendere un rappresentante da una classe e calcolare la distanza verso tutti gli altri casi, quello che si ottiene è quindi un punto medio con cui gestire l’inserimento dei casi nuovi.

La stima di tutte le distanze è possibile in alcuni casi con la disuguaglianza triangolare: dato un input ci, il punto medio cm e un punto generico cg, se si sta la distanza tra cm e ci e tra cm e cg, si può si stimare la distanza ci-cg.

Nel caso in cui la stima effettuata supera il minimo dei punti medi, allora si scarta senza calcolare la distanza effettiva.

Il limite utilizzato viene infine salvato come caso promettente e si effettua lo stesso procedimento con un’altro pivot.

# Metodi logici

## Alberi

La classificazione in base alle feature è possibile anche utilizzando regole e alberi di decisione, inoltre questo metodo è explainable.

Gli alberi di decisione sono strutture dati in cui ogni nodo è una condizione, quelle più discriminanti si trovano vicino alla radice.

Le foglie invece rappresentano l’output predetto.

L’albero di decisione viene creato nel seguente modo:

* si utilizza un set di training per la creazione effettiva;
* si verifica la correttezza con un set di test.

L’idea degli alberi di decisione è quella di capire quale attributi è il più discriminante, come si fa? Si utilizza l’entropia, cioè quanto è casuale la distribuzione.

Con un’entropia bassa, i casi sono distribuiti in modo omogeneo, il contrario invece si ottiene con un alta entropia.

### Procedimento

* si calcola l’entropia per la radice;
* si divide un attributo in due insiemi, ciò serve a verificare eventuali cambiamenti significativi sull’entropia;
* si calcola l'entropia di ogni insieme e si continua così;
* si sceglie la strada che permette di ridurre l’entropia;
* si può fare pruning per tagliare i rami inutili.

L’information gain indica la differenza di entropie tra due figli, quindi se un genitore è più discriminanti verso un figlio rispetto all’altro, di tutti questi si prende quello maggiore.

Dopo aver fatto ciò, si procede allo stesso modo coi nodi restanti.

La costruzione dell’albero termina quando si ottiene un’information gain inconsistente per due alberi di decisione.

## Regole

Un albero di decisione può essere convertito in un insieme di regole, ogni ramo infatti è una regola in cui tutte i valori delle feature sono tra loro in AND, ciò permette di indicare anche la classe.

Ciò che è stato detto in precedenza forma la premessa della regola, la conseguenza invece si ottiene individuando il valore della classe presente nella foglia.

Anche in questo caso, le regole possono essere imparate in base alle associazioni, queste tuttavia risultano meno spiegabili rispetto a quelle ricavate da un esperto del domino, di conseguenze è possibile eliminarne alcune

La valutazione delle regole avviene attraverso tre indicatori:

* il support, cioè il rapporto tra il numero di istanze coperte da due atomi rispetto a quello totale;
* la confidence, il rapporto tra il numero di istanze coperte da due atomi rispetto a quelle ricoperte dal primo atomo;
* il lift, ovvero il rapporto tra il numero di istanze coperte da due atomi rispetto al prodotto del numero di istanze ricoperte dagli stessi atomi ma da soli.

Queste tre componenti possono essere utilizzate per filtrare o ordinare le regole, la prassi è quella di avere regole con confidence vicino a 1, lift superiore a 1 e support più alto possibile.

Per aggiungere più flessibilità, si convertono le regole in fuzzy, una logica in cui la distinzione tra vero e falso non è più netta bensì graduale.

## Logica fuzzy

La logica fuzzy indica asserzioni logiche che descrivono il grado di appartenenza di un elemento all'insieme.

Un insieme fuzzy è un insieme di coppie ordinate nella forma (x,m(x)), in cui x è l’elemento e m(x) il suo grado di appartenenza.

Il grado di appartenenza non è una probabilità, quindi le regole non devono per forza essere a somma 1.

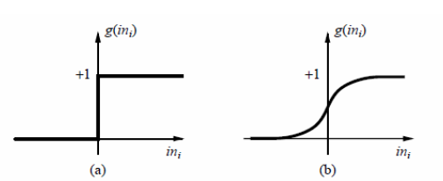
## Architettura di un sistema di predizione fuzzy

La figura a destra indica l’architettura di un sistema fuzzy, precisamente è formato da:

* un fuzzificatore, il quale prende in input dei dati crisp e, attraverso delle funzioni di membership, li trasforma in insiemi fuzzy, associando ogni valore al suo grado di appartenenza;
* Il sistema inferenziale è un sistema a regole classico esteso con condizioni fuzzy ricavate dagli insiemi calcolati in precedenza.. Le regole del sistema hanno una premessa formata da una o più condizione fuzzy relative al grado di appartenenza dell’ input, la conseguenza invece è una previsione espressa col grado di output. La combinazione di più condizioni fuzzy è possibile grazie a degli operatori, precisamente l’AND equivale al prodotto o al minimo mentre l’OR alla somma e al massimo. L’output del sistema inferenziale è un’insieme ricavato componendo le aree delle conseguenze delle relative regole.
* Il defuzzificatore è il complementare del fuzzificatore, dagli insiemi fuzzy in uscita dal sistema inferenziale, produce dei dati crisp, solitamente calcolando un valore medio.

## Reti neurali

L’idea delle reti è quelle di simulare la comunicazione dei neuroni nel cervello, con le dovute differenze:

* il processing è parallelo rispetto a quello sequenziale tradizionale;
* l’attivazione di un neurone è più lenta rispetto a quella delle porte logiche;
* il comportamento è più rumoroso;
* rispetto al cervello umano, ha molta più capacità di generalizzazione.

L’obiettivo è quello di mantenere alcune capacità computazionali di un cervello e una velocità maggiore.

## Il Neurone

L’unità base della rete neurale è appunto il neurone, esso viene attivato dagli altri neuroni vicini accumulando gli input.

L’attivazione di un neurone consiste nella media pesata degli input, i pesi sugli archi indicano la forza della connessione e possono essere positivi o negativi, è inoltre presente un peso bias connesso a un input fisso, equivalente a -1.

La generalizzazione dell’output è una funzione di attivazione che può avere diverse forme, dal sigmoide alla gaussiana.

## Reti feed forward

Le reti feed forward sono reti neurali senza memoria in cui l’attività prosegue in una sola direzione, esse sono adatte per la classificazione.

Le reti feed forward sono formati da:

* un strato di neuroni che prende gli input, ciò viene detto input layer;
* un strato di neurone che produce l’output detto output layer;
* uno o più strati intermedi detti hidden layer.

In questa rete possono variare il numero di nodi e il numero di connessioni pesate.

## Percettroni

I percettroni sono le reti feed forward più semplici, qui non è presente un hidden layer e lo strato di output è formato da un solo neurone che effettua la media pesata dei suoi ingressi.

Questo tipo di rete permette di rappresentare in modo semplice le porte logiche e le funzioni linearmente separabili:

* L’AND viene simulato mettendo il bias a -1, ciò significa che l’uscita positiva si ottiene con tutti gli ingressi a 1;
* L’OR si simula mettendo il bias a -0.5, basta un solo ingresso a 1 per ottenere una risposta positiva.

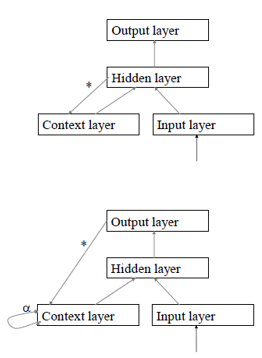
A causa della semplicità, i percettroni sono piuttosto limitati, tuttavia l’apprendimento dei pesi è semplice.

## Reti multilayer

Le reti multilayer sono reti neurali in cui possono variare il numero di nodi, di connessioni pesate e/o di hidden layer, inoltre:

* i neuroni degli hidden layer calcolano la media pesata dei propri ingressi e applicano la funzione di attivazione per ricavare l’output;
* i neuroni dell’output layer fanno essenzialmente la stessa cosa ma senza chiamare la funzione di attivazione.

## Reti ricorrenti

Le reti ricorrenti sono particolari reti neurali che, a differenza delle feed forward, hanno memoria, di conseguenza possono ricalcolare cose già viste.

Il fatto che queste reti abbiano memoria le rende utili in ambiti come il riconoscimento di pattern nei dati, la visione artificiale e l’analisi di serie temporali e di testi.

Alcuni tipo di reti ricorrenti sono:

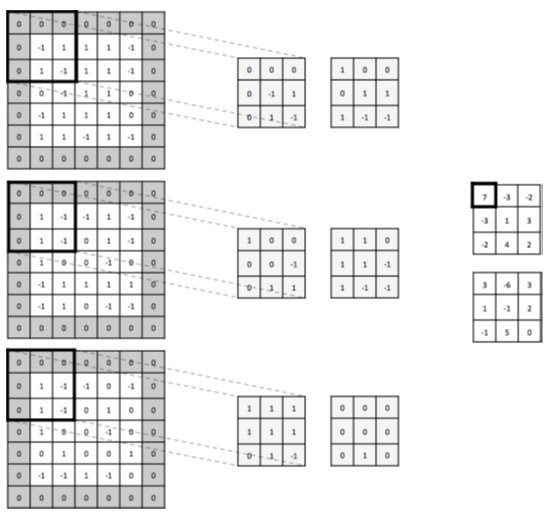
* La rete di Elman è una rete ricorrente che aggiunge un context layer, il suo compito è quello di ricordare l’output più recente dell’hidden layer per poi fornirlo nei calcoli successivi.
* La rete di Jordan, qui il context layer riceve in input l'output più recente della rete, sulla base di questi e quelli vecchi calcola i nuovi valori utilizzando un fattore di decadimento, il risultato verrà messo a disposizione agli hidden layer.

Esistono anche reti più sofisticate in cui è possibile decidere quando utilizzare gli output prodotti.

## Reti convoluzionali

Le reti convoluzionali sono particolari reti neurali in cui l’operazione più importante è la convoluzione. Precisamente si fa scorrere un filtro sull’input, esso viene fatto passare tra i nodi hidden e rappresenta l’insieme delle connessioni tra i neuroni.

Questa rete restituisce una feature map, indicante dove si trova l’elemento cercato.

L’attivazione di un neurone nella feature map in questo caso avviene quando l’elemento se il filtro ha individuato l’elemento da cercare negli input.

Com’è strutturato il filtro? essenzialmente è una matrice di convoluzione i cui valori all’interno vengono moltiplicati per quella parte di input che viene coperta.

Nel caso reale, la convoluzione opera sui volumi, il layer convoluzionale è quindi un’insieme di filtri che, dato un volume, ne produce un’altro in output.

La profondità del filtro equivale a quella dell’input, quella dell’output invece è il numero di filtri.

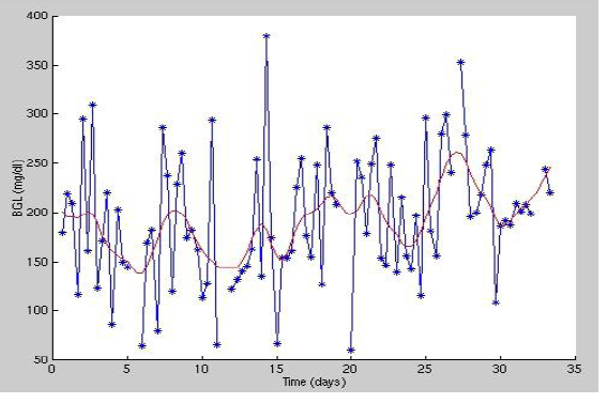
Alla fine della convoluzione si fa un passo di pooling in cui si riassumono le informazioni per ridurre la dimensionalità, in alcuni casi si perdono informazioni ma ciò non è un problema.

# Predizione delle serie temporali

Come si può predire lo sviluppo di una variabile nel tempo? Si utilizzano modelli matematici che calcola un valore sulla base dei precedenti.

## Decomposizione strutturale

Un modello matematico è la decomposizione strutturale, essa divide la serie temporale in tre parti:

* trend;
* ciclo o componente ciclica;
* rumore o componente stocastica.

Questo modello definisce inoltre una funzione di composizione che combina le tre componenti, di solito è la somma ma ne esistono anche di più complesse.

Il vantaggio di questo metodo è la possibilità di stimare le componenti un pezzo alla volta e poi metterle insieme, un esempio è la media mobile per estrarre il trend.

Il grafico a destra indica l’andamento dei pazienti diabetici, qui alcuni dati sono mancanti mentre altri contengono rumore, che si fa?

Ebbene, si possono stimare trend e rumore e da essi ricavare il ciclo, ottenendo la linea rossa.

### Interpretazione della decomposizione

Considerando l’esempio precedente, è possibile interpretare la decomposizione come segue:

* un’astrazione temporale di trend, ottenuta mettendo insieme i vari valori nello stesso tempo, essa indica il pattern di crescita, decrescita e la stazionarietà;
* un'astrazione temporale di cicli, indicante la ricerca dei pattern giornalieri di misure ordinate oppure la verifica delle oscillazioni.

Senza l’utilizzo della decomposizione, si può effettuare l’analisi, farla però è più faticosa a causa dell’andamento e della grandezza degli intervalli.

Rispetto alla sola analisi strutturare, l’interpretazione è più semplice ed è possibile comporre anche con altri metodi di time abstraction. Rispetto a quest’ultima invece l’applicazione su serie filtrate è più robusta, è più semplice trovare i cicli e ci sono meno problemi con outliers e dati mancanti.

## Exponential smoothing

L’exponential smoothing definisce una relazione ricorsiva su cui effettuare le previsioni:

* al passo base, si calcolano le previsioni a tempo 1 come valori attual;
* nel passo ricorsivo invece si combinano linearmente il valore misurato a tempo t con la previsione fatta quel tempo. Tenendo conto di un numero k di osservazioni passate, si possono considerare tutte per il calcolo della previsione (anche scartandole), magati pesando di più i casi più recenti.

Questo metodo è lineare e non tiene conto di trend, ciclo e rumore, ciò può essere un problema dato che i risultati potrebbero non essere realistici.

Inoltre presenta problemi nel caso in cui il segnale non sia lineare.

A fronte di ciò, esistono modelli più avanzati che tengono conto di queste componenti:

* il modello di Holt, il quale stima la componente di trend e la aggiorna ricorsivamente;
* il modello di Winters, in cui si stima anche la stagionalità.

## Come si misura la qualità del modello?

Per capire la qualità di un modello, si utilizza la matrice di confusione, una matrice in cui le righe sono i valori predetti mentre le colonne sono quelle reali. Sulla base di ciò, possono esserci quattro possibili casi:

* Se il valore predetto e quello reale sono veri, allora si mette vero nella cella;
* Si inserisce falso se entrambi i valori sono falsi.
* Con un valore predetto falso e un reale vero, si ha un falso negativo;
* Un falso negativo invece si ottiene con una previsione vera e un valore falso.

Come si procede? si calcola la precisione, il recall, l’error rate e l’accuratezza con le seguenti formule:

Tutto ciò varia anche in base al contesto, in generale però è bene evitare i falsi negativi.

# Ottimizzazione

La fase di ottimizzazione lavora in sincronia con quella di predizione e serve a ottenere il miglior risultato possibile. Come si valuta una soluzione?

Innanzitutto occorre definire come rappresentarla e come valutarne la qualità, la struttura infatti influisce sullo spazio di ricerca e sulla gestione dei vincoli, per quest’ultimo infatti si accettano le sole soluzioni che li soddisfano.

Per quanto riguarda la qualità di una soluzione, si cerca di misurarla quantificando i vantaggi e sottraendo i costi, in alcuni casi potrebbero essere stati violati dei vincoli.

La violazione di un vincolo può essere interpretata come una penalità maggiore, in altri casi invece è vietato violarli.

Settare la penalità di un vincolo in modo quantitativo è macchinoso, valori troppo alti infatti indicano che ci si accontenta di soluzioni accettabili e di bassa qualità, al contrario invece si rischia di accettare soluzioni che violano troppi vincoli.

Al fine di ottimizzare il più possibile, è utile una fase di preprocessing in alcuni casi, ciò si fa di solito in presenza di particolari vincoli.

E’ inoltre possibile definire delle penalità dinamiche inizialmente basse ma che aumentano nel corso dell’esecuzione, ciò permette una convergenza verso le soluzioni accettabili, tutto ciò però risulta molto macchinoso.

Nel caso in cui la soluzione viola un vincolo, questa viene riparata randomicamente oppure basandosi su un’euristica, ciò però potrebbe violare altri vincoli, inoltre i vincoli complessi potrebbero avere effetto su molte feature.

## Effetto Baldwin

L’effetto Baldwin consiste nella riparazione temporanea di una soluzione, come si fa?

Questo metodo valuta una soluzione cercando di capire come sarebbe dopo la riparazione, quindi basandosi sul loro stato attuale, si valutano le soluzioni in base al loro potenziale.

## Effetto Lamarck

L’effetto Lamarck è analogo al precedente con la differenza che qui la riparazione è permanente. Entrambe le strategie hanno pro e contro, quindi si utilizza una strategia mista, di solito per il 90% Baldwin e il 10% Lamarck.

## Ottimizzazione multiobiettivo

Con ottimizzazione multiobiettivo si vuole idealmente ottenere soluzioni ottimali verso più obiettivi che siano differenti e che non siano troppe.

Nella realtà invece ciò è molto difficile, per cui si confrontano tutti gli obiettivi sulla stessa scala, se ne seleziona uno come principale e si trattano tutti gli altri come vincoli.

## Ambiente dinamico

Un ambiente è considerato dinamico quando le sue leggi possono cambiare nel tempo così come le preferenze dell’utente, è quindi utile mantenere in memoria le precedenti decisioni dato che possono influenzare il presente.

## Ottimizzazione locale

Rispetto alla controparte globale (più costosa e difficile), l’ottimizzazione locale ha il compito di ottimizzare una soluzione rispetto al suo intorno nello spazio di ricerca, come si fa? Si utilizza un processo iterativo.

Un secondo metodo per ottimizzare localmente consiste nell’utilizzo di un trade-off, qui si possono avere due casi:

* con uno spazio di ricerca piccolo, si possono esaminare le soluzioni esaustivamente e da esse prendere la migliore, quest’ultima però non è detto che sia ottimale;
* un ragionamento più o meno duale si ottiene con spazi di ricerca grandi, qui è più facile trovare la soluzione ottimale ma non è detto che sia possibile valutare tutte le soluzione, nella pratica infatti ci si ferma dopo k iterazioni.

## Metodi di ottimizzazione

### Hill climbing

L’hill climbing consiste nel trovare il massimo in un intorno locale, come fa? l’algoritmo confronta la soluzione di partenza con quella selezionata prendendo la migliore delle due.

L’algoritmo continua finchè tutto l’intorno non è stato esaminato e passano k iterazioni dall’ultima soluzione migliore trovata.

Il problema di questa tecnica è il fatto che il massimo locale trovato potrebbe non coincidere con quello globale, inoltre non è nota la distanza verso quest’ultimo e si rischia di rimanere intrappolati in avvallamenti o pianori.

Ultima ma non meno importante, il risultato dipende fortemente dalla soluzione iniziale, per questo motivo si randomizza oppure si utilizzano delle euristiche.

### Hill climbing iterativo

L’hill climbing iterativo è un miglioramento dell’hill climbing in cui, sulla base di euristiche o casualmente, vengono generate N soluzioni iniziale.

Per ogni soluzione generata, si applica l’hill climbing normale e si prende la miglior soluzione, la quale non è per forza quella ottimale.

### Hill climbing stocastico

La versione stocastica dell’hill climbing ammette anche soluzioni peggiori rispetto a quella attuale, accettandola con una probabilità p che dipende da:

* la differenza in qualità delle due soluzioni;
* un parametro T costante, un valore elevato lo rende simile a una random search mentre un basso all’hill climbing normale.

### Simulated annealing

Il simulated annealing utilizza un concetto di temperatura per indicare la caoticità di una soluzione. Inizialmente la temperatura e alta, accettando quindi soluzioni peggiori più di frequente, durante l’esecuzione questa si abbassa.

Come per l’hill climbing stocastico, anche qui la soluzione peggiore viene accettata con una probabilità p, calcolata con la seguente formula:

Il simulated annealing prende e valuta una soluzione, procedendo come l’hill climbing per una data temperatura, dopodichè si procede con una temperatura differente.

### Tabu search

La tabu search è un tipo di ricerca che cerca di esplorare aree poco note dello spazio di ricerca, tenendo traccia delle parti già esplorate.

Queste soluzioni già esplorate sono dette tabu e non potranno più essere visitate, ciò permette di evitare di esplorare soluzioni troppo note, le quali potrebbero sfociare in avvallamenti i pianori.

Partendo da una memoria M inizialmente vuota, prende una soluzione S e la valuta.

Dopodichè considera più soluzioni dall’intorno di S, le valuta e prende la migliore, tutte le soluzioni non considerate vengono memorizzate in M per escluderle.

La memoria utilizzata è a breve termine, quindi memorizza per ogni soluzione l’iterazione più recente che viene adottata. In presenza di nuove soluzioni, si vanno a individuare come tabu solo quelle generate di recente, di conseguenza non possono essere scelte anche se migliori di quella attuale.

La brevità temporale della memoria alla lunga potrebbe essere penalizzante, di conseguenza sono stati introdotti dei miglioramenti:

* utilizzare una memoria a lungo termine frequency based che permette di memorizzare quante volte una feature è stata cambiata in un dato periodo. In presenza di soluzioni non tabu peggiorative, la memoria consiglia di operare su feature poco modificate, penalizzando quindi le soluzioni con feature che variano più di frequente;
* utilizzare l’aspiration criteria, ciò sovrascrivere la memoria seguendo dati criteri.

### Aspiration criteria

I criteri dell’aspiration criteria sono i seguenti:

* per obiettivo: accetta le soluzioni migliori di tutte quelle finora trovate;
* per default: sceglie le soluzioni meno recenti;
* per direzione di ricerca: sceglie le soluzioni verso delle direzioni che si sono rivelate promettenti in passato;
* per influenza: sceglie le soluzioni con il cambiamento più grande rispetto allo stato attuale.

### Algoritmi evoluzionistici

Gli algoritmi evoluzionistici sono particolari algoritmi che, date n soluzioni, esse vengono combinate per generare una seconda generazione.

Come si fa ciò? Si crea una popolazione di soluzioni generate casualmente, tutte queste vengono valutate.

Come si selezionano le soluzioni?

La scelta avviene come una sorta di torneo, in cui si estraggono due soluzione a caso scegliendo la migliore seconda una funzione di fitness, una funzione che valuta una soluzione in base a quanto è adatta alla risoluzione del problema.

Si ripete questo procedimento quant’è il numero di genitori che si intende ottenere.

Come vengono generate le soluzioni figlie?

Dall’insieme precedentemente ottenuto, si effettua il crossover, cioè che due genitori mischiano i valori delle feature, alcune di esse sono inoltre soggette a mutazioni, utile per introdurre cambiamenti.

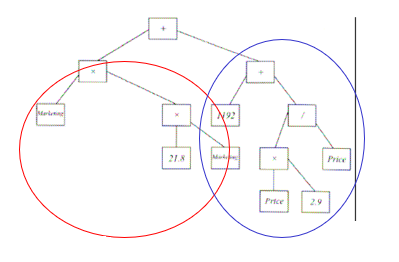
La mutazione è il cambiamento di alcuni valori negli attributi della soluzione, tutto ciò viene fatto con una probabilità tipicamente bassa.

La generazione successiva viene ottenuta ripetendo tutti i passi precedenti.

L’algoritmo termina quando viene prodotta la k-esima generazione di soluzioni oppure quando non ci sono più miglioramenti.

## Tecniche di previsione avanzate

### Algoritmo genetici

Gli algoritmi genetici permettono di apprendere la funzione che serve per calcolare i valori predetti.

Assumendo una struttura generale di una funziona f, all’inizio la struttura potrebbe essere sbagliata, tuttavia migliora iterativamente.

Come funziona? Per prima si selezionano i simboli terminali, cioè variabili e parametri, dopo si selezionano gli operatori e la funzione di valutazione.

Quest’ultima stima se la funzione descrive bene la relazione tra input e output, da essa si può inoltre calcolare l’errore di predizione.

Ultima ma non meno importante è la selezione dei parametri evoluzionistici, cioè la dimensione della popolazione, il numero di generazioni, la probabilità di mutazione, crossover, eccetera.

Le funzioni sono rappresentabili con una struttura ad albero in cui le foglie indicano le variabili, l’idea è quella di partire da funzione semplici e pian piano crearne di complesse.

Per via di ciò, queste funzioni potrebbero contenere calcoli inutili, inoltre possono occorrere molti dati per ottenere un risultato.

Per cercare di evitare tutto questo, si può introdurre della conoscenza di dominio per guidare il processo evoluzionistico.

### Sistemi ad agenti

Un sistema ad agenti è una simulazione in cui gli agenti interagiscono tra di loro sulla base di date regole. In questo modo è possibile ottenere diversi scenari possibili in cui trovare la soluzione sulla base delle interazione.

Tutto questo si fa in assenza di dati e partendo da strutture perlopiù semplici.

Questo algoritmo di predizione si basa su agenti semplici, questi segueno regole probabilistiche di comportamento e di interazione tra loro.

Le interazioni tra agenti e con l’ambiente permette di definire modelli complessi difficili da definire a priori.

Tutto lo sforzo qui è infatti focalizzato sulla definizione degli agenti e delle loro regole, la complessità di tutto il resto verrà quindi definita in base alla loro interazione.

## Tecniche di ottimizzazione avanzate

### Ant System

L’ant system è un sistema ad agenti in cui questi ultimi sono formiche che depositano sul terreno dei feromoni. Più feromone è stato depositato su dato percorso (scegliendo la soluzione ritenuta ottimale), maggiore è la probabilità che un’altra formica segua quella strada, quindi che altri agenti scelgano quella soluzione.

Questo metodo è utilizzato per l’ottimizzazione, ma come funziona?

Prima di tutto si genera una popolazione di formiche, ognuna di esse costruisce una soluzione, ogni soluzione viene valutata da ogni formica e, in base a ciò, le formiche rilasciano dei feromoni su di essa. Si continua così finchè non ci sono abbastanza soluzioni.

La valutazione di una soluzione da parte di una formica è influenzata, oltre che dai livelli di feromone su di essa, anche dalla conoscenza di dominio.

Il feromone in questo caso è un vero e proprio meccanismo di comunicazione tra formiche, infatti dopo un certo numero di interazioni esse assumeranno tutte un comportamento simile dal momento che preferiranno le soluzioni più popolari.

### Coevoluzione

La coevoluzione è un meccanismo utilizzato in teoria dei giochi e basato sugli algoritmi genetici, qui due popolazioni devono definire una propria strategia, prevedere quella nemica e da ciò decidere la prossima mossa. Queste due popolazioni si evolvono nel tempo e selezione le strategie vincenti a ogni interazione.

Inoltre le popolazioni stesse possono avere rappresentazioni differenti delle soluzioni.

## Approcci ibridi

Gli approcci ibridi permettono di definire concetti complessi con performance migliori mettendo insieme più tecniche.

Questi approcci permettono di superare le limitazioni delle singole tecniche.

L’idea del sistema ibrido è quella di costruire più modelli differenti e fargli predire una soluzione per un dato problema, tutte queste vengono poi combinate attraverso la media pesata o tramite un sistema di voting.

Il sistema ibrido varia a seconda della previsione, tuttavia esistono due famiglie:

* Bagging e boosting, i quali utilizzano gli stessi modelli;
* Stacking e gating, che utilizzano modelli differenti.

### Bagging

Nel bagging si utlizzano tutti modelli dello stesso tipo e vengono sviluppati in modo differente scegliendo diversi training test. I risultati vengono poi scelti in base a un classificazione (output più frequente) oppure una media pesata.

### Boosting

Il boosting costruisce i modelli iterativamente sfruttando il risultato del modello precedente. L’obiettivo infatti è quello di concentrarsi su casi più difficili da predire, mettendo pesi alti ai casi difficili del precedente modello.

Nel caso in cui il modello genere predizioni non accurate, questo sarà fortemente penalizzato, quindi sarà complementare rispetto al modello precedente.

### Stacking

Lo stacking utilizza differenti modelli i quali generano il loro output in modo indipendente, poi un altro modello raccoglie input e output e genera la predizione finale. Il problema qui è quello di selezionare i metodi e di suddividere il dataset per generare i training set dei vari modelli.

### Gating

Il gating utilizza un modello per scegliere il modello di predizione da utilizzare, quest’ultimo verrà utilizzato per generare la previsione finale.

Questa tecnica risulta utile nelle applicazioni time critical, tuttavia non è presente una vera e propria collaborazione e quindi non esistono combinazione di output diversi.

## Sistemi ibridi per l’ottimizzazione

Dal punto di vista dell’ottimizzazione, i sistemi ibridi possono essere utilizzati:

* in modo competitivo in cui ogni metodo cerca la soluzione ottima, il primo che la trova vince;
* in modo cooperativo, scambiandosi informazione in modo da orientare la propria ricerca nelle aree dello spazio di ricerca più promettenti.

L’idea di questo approccio è quella di far convergere la computazione verso una soluzione ottima con maggior probabilità.

# Adattamento

La fase di adattamento tratta il tuning dei parametri del modello, una fase di per sé critica in ogni modello. Cosa si può fare?

Ciò che è possibile è analizzare i nuovi dati prodotti dal modello, confrontarli con i valori effettivamente validi e da essi calcolare gli errori fatti dal modello. Tutto questo permette di procedere con un tuning dei parametri che migliori la qualità del modello.

Per tenere conto dei trade-off tra cambi frequenti e costosi e previsioni non aggiornate, è possibile sviluppare un modulo di adattamento che si occupa dell’aggiornamento dei parametri. Ciò che essenzialmente fa questo modulo è confrontare il dati prodotti dal modello con quelli reali, andando ad agire di conseguenza sull’ottimizzazione e/o sulla previsione.

## Tuning

La fase di tuning può essere fatta in due momenti distinti:

* tra due esecuzioni, effettuando un controllo deterministico senza però ricevere alcun feedback;
* durante l’esecuzione con un controllo adattativo che si basa sui feedback ricavati dai risultati parziali, ciò è utile per direzionare il cambiamento. In questo caso sono possibili tecniche evoluzionistiche.

L’implementazione di un modulo di adattamento permette di ottenere un update più rapido dei parametri, automatico e più frequente, con una scoperta immediata dei pattern significativi.