Riassunti del corso di Machine Learning e Data Mining

Federico Marchetti
Lorenzo Tullini

1 novembre 2021

Indice

Note

Questo documento **non** è un riassunto ufficiale, ma raccoglie il contenuto delle slide e degli appunti presi a lezione

Parte I

Machine Learning

Capitolo 1

I dati

Prima di poter imparare dai dati occorre dire cosa è un dato e quali tipi di dati esistono. Per prima cosa occorre considerare che esistono vari tipi di dato ognuno dei quali riflette una diversa tipologia di informazioni e consente di effettuare un diverso insieme di operazioni. Inoltre occorre tener conto del fatto che i dati sono di qualità differente, infatti è possibile che vi siano dati mancanti o errati a causa di errori in fase di raccolta. Vi è inoltre il problema riguardante i valori anomali, ovvero piccole quantità di dati che sono radicalmente diversi dagli altri, ci sono casi in cui questi sono rilevanti in quanto rappresentano condizioni fuori dalla norma ma comunque reali e altri in cui possono essere addirittura distruttivi in quanto le deviazioni sono causate da errori o da rumore.

1.1 Tipi di dato

Data Type		*Description*	*Examples*	*Descriptive statistics allowed*
Categorical	Nominal	The values are a set of labels, the available information allows to distinguish a label from another Operators: $=$ and $\neq$	zip code, eye color, sex, ...	mode, entropy, contingency, correlation, χ^2 test
	Ordinal	The values provide enough information for a total ordering Operators: $<>\leq\geq$	hardness of minerals, non–numerical quality evaluations (bad, fair, good, excellent)	median, percentiles, rank correlations
Numerical	Interval	The difference is meaningful Operators: $+-$	Calendar dates, temperatures in centigrades and Fahrenheit	average, standard deviation, Pearson's correlation, F and t tests
	Ratio	Have a univocal definition of 0 Allow all the mathematic operations on numbers	Kelvin temperatures, masses, length, counts	geometric mean, harmonic mean, percentage variation

Figura 1.1: Tipi di dato

Come si può vedere dalla tabella 1.1 i dati sono inizialmente distinti inizialmente tra numerici e categorici. A loro volta queste due categorie sono divise in nominali e ordinali per i tipi categorici e in intervalli e rapporti per i dati numerici. I dati categorici nominali sono costituiti da un insieme di etichette il cui confronto consente solo di distinguerle. Con questo tipo di dati si possono effettuare statistiche descrittive. I dati categorici ordinali invece rappresentano un insieme di valori dotati di una relazione d'ordine (ad esempio lo soddisfazione in un servizio di assistenza clienti: pessima, buona, ottima, eccellente). L'ulteriore operazione consentita su questo tipo di dati è il confronto (ad esempio "l'utente x ha avuto una migliore esperienza dell'utente y").
I dati numerici invece sono costituiti da numeri. Gli intervalli (*interval*) indicano dati per cui non è data una definizione univoca di "0" (ad esempio lo 0 dei gradi celsius e quello dei gradi Fahrenheit è diverso), tra dati di questo tipo hanno senso solo operazioni di somma e differenza. I dati numerici di tipo rapporto (*ratio*) invece sono dotati di una definizione univoca di "0" e quindi consentono ogni tipo di operazione matematica. La differenza fra intervalli e rapporti sta quindi nel fatto che gli intervalli hanno un valore dello 0 arbitrario mentre i rapporti hanno un valore dello 0 assoluto. Si veda la seguente tabella con le possibili trasformazioni accettate da ogni tipo di dato:

Il numero di valori caratterizza il tipo di dominio:

Data Type		*Transformation*	*Comment*	
Categorical	Nominal	Any one–to–one correspondence	the SSN can be arbitrarily reassigned (masking)	
	Ordinal	Any order preserving transformation new $\leftarrow f$(old) where f is a monotonic function	(bad, fair, good, excellent) can be substituted by (1,2,3,4)	
Numerical	Interval	Linear functions new $\leftarrow a + b *$ old	centigrades and Fahrenheit temperatures can be converted either way	
	Ratio	Allow any mathematical function, *standardization*, variation in percentage	Kelvin temperatures, masses, length, counts	

Figura 1.2: Trasformazioni consentite

- In domini discreti è accettato un numero di valori finito
- In domini continui è in teoria accettato un numero infinito di valori, nella pratica è accettato il numero di valori consentito della rappresentazione (es. float e double)

Tipicamente i dati nominali e ordinali sono discreti, preferibilmente binari, mentre intervalli e rapporti sono espressi tramite valori continui.

I metodi per memorizzare i dati possono essere di differente tipologia:

- **Tabelle**: si tratta di tabelle relazionali in cui il numero delle colonne è fisso, per questo motivo l'insieme degli attributi è fisso per ogni tupla
- **Transazioni**: ogni riga è caratterizzata da un TID (*tuple identifier*) e da una lista di items che cambia per ogni tupla. Un esempio di transazione è la relazione tra un identificativo di un cliente di un supermercato e l'insieme dei prodotti acquistati
- **Matrice di dati**: simili è una tabella in cui ogni valore è esclusivamente di tipo numerico
- **Matrice sparsa di dati**: matrice di dati in cui molti campi sono vuoti o nulli
- **Grafi di dati**

1.2 Qualità dei dati

Uno dei problemi più rilevanti nell'apprendimento dai dati è la loro qualità. Infatti, per costruire algoritmi di apprendimento in grado di produrre conoscenza corretta è necessario che i dati siano il più possibile completi e corretti.

1.2.1 Rumore

Esistono due definizione di rumore dei dati:

- Una modifica di un valore originale
- Dati senza interesse che sono mischiati a dati di interesse

Esempio 1.1

Nell'analisi degli accessi ad un sito web potremmo essere interessati ai soli accessi da parte di esseri umani. In questo caso gli accessi dovuti ai web crawler rappresentano rumore

1.2.2 Valori anomali (outliers)

Gli outliers sono valori anomali, le cui caratteristiche sono significativamente diverse da quelle dei restanti dati. Va notato che gli outlier non sono necessariamente errori, ma possono anche rappresentare eventi estremamente rari che dunque si discostano dagli altri dati. Occorre però capire se gli outlier sono di interesse o meno, perché potrebbero rendere più complesso il processo di apprendimento oppure addirittura deviarlo.

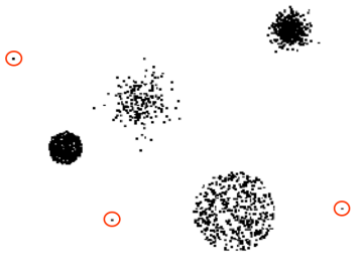


Figura 1.3: Outlier

1.2.3 Valori mancanti

Non è pensabile che un dataset sia perfetto, soprattutto perché spesso essi sono generati automaticamente, ad esempio, grazie a sensori che raccolgono e memorizzano i dati in maniera automatica. Per questo motivo, è lecito pensare che possano esservi valori mancanti. Questi possono essere dati che non sono stati raccolti oppure informazioni non utilizzabili.

Per risolvere questo problema si può:

- Ignorare ogni record contenente un valore mancante. Tipicamente questo riduce di molto la popolazione di dati rendendone disponibili molti meno rispetto a quelli che sarebbero in realtà disponibili
- Provare a sostituirli con una stima in base a tutti gli altri valori del data set (ad esempio con il valore medio, o con la mediana)
- Ignorare solo i valori mancanti e non gli interi record
- Inserire un valore possibile ragionando in termini di probabilità e di distribuzioni di probabilità. Occorre però che tali distribuzioni siano note prima dell'analisi dei dati

1.2.4 Dati duplicati

Così come possono esserci dati mancanti è anche possibile che all'interno di un dataset ci siano dati duplicati. Questo può verificarsi nel momento in cui si uniscono più fonti di dati diverse oppure a causa di errori nel momento in cui vengono raccolti i dati. La presenza di dati duplicati modifica le reali distribuzioni dei dati, attribuendo più peso ai record duplicati causando ad esempio overfitting.

In generale il processo riguardante la gestione dei dati duplicati è complesso, soprattutto se i dati sono duplicati soltanto in parte.

1.3 Pre-trattamento dei dati

Una parte rilevante del machine learning riguarda il pre trattamento dei dati. Con questo termine si intende una elaborazione preliminare dei dati affinché siano più adatti ad essere usati per l'apprendimento automatico.

Esistono varie tecniche di trattamento:

- Aggregazione
- Campiionamento
- Riduzione delle dimensionalità
- Creazione di feature
- Discretizzazione e binarizzazione

.

1.3.1 Aggregazione

Per **aggregazione** si intende la combinazione di due o più attributi a formare un nuovo singolo attributo. L'aggregazione si pone più obiettivi tra cui:

- **Riduzione dei dati**. Questo consente di ridurre il numero di attributi di un oggetto
- **Cambiare la scala** di visualizzazione al fine di modificare il livello di dettaglio. Ad esempio è possibile aggregare le città in province, regioni o stati
- **Aumentare la stabilità dei dati**. Aggregando i dati si tende a ridurne la variabilità, usando questa tecnica in maniera oculata diventa possibile ridurre anche controbilanciare i possibile effetti causati dal rumore all'atto della raccolta dei dati

1.3.2 Campionamento

Il **campionamento** è il più comune strumento di pre-trattamento dei dati, perché in agli albori dell'analisi dei dati gli statistici non erano dotati di strumenti automatici per il calcolo, per questo motivo avevano a che fare con un enorme mole di dati da dover raccogliere e processare a mano.
Anche oggi i processi di analisi hanno a che con un enorme mole di dati, per questo motivo vengono considerati dei sottoinsiemi dell'insieme iniziale dato che l'utilizzo dell'intero dataset potrebbe essere impossibile o troppo costoso da maneggiare. L'utilizzo di un campione dei dati può offrire quasi gli stessi risultati rispetto all'utilizzo del dataset completo, questo a patto che il campione sia rappresentativo della popolazione, ovvero che goda nel complesso delle medesime proprietà e possegga le medesime caratteristiche.

Il campionamento viene fatto per:

- Investigare in via preliminare i dati
- Analizzare i risultati finali
- Abbattere i costi
  - Ottenere l'intero dataset potrebbe essere impossibile o troppo costoso, quindi è utile ricorrere ad una prospettiva statistica sui dati stessi
  - Processare l'intero dataset potrebbe essere troppo costoso oppure potrebbe richiedere troppo tempo

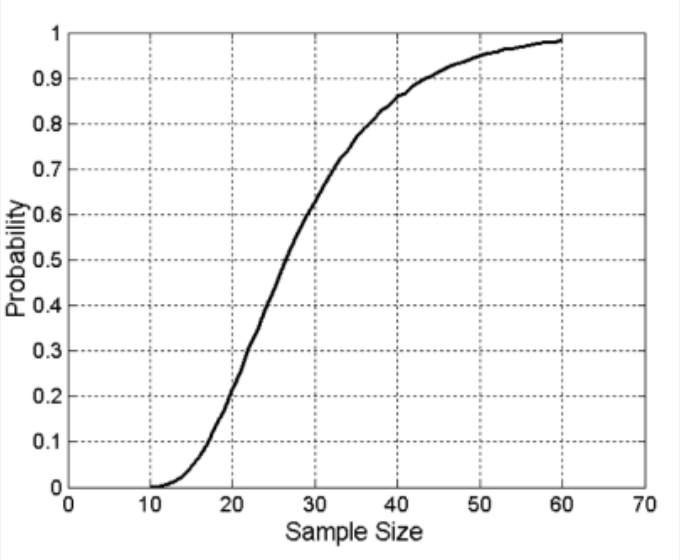
Esistono diversi tipi di sampling:

- Campionamento casuale semplice: scelta casuale di un singolo oggetto data una certa distribuzione di probabilità
- Con sostituzione: ripetizione di N estrazioni indipendenti. Questa tecnica è decisamente più semplice da implementare rispetto al campionamento senza sostituzione e più semplice da interpretare da un punto di vista statistico dato ch le estrazioni sono tra loro indipendenti
- Senza sostituzione: ripetizione di N estrazioni, ma ogni volta l'elemento estratto viene rimosso dalla popolazione iniziale in modo che non possa venir estratto nuovamente
- Stratificazione: in questo caso i dati vengono divisi in varie partizioni secondo un qualche criterio e successivamente da ogni partizione vengono estratti casualmente dei campioni. Questo tipo di campionamento viene usato tipicamente quando il dataset iniziale è diviso in sottoinsiemi diversi con caratteristiche eterogenee. La rappresentatività viene però garantita solamente all'interno della singola partizione e non relativamente all'intera popolazione

A questo punto occorrere chiedersi quale sia la dimensione corretta per un campione, in quanto è la dimensione minima del campione a garantirne la rappresentatività e il campionamento di per sé è un procedimento *lossy*. A tal proposito la statistica offre tecniche per comprendere quale sia la dimensione ottimale del campione e quale sia la significatività del campione in modo da poter trovare il miglior compromesso tra riduzione della mole di dati da analizzare e precisione.

Esempio 1.2: Campionamento e classi mancanti

Il grafico in figura mostra la probabilità di campionare almeno un elemento di ognuna delle dieci classi presenti nel dataset. È interessante notare inoltre che questo valore è indipendente dalla dimensione della popolazione stessa.



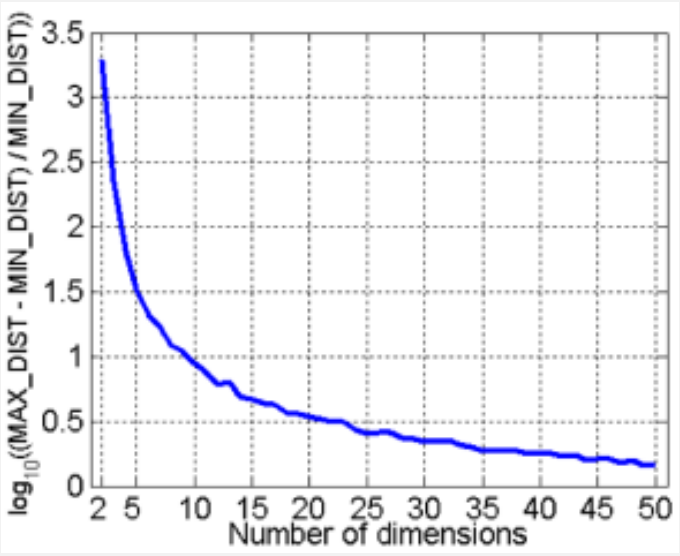
1.3.3 Riduzione della dimensionalità

Maledizione della dimensionalità

Nei casi in cui i dati siano rappresentati su molte dimensioni l'occupazione dello spazio diventa relativamente sparsa e la discriminazione dei dati sulla base delle distanze relative diventa complessa in quanto diminuisce la distanza minima media tra due punti. È interessante notare come questa maledizione non può essere spezzata mediante normalizzazione dei dati.

Esempio 1.3

In questo esempio sono stati generati 500 punti casuali in spazi aventi via via sempre più dimensioni. Il grafico mostra come all'aumentare del numero di dimensioni dello spazio decresce (molto rapidamente) la distanza relativa minima tra due punti.



Riduzione della dimensionalità

La riduzione della dimensionalità può essere effettuata per varie ragione oltre a scongiurare la maledizione prima esposta.

- Riduzione del rumore
- Riduzione della complessità spaziale e temporale degli algoritmi di mining
- Semplicità di visualizzazione. Per ovvi motivi gli unici spazi visualizzabili sono quelli a una, due e tre dimensioni
- Miglioramento della precisione del modello
- Riduzione della probabilità di overfitting

Vi sono varie tecniche atte a ridurre le dimensioni dello spazio in cui risiedono i dati:

- ***Principal component analysis***
- Singular values decomposition
- Tecniche supervisionate
- Tecniche non lineari

Principal component analisys L'idea alla base di questa tecnica è quella di trovare un nuovo spazio su cui proiettare in grado di catturare la massima variazione dei dati. Questo nuovo spazio è definito dagli autovettori della matrice di covarianza. Una volta effettuata la trasformazione il nuovo dataset conterrà solo gli attributi che catturano la variazione massima dei dati.

La prima dimensione selezionata è quella che cattura la maggior parte di variabilità, la seconda cattura la maggior parte della restante e così via, fino a che la variabilità rimanente è al di sotto della soglia desiderata. Da questo deriva che un piccolo numero di nuove variabili (ottenuto come combinazione lineare degli autovettori) cattura la maggior parte di variabilità dei dati.

Esempio 1.4: S

consideri il classico Iris dataset, si rappresentano in un grafico gli attributi sepalwidth e sepallength.

Nella figura di sinistra si possono vere i tre gruppi di fiori distinti mediante il colore dei punti. Si notano tre classi che tuttavia sono difficilmente divisibili. Questo è un set di dati numerici. Applicando PCA si nota che le prime due componenti ora catturano il 95% della varianza. Se si traccia in un grafico frazione di varianza per ogni componente principale, si nota che la prima componente cattura il 92% e la seconda una porzione molto esigua $(3-4\%)$

Scaling multidimensionale Questa tecnica non viene utilizzato per la selezione delle caratteristiche, ma piuttosto per la presentazione. Essa è in grado di ragionare bene in 2 dimensioni, quindi se si ha un'immagine 2-D possiamo facilmente affermare "questo è qui, questo è qui...".

Se abbiamo più di due dimensioni, possiamo scegliere le coppie in modo casuale o esaustivo oppure possiamo usare un metodo matematico per rimappare ogni punto in un nuovo spazio 2-D. Partendo dalla distanza tra gli elementi del dataset adatta la proiezione degli elementi in uno spazio ad m dimensioni in modo tale che le distanze tra gli elementi siano preservate. Esistono varianti di questo algoritmo sia per spazi metric che non metrici.

Selezione delle feature

La selezione di un sottoinsieme delle feature rappresenta una tecnica locale per ridurre la dimensione dello spazio in cui si trovano i dati ed allo stesso tempo per selezionare solo quegli attributi che ci consentano di effettuare il processo di apprendimento al meglio.

Può essere utilizzata per rimuovere gli attributi ridondanti, che dunque contengono informazioni già contenute in altri attributi, oppure per rimuovere attributi irrilevanti ai fini dell'apprendimento. Inoltre alcuni attributi possono essere fuorvianti e quindi confondere il processo di apprendimento (ad esempio nello studio della relazione tra salute, esercizio fisico, età e sesso se le età di uomini e donne appartengono a range diversi i risultati saranno lontani da quelli veri). Infine può accadere che un attributo sia fortemente correlato con un altro nella maggior parte dei casi ma non nei restanti in quanto i dati raccolti possono non essere omogenei. Si nota sempre quindi che i dati contengono informazioni da estrarre ma che essi

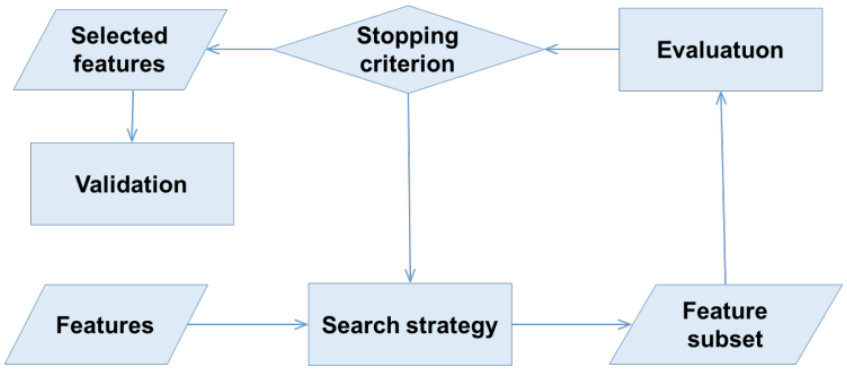


Figura 1.4: Architettura generale per la selezione delle feature da includere

non devono essere utilizzati senza prima ragionare sul loro significato.

Per ridurre la dimensione dello spazio in cui si trovano gli attributi sono possibili più tecniche a seconda che ci si trovi in presenza di dati supervisionati o meno:

- Se i dati sono non supervisionati è possibile utilizzare solamente un approccio a **forza bruta**. Esso consiste nel provare tutti i possibili sottoinsiemi di feature come input degli algoritmi di data mining e misurare l'efficacia di ogni ogni tentativo con il dataset ridotto al fine di trovare il sottoinsieme più adatto all'apprendimento in relazione alle dimensioni dello spazio che lo contiene
- Se siamo in presenza di dati etichettati, oltre a precedente approccio (comunque utilizzabile, anche se con pessime performance), è possibile utilizzare altre tecniche più efficaci:
  - **Approccio filtrante** (ovvero approcci indipendenti dagli schemi di esecuzione). Le feature di interesse vengono selezionate prima che gli algoritmi vengano fatti girare
  - Approcci dipendenti dallo schema di esecuzione
    - **Approccio implicito** (*embedded*). La selezione avviene naturalmente come parte dell'algoritmo stesso (ad esempio succede negli alberi decisionali dove vengono selezionati per primi gli attributi a cui è associato il maggior guadagno in termini di informazione, in questo caso se viene raggiunto il criterio di terminazione alcuni attributi non vengono considerati)
    - **Approccio wrapper**. È l'algoritmo stesso a scegliere il miglior sottoinsieme delle feature da considerare, similmente a come succede con l'approccio a forza bruta ma senza effettuare una ricerca esaustiva

Metodi filtranti Questi metodi si basano sulle caratteristiche generali dei dati. L'idea di base consiste nel selezionare un insieme di attributi indipendentemente dal modello di mining da utilizzare per apprendere pattern (per esempio attributi tra loro indipendenti che però correlano bene con la classe). Alcuni metodi sono:

- Correlazione di Pearson: una misura quantitativa (i cui valori vanno tra $[-1, 1]$) della dipendenza (o indipendenza) tra due variabili continue

- Liner Discrimination Analysis (LDA): usata per trovare combinazioni lineari di feature che caratterizzano o separano due o più classi
- ANOVA (Analysis Of Variance): simile a LDA se non per il fatto che usa feature categoriche ed una sola feature continua
- Chi-quadro: test statistico applicato ad un insieme di feature categoriche to valutare la verosimiglianza di correlazione o associazione tra di esse usando le distribuzioni di frequenza

Feature Response	*Continuous*	*Categorical*
Continuous	Pearson's Correlation	LDA
Categorical	ANOVA	Chi-Square

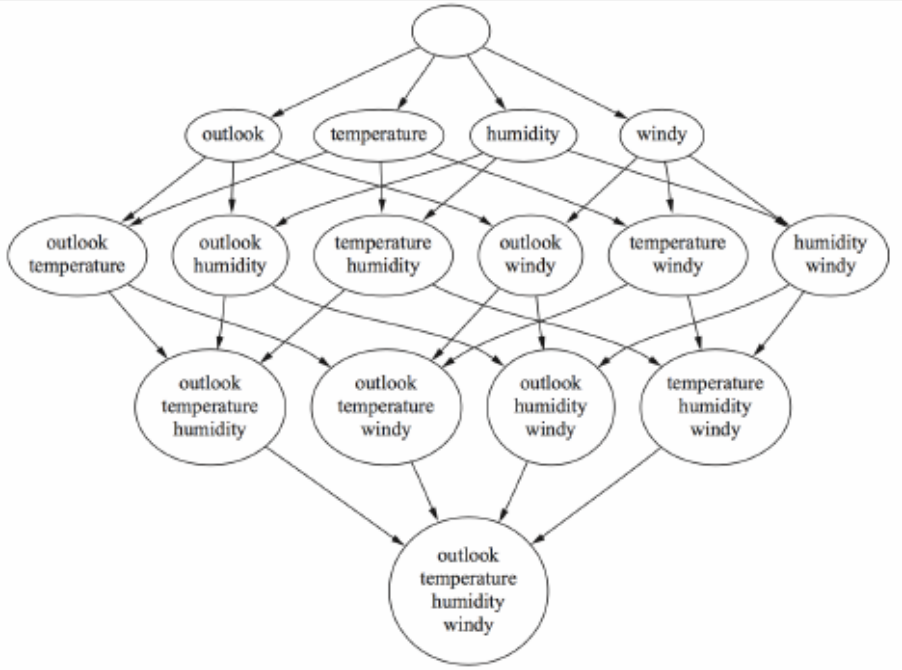
Figura 1.5: Schema riassuntivo dei metodi

In generale lo schema da seguire è nel suo complesso:

1. Insieme di tutte le feature
2. Selezione delle feature migliori
3. Utilizzo degli algoritmi di apprendimento
4. Tuning e valutazione delle performance

Metodi wrapper In generale questi metodi iterano i precedenti passi 2 e 3 più volte al fine di trovare il miglior sottoinsieme di feature. Questo fa sì che la complessità del problema non sia banale soprattutto in termini temporali, in quanto occorre allenare ripetutamente l'algoritmo di apprendimento. Tuttavia utilizzando questo schema il problema si riduce ad una ricerca all'interno di uno spazio degli stati.

Esempio 1.5



Usando un dataset per le previsioni meteo si può creare un grafo contenente tutte le possibili combinazioni di attributi e poi usare algoritmi di ricerca greedy per trovare il miglior insieme di feature

Differenze tra metodi wrapper e filtering Di seguito le principali differenze tra metodi di filtering e di wrapping:

- I metodi di filtering misurano la rilevanza delle caratteristiche in base alla loro correlazione con la variabile dipendente mentre i metodi di wrapping misurano l'utilità di un sottoinsieme di funzionalità addestrando effettivamente un modello
- I metodi di filtering sono molto più veloci rispetto ai metodi di wrapping in quanto non implicano l'addestramento di modelli. I metodi di wrapping sono molto costosi dal punto di vista computazionale
- I metodi di filtering utilizzano metodi statistici per la valutazione di un sottoinsieme di caratteristiche mentre i metodi di wrapping utilizzano la cross validation
- I metodi di filtering potrebbero non riuscire a trovare il miglior sottoinsieme di funzionalità in molte occasioni, mentre i metodi di wrapping possono sempre fornire il miglior sottoinsieme di features (quindi sono più affidabili)
- L'utilizzo del sottoinsieme di funzioni dai metodi di wrapping rende il modello più incline overfitting (saremo molto ben adattati a quello specifico set di dati) rispetto all'utilizzo di subset di feature date da metodi di filtering

1.3.4 Creazione di attributi

Aggiungere nuovi attributi aumenta la dimensione dello spazio in cui sono rappresentati di dati, ma permette anche di catturare più efficacemente ed efficientemente alcune caratteristiche dei dati. A tal proposito è possibile:

- Estrarre caratteristiche a partire dai dati già esistenti (ad esempio mappare strutture di pixel in tratti facciali come la distanza fra gli occhi)
- Mappare feature in un nuovo spazio (ad esempio nell'analisi audio spesso risulta utile passare al dominio delle frequenze da quello del tempo)
- Creazione di nuove feature (ad esempio mappare volume e peso nella densità)

1.3.5 Discretizzazione

Dietro alla necessità di discretizzazione risiedono varie motivazioni. Alcuni algoritmi, ad esempio, sono in grado di fornire risultati migliori in presenza di dati categorici dunque la discretizzazione consente di raggruppare dati originariamente appartenenti ad un dominio continuo. Inoltre avere un numero relativamente piccolo di valori distinti consente da un lato l'emergere di schemi in maniera più netta e dall'altro di diminuire l'impatto che il rumore ha sui risultati finali ottenuti.

Esempio 1.6

Nel considerare i salari dei dipendenti di un'azienda può convenire raggrupparli in classi anziché considerare ogni singolo valore

La discretizzazione può essere utilizzata per trasformare dati appartenenti ad un dominio continuo in dati appartenenti ad un dominio discreto (vedi Figura 1.6) oppure per ridurre il numero di valori di valori discreti all'interno di uno stesso dominio discreto. In entrambi i casi occorre utilizzare delle soglie (nel caso limite una sola soglia produce una binarizzazione dei dati) per dividere lo spazio in varie aree ognuna corrispondente ad un attributo discreto. In particolare nel secondo caso, ma anche nel primo, questa procedura deve essere guidata da una profonda conoscenza del dominio in modo da discretizzare i dati senza perdere troppe informazioni.

Come si vede in figura 1.6 sono possibili varie tecniche per disporre le soglie ognuna con pro e contro. La prima soluzione è la più semplice tuttavia non è la migliore in quanto taglia a metà alcune nuvole. La seconda soluzione è migliore rispetto alla precedente in quanto considera la distribuzione dei dati secondo frequenza, tuttavia taglia comunque alcune nuvole. La terza ed ultima soluzione invece è una tecnica adottata nel clustering che consente di isolare le varie nuvole in maniera ottimale.

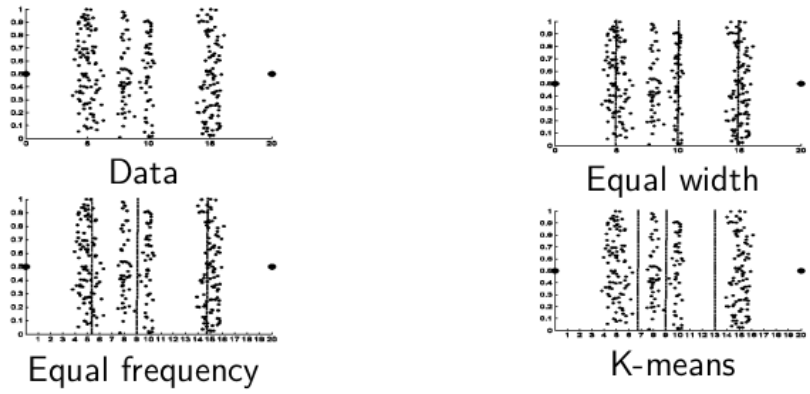


Figura 1.6: Trasformazione di dati da un dominio continuo ad un dominio discreto

Talvolta queste tecniche non sono sufficienti. Ad esempio `sklearn` implementa i metodi di classificazione solo per attributi numerici, pertanto non è in grado di elaborare attributi categorici in generale. In questi casi si può propendere per la binarizzazione di attributi discreti. Supponendo di avere un attributo con V valori discreti si possono generare V attributi binari mutuamente esclusivi (vedi Figura 1.7), ognuno dei quali è associato ad uno dei valori originali (può essere pensata come una stringa di V bit in cui un solo bit può essere 1).

Quality	*Quality–Awful*	*Quality–Poor*	*Quality–OK*	*Quality–Good*	*Quality–Great*
Awful	1	0	0	0	0
Poor	0	1	0	0	0
Ok	0	0	1	0	0
Good	0	0	0	1	0
Great	0	0	0	0	1

Figura 1.7: Binarizzazione di un attributo discreto

Trasformazione degli attributi

In questo caso vogliamo mappare l'intero insieme di valori in un nuovo dominio secondo una certa funzione (ad esempio x^k, $\log(x)$, $|x|$). In generale questa procedura modifica la distribuzione dei dati. Questo effetto potrebbe essere desiderato in modo da far emergere caratteristiche desiderate, in altri casi potrebbe essere deleterio in quanto rende più complesso notare certi schemi nei dati.

Se i valori originali seguono però una distribuzione normale è possibile trasformarli in modo che seguano una distribuzione normale standard (con media nulla e varianza unitaria). In questo caso la funzione specifica mappa

$$x \to \frac{x - \mu}{\sigma}$$

Un'ulteriore alternativa è data dalla normalizzazione. In questo caso i dati vengono modificati in modo che il range a cui appartengono sia $[0, 1]$ oppure $[-1, 1]$ senza però modificarne la distribuzione. Tipicamente viene effettuata questa scelta affinché le scelte effettuata dall'algoritmo non siano influenzate dalla scala dei dati. In questo caso la funzione specifica mappa

$$x \to \frac{x - x_{min}}{x_{max} - x_{min}}$$

oppure

$$x \to \frac{x - \frac{x_{max}+x_{min}}{2}}{\frac{x_{max}-x_{min}}{2}}$$

1.4 Somiglianza e diversità

Si dice **somiglianza** una misura numerica di quanto due oggetti siano simili. Tipicamente essa ricade all'interno dell'intervallo $[0, 1]$, dove 1 indica la massima somiglianza.

La **diversità** indica invece quanto due oggetti siano differenti. In questo caso tipicamente essa ricade nell'intervallo $[0, k]$, dove k è un valore dipendente dall'applicazione che indica il massimo grado di dissomiglianza.
Il termine **prossimità** può riferirsi, a seconda del contesto, sia alla somiglianza che alla diversità.

Dati due valori p e q a seconda del tipo di dato si possono definire somiglianza e diversità in vari modi:

- Per gli attributi nominali si ha:
  - $s = \begin{cases} 1 & \text{se } p = q \\ 0 & \text{if } p \neq q \end{cases}$
  - $d = \begin{cases} 0 & \text{se } p = q \\ 1 & \text{if } p \neq q \end{cases}$
- Per gli attributi ordinali (mappati su valori numerici nell'intervallo $[0, V-1]$) si ha:
  - $s = 1 - \frac{|p-q|}{V-1}$
  - $d = \frac{|p-q|}{V-1}$
- Per gli intervalli o i rapporti si ha:
  - $s = \frac{1}{1+d}$
  - $s = 1 - \frac{d - \min(d)}{\max(d) - \min(d)}$
  - $d = |p - q|$

1.4.1 Distanza

Qualsiasi sia il tipo di distanza scelto, vi sono delle proprietà comuni:

1. $dist(p, q) \geq 0 \ \forall p, q$
2. $dist(p, q) = dist(q, p)$
3. $dist(p, q) \leq dist(p, r) + dist(q, r) \ \forall p, q, r$

Distanza euclidea

Quando ci si riferisce alla diversità spesso si fa riferimento alla distanza euclidea (L_2)

$$L_2 = \sqrt{\sum_{d=1}^{D} (p_d - q_d)^2}$$

dove D è il numero di dimensioni che caratterizza lo spazio in cui sono espressi i dati, mentre p_d e q_d sono i d-esimi attributi degli oggetti p e q.

Tipicamente se le scale dei dati differiscono occorre effettuare un passo di standardizzazione (se i dati seguono entrambi una gaussiana) o di normalizzazione.

Distanza di Minkowski

Un caso più generale è dato dalla distanza di Minkowski

$$dist = \left(\sum_{d=1}^{D} |p_d - q_d|^r\right)^{\frac{1}{r}}$$

In questo caso r è un parametro scelto dipendentemente dal dataset e dall'applicazione finale.

Nel caso in cui si abbia $r = 1$ si ha la distanza di Manhattan (L_1). Essa è il modo migliore per distinguere gli oggetti con distanza pari a 0 dagli oggetti con distanza circa 0 in quanto qualsiasi variazione nella distanza tra i due oggetti si riflette direttamente sulla distanza complessiva. Per questo fatto e per

la sua semplicità è migliore della distanza euclidea nei casi in cui si abbia a che fare con spazi aventi molte dimensioni.

Nel caso in cui si abbia $r = 2$ si ricade nella distanza euclidea.

Nel caso in cui si abbia $r = \infty$ si ha la distanza di Chebyshev (L_∞) la quale considera solamente la dimensione in cui la differenza è massima, dando una valutazione semplificata scartando tutte le altre dimensioni.

$$dist_\infty = \lim_{r \to +\infty} \left(\sum_{d=1}^{D} |p_d - q_d|^r\right)^{\frac{1}{r}} = \max_d |p_d - q_d|$$

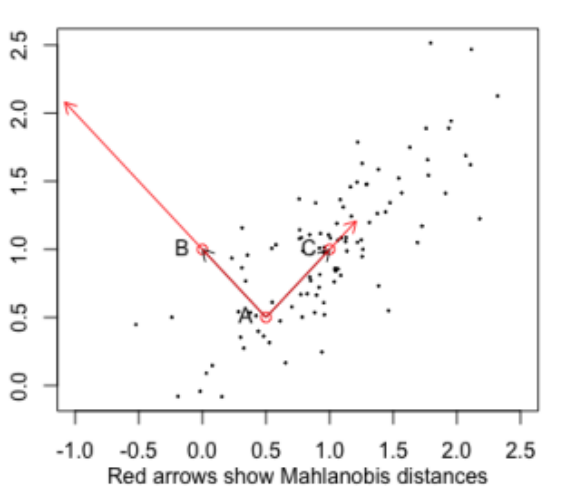
Distanza di Mahalanobis

Questa distanza considera la distribuzione dei dati. La distanza di Mahalanobis tra due punti p e q decresce se mantenendo la medesima distanza euclidea il segmento che connette i due punti è parallelo alla direzione in cui è massima la variazione dei dati.

La distribuzione è descritta usando la matrice di covarianza dei dati Σ:

$$dist_m = \sqrt{(p-q)\Sigma^{-1}(p-q)^T}$$

Esempio 1.7



Si hanno:

$$A = (0.5, 0.5) \quad B = (0, 1) \quad C = (1, 1)$$

$$\Sigma = \begin{bmatrix} 0.3 & 0.2 \\ 0.2 & 0.3 \end{bmatrix}$$

Si ha:

$$L_2(A, B) = 0.707 \quad dist_m(A, B) = 2.236$$

$$L_2(A, C) = 0.707 \quad dist_m(A, C) = 1$$

Matrice delle covarianze Tale matrice è una generalizzazione della covarianza al caso di dimensione dello spazio maggiore di due. Rappresenta la variazione di ogni variabile rispetto alle altre (inclusa se stessa). È una matrice simmetrica.

$$\Sigma_{ij} = \frac{1}{N-1} \sum_{k=1}^{N} (e_{ki} - \bar{e_i})(e_{kj} - \bar{e_j})$$

La diagonale della matrice contiene la varianza di ogni variabile. In generale una cella è positiva se le due variabili crescono assieme, mentre è negativa se mentre una cresce l'altra decresce. Nel caso in cui le variabili siano rappresentate da una gaussiana standard nella diagonale tutte le varianze sono unitarie. Se la matrice è diagonale allora le variabili sono indipendenti, mentre le nel caso di variabili indipendenti

distribuite secondo una gaussiana standard si ha una matrice identità e la distanza di Mahalanobis coincide con la distanza euclidea.

1.4.2 Somiglianza

La somiglianza gode di alcune proprietà generali:

1. $sim(p, q) = 1$se e solo se $p = q$
2. $sim(p, q) = sim(q, p)$

Somigliaza tra vettori binari

Nel caso di attributi binari si può procedere come segue. Detti:

- M_{00} il numero di attributi per cui $p = 0$ e $q = 0$
- M_{01} il numero di attributi per cui $p = 0$ e $q = 1$
- M_{10} il numero di attributi per cui $p = 1$ e $q = 0$
- M_{11} il numero di attributi per cui $p = 1$ e $q = 1$

si possono descrivere i coefficienti:

- Simple Matching coefficient (SMC): $SMC = \frac{M_{00}+M_{11}}{M_{00}+M_{01}+M10+M_{11}}$
- Coefficiente di Jaccard (JC): $JC = \frac{M_{11}}{M_{01}+M10+M_{11}}$

Caso generale

Nel caso più generale è possibile utilizzare altre metriche:

- Somiglianza cosinusoidale: $cod(p, q) = \frac{p \cdot q}{\|p\|\|q\|}$. Rappresenta il coseno dell'angolo compreso tra i due vettori
- Coefficiente di Jaccard esteso (Tanimoto): $T(p, q) = \frac{p \cdot q}{\|p\|+\|q\|-p \cdot q}$

1.4.3 Correlazione

In generale una correlazione è una relazione tra due variabili tale per cui a ciascun valore della prima corrisponda un valore della seconda, seguendo una certa regolarità. Nel caso in cui due variabili siano indipendenti il coefficiente di correlazione è nullo, mentre il fatto che il coefficiente di correlazione tra due variabili sia nullo non implica che esse siano indipendenti. Inoltre in generale valori positivi implicano una correlazione lineare positiva (Figura 1.8.)

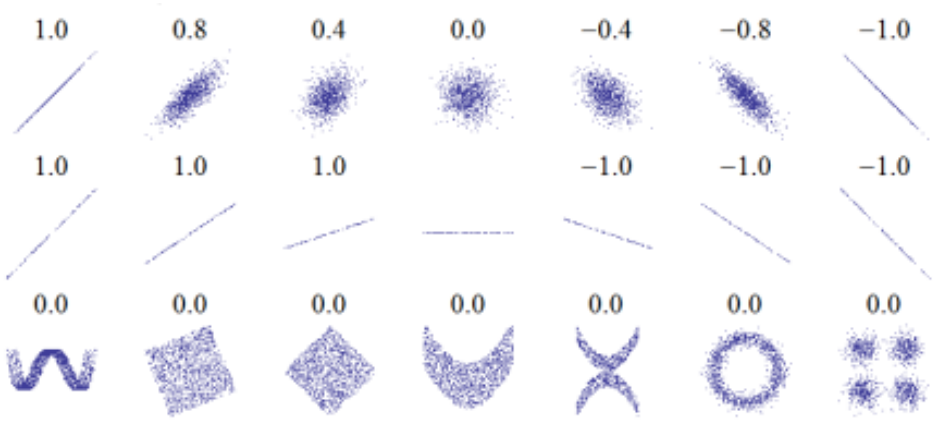


Figura 1.8: Esempio di correlazione di una nuvola di punti

Il calcolo della correlazione per dati numerici può essere eseguito in questo modo:

1. Per due attributi p e q considerare come vettori la lista ordinata dei valori per tutti i record
2. Standardizzazione dei valori dei vettori ottenendo i vettori p' e q'
3. La correlazione è data dal prodotto scalare tra i due vettori, ovvero

$$corr(p, q) = p' \cdot q'$$

Per quanto riguarda i dati nominali è possibile calcolare la correlazione attraverso il calcolo dell'incertezza simmetrica (symmetric uncertainty):

$$U(p, q) = 2\frac{H(p) + H(q) - H(p, q)}{H(p) + H(q)} \quad 0 \leq U(p, q) \geq 1 \forall p, q$$

dove $H()$ è l'entropia del singolo attributo mentre $H(,)$ è l'entropia comune dei due attributi.

Capitolo 2

Classificazione

La classificazione è una tecnica di apprendimento supervisionata, ovvero una tecnica in cui i dati usati per l'apprendimento sono accompagnati da ulteriori informazioni che specificano quale sia la classificazione corretta. Queste etichette (*labels*) possono essere ottenute in vari modi, come grazie all'intervento di esperti oppure grazie all'esperienza passata. Queste informazioni aggiuntive vengono utilizzate per imparare il corretto schema di classificazione, schema che poi può essere usato per classificare anche quagli individui sprovvisti di classe. Si consideri l'esempio dei semi di soia fatto in precedenza, l'attributo che definisce la malattia contratta dall'individuo è stato ottenuto grazie alla valutazione da parte di un esperto (**supervisore**).

Un modello di classificazione è un algoritmo che è in grado di associare una classe ad un individuo per cui questa è sconosciuta. Questo genere di algoritmi è di tipo parametrico in modo da poter effettuare ottimizzazioni specifiche per il caso in esame. Terminato l'addestramento il modello di classificazione viene valutato per capirne le performance e soprattutto la percentuale di successo (che stima la percentuale di successo per dati non etichettati). Una volta valutato un modello con caratteristiche soddisfacenti esso può essere utilizzato a run-time per classificare oggetti.

Il workflow tipico per la classificazione è il seguente:

1. Apprendimento del modello a partire dai dati contenuti all'interno del training set (processo induttivo). I dati all'interno del training set sono etichettati (non è detto che tutte le etichette siano corrette in quanto durante il processo di classificazione manuale potrebbero essere stati commessi degli errori) e dovrebbero essere rappresentativi dell'intero dataset (quindi scelti con un processo casuale in modo però da mantenere le relative proporzioni)

2. Stima della accuratezza del modello tramite labeling del test set (processo deduttivo). I dati contenuti all'interno del test set sono in realtà già classificati, questo consente di confrontare la classificazione operata dal modello con quella "reale" ottenendo quindi una percentuale di successo

3. Utilizzo del modello per classificare nuovi individui per cui non è disponibile una precedente classificazione

Le classificazioni possono essere di due tipi:

1. *Crisp classification*: il classificatore assegna una classe specifica ad ogni individuo

2. *Probabilistic classification*: il classificatore assegna ad ogni individuo una probabilità per ogni classe disponibile

2.1 Classificazione tramite alberi decisionali

Gli alberi decisionali sono uno degli strumenti più utilizzati e generano un classificatore strutturato ad albero. Ciò significa che, dato un individuo che si vuole classificare, posto nella radice dell'albero, si comincia a scendere verso le foglie. Ogni livello è caratterizzato ad un blocco condizionale in cui viene confrontato un attributo dell'individuo con un valore di test. In questo modo ad ogni livello viene effettuata una scelta in cui il valore degli attributi guida la discesa verso le foglie, le quali sono associate alla classe a cui l'individuo appartiene. Il raggiungimento di una foglia significa che non sono più disponibili domande da porre all'individuo per cercare di classificarlo. La più grade limitazione di questo albero è che è possibile considerare solo un attributo per volta.

2.1.1 Generazione del modello

Date le considerazioni precedenti è evidente come il fulcro dell'algoritmo C4.5 siano le domande da porre agli individui, ovvero un insieme E di elementi per cui la classe è nota l'albero decisionale cresce nella maniera seguente:

- se tutti gli elementi appartengono alla stessa classe c oppure E è piccolo, l'algoritmo genere una foglia con l'etichetta c
- l'algoritmo sceglie un test basato su un singolo attributo con due o più risultati e rende questo test la radice di un albero con un ramo per ciascuno dei risultati del test. Quindi l'algoritmo partiziona E in un certo numero di sottoinsiemi, ognuno corrispondente ad uno dei risultati del test. A questo punto viene riapplicata da capo l'intera procedura ai sottoinsiemi

A questo punto si pongono alcuni problemi. In primo luogo occorre decidere quali siano gli attributi "interessanti", ossia quelli da usare nei vari test. Inoltre occorre definire quali tipi di test possono essere effettuati per ognuno dei tipi di dato. Infine, occorre definire cosa significhi il fatto che "E è piccolo". Per risolvere tutti questi problemi si ricorre a strumenti statistici.

Esempio 2.1

age	employmer	education	edun	marital	...	job	relation	race	gender	houn	country	wealth
39	State_gov	Bachelors	13	Never_mar	...	Adm_cleric	Not_in_fam	White	Male	40	United_Sta	poor
51	Self_emp_r	Bachelors	13	Married	...	Exec_mana	Husband	White	Male	13	United_Sta	poor
39	Private	HS_grad	9	Divorced	...	Handlers_c	Not_in_fam	White	Male	40	United_Sta	poor
54	Private	11th	7	Married	...	Handlers_c	Husband	Black	Male	40	United_Sta	poor
28	Private	Bachelors	13	Married	...	Prof_specia	Wife	Black	Female	40	Cuba	poor
38	Private	Masters	14	Married	...	Exec_mana	Wife	White	Female	40	United_Sta	poor
50	Private	9th	5	Married_sp	...	Other_serv	Not_in_fam	Black	Female	16	Jamaica	poor
52	Self_emp_r	HS_grad	9	Married	...	Exec_mana	Husband	White	Male	45	United_Sta	rich
31	Private	Masters	14	Never_mar	...	Prof_specia	Not_in_fam	White	Female	50	United_Sta	rich
42	Private	Bachelors	13	Married	...	Exec_mana	Husband	White	Male	40	United_Sta	rich
37	Private	Some_colle	10	Married	...	Exec_mana	Husband	Black	Male	80	United_Sta	rich
30	State_gov	Bachelors	13	Married	...	Prof_specia	Husband	Asian	Male	40	India	rich
24	Private	Bachelors	13	Never_mar	...	Adm_cleric	Own_child	White	Female	30	United_Sta	poor
33	Private	Assoc_acd	12	Never_mar	...	Sales	Not_in_fam	Black	Male	50	United_Sta	poor
41	Private	Assoc_voc	11	Married	...	Craft_repai	Husband	Asian	Male	40	*MissingVa	rich
34	Private	7th_8th	4	Married	...	Transport_r	Husband	Amer_India	Male	45	Mexico	poor
26	Self_emp_r	HS_grad	9	Never_mar	...	Farming_fis	Own_child	White	Male	35	United_Sta	poor
33	Private	HS_grad	9	Never_mar	...	Machine_o	Unmarried	White	Male	40	United_Sta	poor
38	Private	11th	7	Married	...	Sales	Husband	White	Male	50	United_Sta	poor
44	Self_emp_r	Masters	14	Divorced	...	Exec_mana	Unmarried	White	Female	45	United_Sta	rich
41	Private	Doctorate	16	Married	...	Prof_specia	Husband	White	Male	60	United_Sta	rich

In questa tabella è rappresentata una porzione dei dati del censimento degli anni 1994/1995. Il dataset completo contiene circa 300.000 record in forma tabellare. Supponiamo di voler predire il benessere economico di ognuno degli individui a partire degli altri attributi.
Per prima cosa occorre valutare il dataset a mano per analizzare la frequenza degli attributi ad esempio utilizzando istogrammi o delle contingency table. L'utilizzo di questi strumenti, in particolare delle contingency table può permettere di intuire una correlazione tra i dati e la classe, tuttavia la dimensione di questo genere di tabelle cresce piuttosto velocemente, dunque nel caso in cui si vogliamo osservare allo stesso tempo molti attributi diventa complesso visualizzare le relazioni tra i dati. Ad esempio avendo 16 attributi ci sono 120 diverse tabelle bidimensionali e 560 diverse tabelle tridimensionali.

L'esempio mostra come siano necessari altri strumenti più generali al fine di valutare correttamente le relazioni tra i dati

2.1.2 Entropia e Information Gain

Uno dei metodi più utilizzati per capire se un pattern è interessante o meno è quello di utilizzare il concetto di entropia.

Esempio 2.2

Consideriamo una trasmissione di bit, si vogliono trasmettere quattro simboli: A, B, C e D, ogni simbolo appare nella trasmissione con probabilità

$$P(A) = 0,25\ P(B) = 0,25\ P(C) = 0,25\ P(D) = 0,25$$

Possiamo associare ad ogni simbolo una coppia di bit

$$A = 00, B = 01, C = 10 e D = 11$$

Data la sequenza $BAACBADCDADDDA$ la frase trasmessa sarà 01000010010011101100111111100.
Se prediamo però come assunzione che la probabilità dei simboli non sia uguale per tutti, ma sia

$$P(A) = 0,5, P(B) = 0,25, P(C) = 0,125, P(D) = 0,125$$

si può ottenere una codifica binaria che faccia uso di meno bit, nello specifico una media di 1,75 bit per simbolo, ovvero

$$A = 0, B = 10, C = 110 e D = 111$$

Questo deriva dal fatto che verranno usati meno bit per i simboli più frequenti e più bit per i simboli meno frequenti.

Generalizzando l'esempio appena fatto si giunge al concetto di entropia.

Definizione 2.1: Entropia

Data una sorgente X con V possibili valori aventi distribuzione di probabilità

$$P(v_i) = p_i \quad \sum_{i=1}^{V} p_i = 1$$

La migliore codifica conterrà un numero di simboli medio pari a:

$$H(X) = -\sum_{j} p_j \log_2 p_j$$

dove $H(X)$ è l'entropia della sorgente X

Il significato della formula precedente è che se si ottiene un valore di entropia alto allora la probabilità di apparizione dei simboli è simile tra di loro e non è possibile fare grosse previsioni su di questi perché vi è molta confusione, al contrario un valore di entropia bassa comporta che alcuni simboli abbiano probabilità di comparire nel testo molto più alta di altri, un istogramma che rappresenta il numero di occorrenze di ogni simbolo ha sicuramente dei picchi.

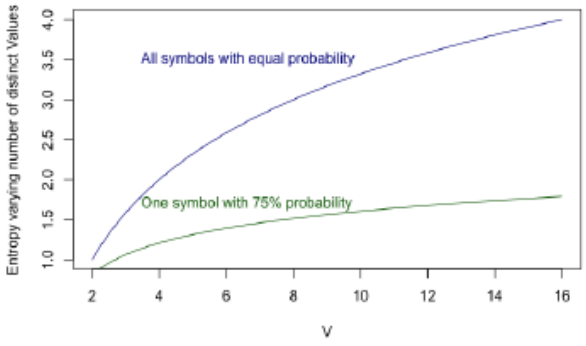


Figura 2.1: Livelli di entropia all'aumentare del numero di simboli. Nel caso in cui tutti i simboli abbiano la medesima probabilità il livello di entropia è maggiore, nel caso in cui uno dei simboli abbia invece probabilità pari al 75% l'entropia è minore

Dal concetto di entropia si può definire il concetto di *entropia condizionata specifica*.

Esempio 2.3

Si consideri come esempio un dataset sotto forma di tabella relazionale con due colonne: il titolo di laurea e un valore binario che indica se all'individuo è piaciuto o no il film Il Gladiatore

X	Y
Math	yes
History	no
CS	yes
Math	no
Math	no
CS	yes
History	no
Math	yes

Da questa tabella è possibile derivare alcune probabilità:

$$P(LikeG = yes) = 0.5$$
$$P(Major = Math \wedge LikeG = no) = 0.25$$
$$P(Major = Math) = 0.5$$
$$P(LikeG = yes \vee Major = History) = 0$$

Definizione 2.2: Entropia condizionata specifica

Definiamo entropia condizionata specifica $H(Y|X = v)$ come l'entropia di Y considerando solo le righe per cui si ha $X = v$

Esempio 2.4

Riprendendo il precedente esempio si ha ad esempio:

$$H(Y) = -0.5 \log_2 0.5 - 0.5 \log_2 0.5 = 1$$
$$H(Y|X = Math) = -0.5 \log_2 0.5 - 0.5 \log_2 0.5 = 1$$
$$H(Y|X = History) = -1 \log_2 1 - 1 \log_2 1 = 0$$
$$H(Y|X = CS) = -1 \log_2 1 - 1 \log_2 1 = 0$$

Definizione 2.3: Entropia condizionata

Definiamo entropia condizionata $H(Y|X)$ come la media pesata delle entropie condizionate specifiche di Y considerando ogni valore di X

$$H(Y|X) = \sum_j P(X = v_j) H(Y|X = v_j)$$

Questo valore rappresenta anche il numero medio di bit necessari a trasmettere il valore di Y se entrambi le parti della comunicazione conoscono il valore di X

Esempio 2.5

Riprendendo il precedente esempio si ha ad esempio:

$$\begin{aligned} H(Y|X) &= 0.5 * H(Y|X = Math) + 0.25 * H(Y|X = History) + 0.25 * H(Y|X = CS) \\ &= 0.5 * 1 + 0.25 * 0 + 0.25 * 0 = 0.5 \end{aligned}$$

Come si vede da questo esempio conoscendo il valore di X l'entropia di Y cala a 0.5

Definizione 2.4: Information Gain

Definiamo l'"information gain" (IG) come il quantitativo di informazione che la conoscenza di X riesce a dare per la predizione di un valore di Y, ovvero

$$IG(Y|X) = H(Y) - H(Y|X)$$

Rappresenta il numero di bit che mediamente è possibile non trasmettere se entrambi i capi della trasmissione conoscono il valore di X

Esempio 2.6

Riprendendo il precedente esempio si ha ad esempio:

$$IG(Y|X) = 1 - 0.5 = 0.5$$

Riconsideriamo l'esempio del censimento, e facciamo delle considerazioni sull'entropia condizionata specifica del benessere economico relazionato al genere della persona. Notiamo che $IG(wealth|gender) = 0,0367$ circa, ciò significa che il genere permette di dare intuizioni per quanto riguarda il benessere economico di un individuo del set.

Questo risultato è importante perché ci permette di definire quali siano gli attributi da testare ad ogni livello dell'albero decisionale dell'algoritmo C4.5.

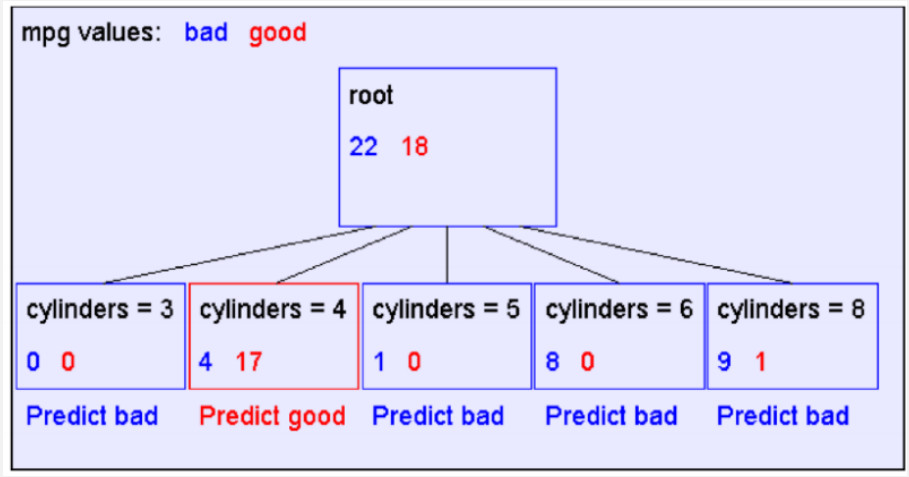
2.1.3 Costruzione dell'albero decisionale

Da quanto appena detto risulta abbastanza intuitivo il fatto che verranno testati prima i le feature aventi un valore di IG più alto, ovvero quelle feature che permettono di ottenere il maggior quantitativo di informazione possibile tramite un singolo test. Via via verranno usati gli attributi con IG minore, i quali catturano sempre meno informazioni.

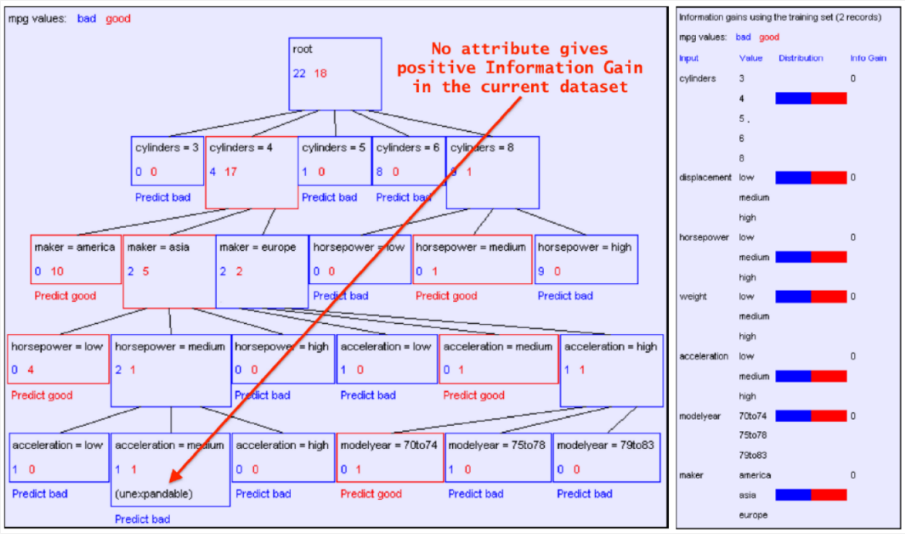
Esempio 2.7

cylinders	*displacement*	*horsepower*	*weight*	*acceleration*	*model_year*	*origin*	*mpg*
6	low	medium	medium	medium	75to78	europe	good
8	high	medium	high	medium	75to78	america	good
8	medium	medium	high	medium	70to74	america	good
5	low	low	medium	high	79to83	europe	good
3	low	medium	low	low	75to78	japan	good
4	low	low	low	medium	70to74	japan	bad
8	high	high	high	low	75to78	america	good
6	medium	low	medium	medium	79to83	america	good
8	medium	medium	medium	low	70to74	america	good
4	low	low	low	high	75to78	america	good
6	medium	low	medium	medium	75to78	america	good
6	medium	low	medium	medium	75to78	america	good
6	medium	low	medium	medium	70to74	america	good
8	high	high	high	low	75to78	america	good
6	low	medium	low	low	79to83	america	bad
8	high	medium	medium	medium	79to83	america	good
4	low	low	low	medium	79to83	america	bad
...	...	...	...	...	...	...	...

In questo esempio consideriamo gli attributi associati a delle automobili e si vuole prevedere qualitativamente come sarà il consumo di carburante (colonna *mpg*, ovvero *miles per gallon*). Per fare questo calcoliamo l'information gain dei vari attributi e notiamo che il maggiore tra di essi è quelli della colonna *cilinders*. A questo punto è possibile partizionare il dataset sulla base dei valori di questo attributo.



A questo punto si rimuove la colonna della cilindrata da quelle tra cui è possibile scegliere e si ripete ricorsivamente la stessa procedura ottenendo



2.1.4 Errori ed overfitting

Una volta generato il modello questo può essere valutato facendogli classificare i dati appartenenti al test set. La qualità del modello può essere valutata ome percentuale di oggetti classificati correttamente rispetto al totale degli oggetti. L'errore rilevato è chiamato *training set error* e spesso è maggiore di zero, possiamo dire quindi che esso è una stima dell'errore minimo che si può avere in un processo di classificazione su un set di dati reali. Tuttavia risulta molto più interessante l'errore massimo.

Per avere una stima dell'errore massimo che è possibile avere si può dividere il dataset in due parti:

- Training set: usato per generare il modello di classificazione
- Test set: usato per calcolare il test set error

Il test set error è più indicativo sulle prestazioni che il modello realmente avrà nel caso in cui debba classificare dati nuovi.

Overfitting

Definizione 2.5: Overfitting

L'overfitting è un fenomeno di disturbo che abbassa l'accuratezza di un modello di classificazione ed è principalmente dovuto alla presenza di rumore all'interno del training set. In questo caso il modello apprende anche il rumore

È possibile rilevare la presenza di overfitting con relativa facilità, in quanto si è in presenza di overfitting se vi è una forte discrepanza tra il valore dell'errore sul training set ed il valore dell'errore sul test set. In presenza di overfitting è utile ridurre il grado di addestramento del modello in modo da evitare che appenda anche il rumore e gli errori.

Detto più formalmente

Definizione 2.6: Overfitting

Un albero decisionale è una ipotesi della relazione tra un attributo usato per predire una classe e la classe stessa. Detta h l'ipotesi, $error_{train}(h)$ l'errore dell'ipotesi sul training set e $error_{\epsilon}(h)$ l'errore dell'ipotesi sull'intero dataset possiamo dire che siamo in presenza di overfitting del modello h se esiste un modello alternativo h' tale per cui

$$error_{train}(h) < error_{train}(h')$$
$$error_{\epsilon}(h) > error_{\epsilon}(h')$$

Ci sono varie cause per quanto riguarda l'overfitting oltre alla presenza di rumore nel dataset. Un altro motivo che spesso si presenta è la mancanza di istanze rappresentative: questo significa che alcune situazioni che accadono nel mondo reale non vengono (o non possono essere) rappresentate nel dataset.

Limitare il training set può aiutare a ridurre l'overfitting, ma questo non può essere fatto in maniera casuale, spesso si usano metodologie di potatura dell'albero decisionale che seguono linee guida precise e quantitative.

2.1.5 Potatura dell'albero decisionale

Esistono due tecniche principali di potatura:

- Pre-pruning, prevede di imporre un limite a priori oltre cui l'albero non può crescere
- Post-pruning, prevede di costruire l'intero albero e potare successivamente i rami seguendo precisi criteri

Solitamente le tecniche di post-pruning sono preferibili in quanto non è semplice stabilire a priori quale sia la profondità massima dell'albero.

Le principali tecniche che permettono di potare l'albero sono:

- Validation set: consiste nell'usare un dataset supervisionato distinto per valutare gli effetti della potatura
- Statistical Pruning: consiste nell'attuare test di tipo statistico per stimare se eliminare o no un determinato nodo
- Principio del minimum description length (MDL): consiste nell'utilizzo di una misura esplicita della complessità necessaria a codificare il training set e l'albero. La crescita dell'albero viene arrestata nel momento in cui

  $$size(size(tree) + size(misclassifications(tree, train)))$$

  è minima

Se da un lato potare l'albero troppo poco può generare overfitting, potarlo troppo al contrario genera undefitting (una situazione in cui il modello non rispecchia a sufficienza i dati usati per l'addestramento).

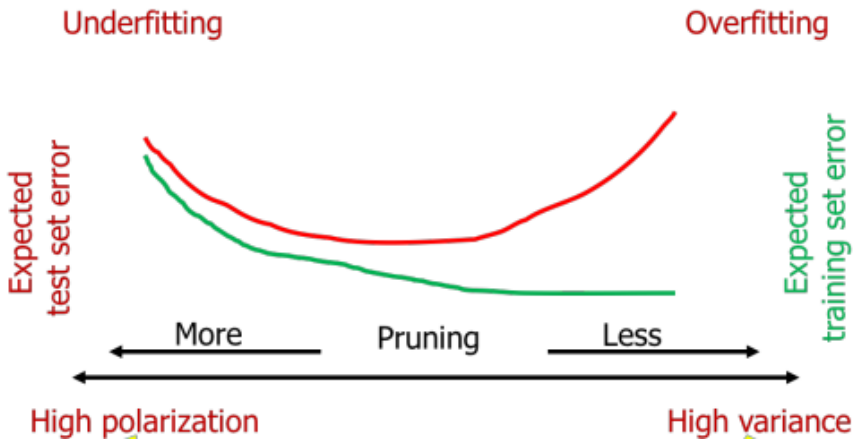


Figura 2.2: Nel caso in cui l'albero venga potato eccessivamente il modello classifica correttamente solo pochi casi, mentre se l'albero viene potato troppo poco vengono classificati bene molti casi appartenenti al training set, tuttavia l'errore nel caso del test set è comunque elevato in quanto il modello ha appreso anche il rumore insito all'interno del dataset stesso

Per questo motivo le metodologie di potatura devono essere precise (vedi figura 2.2) per ottenere un effettivo miglioramento delle performance del modello.

Una prima miglioria che può essere apportata consiste nell'introdurre un validation set. Ciò significa che il dataset supervisionato è diviso ora in tre sottoinsiemi indipendenti:

- Training set: consente la costruzione del modello
- Validation set: consente di ottimizzare il modello per ridurre quanto più possibile l'errore
- Test set: consente di calcolare l'errore finale atteso

Minimum description length

Il principio del Minimum Description Length (MDL) si basa sul fatto che il processo di apprendimento produce una teoria riguardante un insieme di dati. La nuova teoria può essere usata per predire valori su nuovi dati però, allo stesso tempo, la teoria può commettere errori.
Sia la teoria che gli errori possono essere codificati (gli errori possono essere visti come eccezioni alla teoria). La dimensione della descrizione della teoria è data dalla somma delle dimensioni della codifica della teoria stessa e degli errori. Il principio del Minimum Description Length asserisce che è preferibile la teoria la cui descrizione ha dimensione minima.

Per capire meglio questo principio si considerino due soggetti, A e B, questi conoscono gli attributi D_p utili alla classificazione per ogni elemento e. In più A conosce il valore della classe c per ogni elemento e. Chiediamoci ora quale sia il modo più compatto per trasmettere la conoscenza che A possiede in più rispetto a B. Esistono due modi principali, o si trasmette tutto il dataset di A a B oppure si trasmette una teoria e le sue eccezioni. Data la teoria T e il training set ϵ, sono definiti:

- $L(T)$ la lunghezza della codifica di T in bit
- $L(\epsilon|T)$ la lunghezza della codifica delle eccezioni di T (gli errori) in bit

Detto ciò, si vuole trovare la teoria più probabile. In accorda con il teorema di Bayes:

$$P(T|\epsilon) = \frac{P(\epsilon|T)P(T)}{P(\epsilon)}$$

Che è equivalente a:

$$-\log P(T|\epsilon) = -\log P(\epsilon|T) - \log P(T) + \log P(\epsilon)$$

Massimizzare la probabilità significa minimizzare il valore assoluto del logaritmo, perciò, dato che $P(\epsilon)$ è un valore noto, significa minimizzare:

$$-\log P(\epsilon|T) - \log P(T)$$

il valore è strettamente correlato con la minimizzazione della codifica:

$$-L(\epsilon|T) - L(T)$$

Pruning in pratica

L'algoritmo di apprendimento con albero decisionale stato ideato nella sua forma iniziale da Hunt nel 1962. Questo algoritmo usa una strategia greedy: ad ogni iterazione è scelto un ottimo locale e non viene mai fatto backtracking per cambiare la scelta effettuata. L'algoritmo possiede diversi gradi di libertà come la possibilità di specificare il test sull'attributo scelto, la possibilità di scegliere l'attributo utilizzato per dividere il dataset, il momento in cui smettere di dividere il dataset. È importante notare però che questo algoritmo può dividere il dataset solamente sulla base di un attributo per volta, questa è una importante limitazione perché possono verificarsi situazioni in cui la ramificazione migliore sarebbe da effettuarsi su due attributi contemporaneamente (vedi figura 2.3).

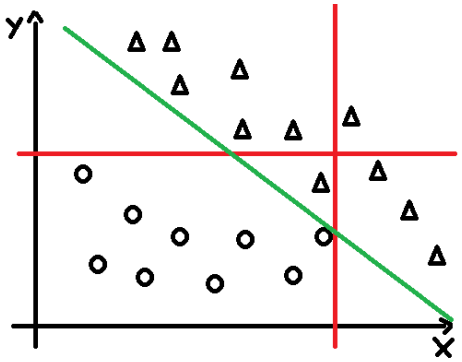


Figura 2.3: In questo caso sarebbe molto meglio utilizzare un unico taglio obliquo anziché due tagli paralleli ad una delle dimensioni in quanto permetterebbe di classificare correttamente tutte le istanze senza commettere errori

L'algoritmo Decision Tree è un algoritmo non parametrico, ovvero non fa alcuna assunzione sulla distribuzione dei valori all'interno del dominio. Trovare il migliore albero decisionale è un problema NP-Completo, infatti l'algoritmo non trova il migliore albero decisionale ma solo uno funzionante. Nel caso di questo algoritmo la complessità computazionale di questo algoritmo può essere approssimata con $O(h)$ con h altezza dell'albero. L'algoritmo risulta resistente al rumore, infatti è possibile potare l'albero fino a che il rumore diventa irrilevante. Inoltre l'algoritmo non soffre in presenza di attributi ridondanti nel dataset, dove per attributi ridondanti si intendono attributi linearmente dipendenti.

2.1.6 Scelta degli attributi e dei test

Cerchiamo ora di capire quale sia il test migliore da fare su un attributo dell'individuo da classificare.

Nel caso in cui il dominio dell'attributo scelto per la ramificazione sia discreto con V valori, il nodo associato a tale attributo genera V nodi figli.

Nel caso in cui il dominio sia continuo la situazione si diventa più complessa. Infatti, applicando lo schema precedente avremmo un nodo con infiniti figli (più precisamente ci sarebbe un nodo figlio per ognuno dei valori trovati nel dataset). In questo caso si avrebbero molti nodi ognuno dei quali contenente un piccolo insieme di valori la cui rilevanza dunque diverrebbe minima. Si può dunque procedere a discretizzare l'attributo in questione. Questa procedura prevede che l'intero dominio sia suddiviso in un numero finito di sottoinsiemi ognuno rappresentato da un valore specifico. Si procede poi ad assegnare ogni valore del dominio iniziale ad uno dei valori del nuovo dominio secondo una qualche tecnica specifica per il dominio. In generale la discretizzazione va effettuata tenendo conto delle caratteristiche specifiche del dominio e della distribuzione dei valori all'interno del dominio. Un caso estremo di discretizzazione è chiamato binarizzazione. In questo caso i valori del dominio vengono mappati su due unici valori a seconda che siano sopra o sotto una certa soglia predeterminata.

Definizione 2.7: Divisione basata su soglia

Sia $d \in D$ un attributo reale e continuo, $t \in d$ un valore di D e c una classe. Definiamo l'entropia di c rispetto a d con soglia t come

$$H(c|d:t) = H(c|d<t)P(d<t) + H(c|d \geq t)P(d \geq t)$$

Definiamo a questo punto il guadagno di informazione ottenuto scegliendo una soglia t come

$$IG(c|d:t) = H(c) - H(c|d:t)$$

Definiamo dunque il guadagno di informazione associato ad un attributo reale come

$$IG(c|d) = \max_t IG(c|d:t)$$

mentre la soglia $\hat{t}$ sarà scelta come

$$\hat{t} = \operatorname*{argmax}_t IG(c|d:t)$$

La complessità computazionale di $IG(c|d)$ diventa quindi

$$N \log N + 2NV_d$$

dove N è il numero di individui nel nodo preso in considerazione e V_d è il numero di valori distinti associati all'attributo d

Per effettuare la miglior scelta possibile sull'attributo da usare per effettuare la ramificazione siamo in cerca della massima purezza, la quale è indice di quanto un nodo fornisca contenuto informativo ai fini della classificazione. Ad esempio un nodo contenente gli elementi di due classi nella stessa proporzione ha un basso valore di purezza, mentre un nodo contenente gli elementi di una sola classe ha un elevato valore di purezza. Si possono prendere in considerazione dunque tre indicatori:

- Information Gain
- Indice Gini
- Misclassification Error

Definizione 2.8: Indice Gini

Si consideri un nodo p con classi C_p. Viene calcolata qual è la frequenza di una classificazione errata in una classe j dato un individuo da classificare casuale basato solamente sulla frequenza delle classi nel nodo corrente. Per la classe j si definiscono:

- La frequenza $f_{p,j}$
- La frequenza delle altre classi $1 - f_{p,j}$
- La probabilità di un assegnamento errato $f_{p,j} * (1 - f_{p,j})$

L'indice Gini è definito come:

$$\sum_j f_{p,j} * (1 - f_{p,j}) = \sum_j f_{p,j} - \sum_j f_{p,j}^2 = 1 - \sum_j f_{p,j}^2$$

Si ha un massimo nel caso in cui i valori siano uniformemente per ogni classe ed è $1 - \frac{1}{C_p}$. Si ha un minimo (ovvero 0) nel caso in cui tutti i valori appartengono alla medesima classe.
Si sceglie su quale valore effettuare la divisione sulla base del valore che consente di ottenere la massima riduzione di Gini

Definizione 2.9: Misclassification error

È un indicatore di purezza basato sul valore di accuratezza descritto sopra. Infatti, il Misclassification Error è il complemento a 1 del valore di accuratezza. La scelta del punto in cui effettuare la divisione è data da:

$$ME(p) = 1 - \max_j f_{p,j}$$

Confronto

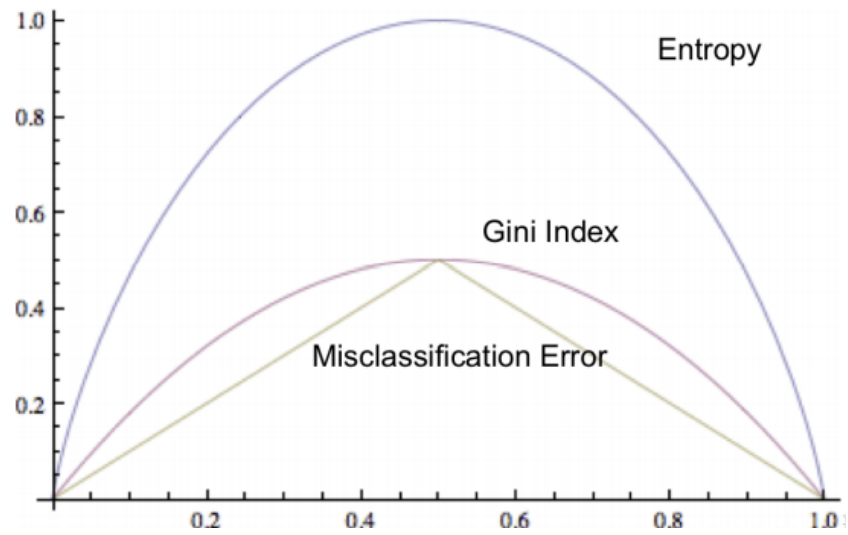


Figura 2.4

Dal grafico sopra notiamo che ME è lineare, per questo motivo un errore nella frequenza è trasferito completamente nel calcolo dell'impurità. L'entropia e il Gini index invece hanno una derivata non lineare, con un minimo al centro, per questo sono resistenti rispetto agli errori di frequenza, quando le frequenze delle due classi sono simili. Calcoliamo ora la complessità dell'algoritmo Decision Tree. Sono

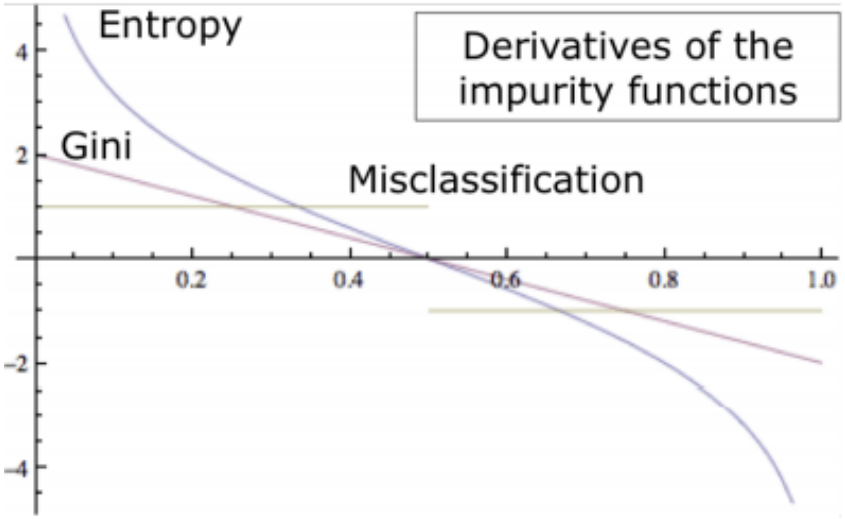


Figura 2.5

date N istanze e D attributi nel dataset. Ogni livello dell'albero richiede la considerazione dell'intero dataset. Ogni nodo Richiede la considerazione di tutti gli attributi. Quindi complessivamente il costo computazionale totale di un algoritmo Decision Tree è pari a:

$$O(DN \log N)$$

2.2 Selezione del modello: valutazione di un classificatore

Vediamo ora come valutare un classificatore in modo da capire quale dei modelli disponibili sia il migliore e quale tra gli algoritmi disponibili sia il migliore in termini della configurazione degli iper parametri.

Esempio 2.8

Si consideri un sistema usato per identificare le perdite di petrolio in mare o i furti di carburante tramite l'utilizzo di immagini satellitari, le quali però possono avere alcuni problemi come la presenza di nubi nel cielo (che verrebbero confuse con dei falsi positivi). Inoltre, dato che esistono pochi esempi di immagini in cui vi è realmente una macchia di petrolio in mare e quindi il numero di esempi positivi è molto minore del numero di esempi negativi, il dataset ottenuto è sbilanciato. Questo classificatore è binario (c'è stata una perdita oppure non c'è stata una perdita), per questo problema gli errori che si possono commettere possono essere solo due:

- Una immagine che presenta macchie di petrolio viene riconosciuta come priva di macchie: falso negativo
- Una immagine senza macchie di petrolio viene riconosciuta come con presenza di macchine: falso positivo

È evidente che è necessario una bilanciamento tra i falsi positivi ed i falsi negativi. Si potrebbe utilizzare un dataset supervisionato (con i vantaggi ed i problemi che comporta), tuttavia le performance che si otterrebbero sarebbero più che ottimistiche e dunque si rende necessario un limite inferiore per stimare l'errore e quanto la teoria sia in grado di spiegare i dati. Inoltre bisogna considerare che i dati supervisionati sono rari e che il dataset supervisionato deve essere diviso in tre parti, diminuendo quindi il numero di dati supervisionati contenuto in ciascuna parte.

Si nota come tale valutazione sia indipendente dall'algoritmo specifico usato per classificare i dati.

2.2.1 Stima dell'errore

Supponiamo che il test set sia una buona rappresentazione (mediamente) dell'intero dataset, ovvero che i dati siano presenti in proporzioni simili a quelle reali e che non vi siano casi non rappresentati. La relazione tra il training set e l'intero dataset sarà soggetta alla variabilità dovuta alla probabilità. La valutazione può essere fatta in due gradi di astrazione:

- Valutazione a livello generale: valuta a livello generale l'intero dataset
- Valutazione a livello locale: valuta un pezzo del modello come può essere un nodo del Decision Tree

Detto ciò è naturale che i risultati ottenuti a runtime (compreso l'errore calcolato) sia dotati di un certo margine di incertezza.

Nella stima di un errore è naturale pensare al processo di Bernoulli, infatti prevedere ogni elemento del test set è come un esperimento di un processo bernoulliano. Questo è equivalente a N esperimenti binari indipendenti dello stesso tipo, dove indichiamo con S il numero di previsioni corrette e con E il numero di previsioni errate. Possiamo misurare $e0\frac{E}{N}$ ovvero la frequenza empirica dell'errore ed $s = 1-e$ ovvero la frequenza empirica di successo. A questo punto ci si chiede quali siano i limiti di questi due valori, ovvero la probabilità di errore e la probabilità di successo.

In un processo binario la frequenza empirica ha una distribuzione normale attorno al vero valore della probabilità (per campioni di cardinalità sufficientemente elevata). Inoltre all'aumentare del numero di campioni aumenta anche l'aderenza del valore empirico sul valore reale[1] (vedi figura 2.6).

2.2.2 Potatura statistica dell'albero decisionale

Consideriamo la strategia di pruning statistico dell'albero decisionale per dell'algoritmo C4.5:

- Si considerano i sotto alberi vicini alle foglie

[1] Vedi pp. 10-13 pacco di slide classificazione II

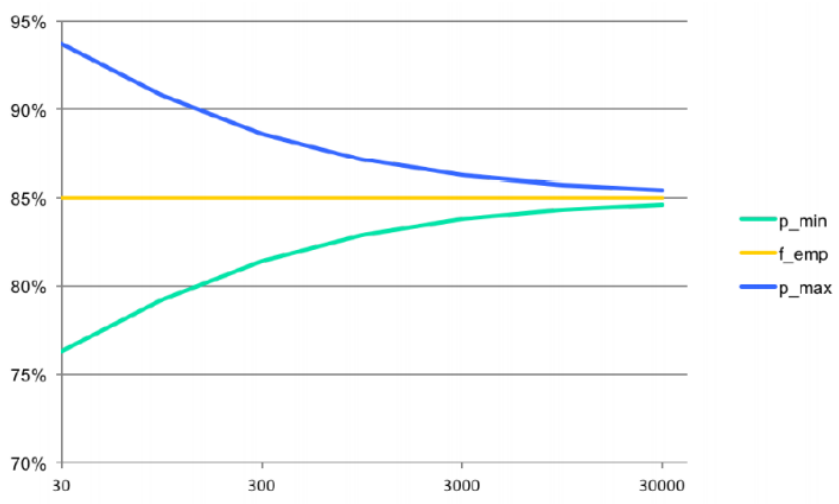


Figura 2.6: Per valori training set poco numerosi il margine di incertezza è ampio ed ha al centro la frequenza empirica, man mano che la dimensione del training set aumenta le curve p_{min} e p_{max} tendono asintoticamente a f_{emp}

- Si calcola il loro massimo errore e_{max_l} come una somma pesata tra il massimo errore delle foglie associate a quel sotto albero
- Si calcola il massimo errore e_{max_r} della radice del sotto albero considerato che si è trasformato in una foglia
- Si pota il ramo solo se $e_{max_r} \leq e_{max_l}$

Il razionale di questa operazione è il seguente: potando un ramo aumenta potenzialmente la frequenza dell'errore, ma aumenta anche il numero di oggetti all'interno del nodo e dunque l'errore massimo associato ad un nodo può diminuire.

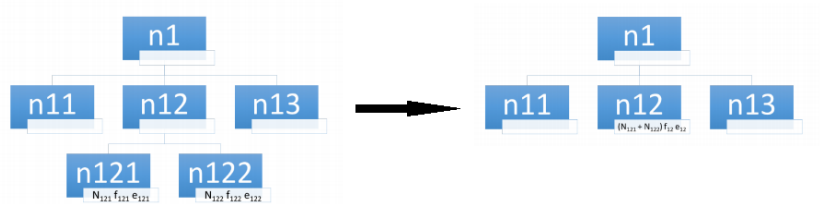


Figura 2.7: Albero prima e dopo la potatura

In figura 2.7 si vede prima e dopo la potatura. Questa è avvenuta perché

$$e_{max_r} \leq e_{max_l}$$
$$(N_{121} + N_{121}) * e_{12} \leq N_{12} * e_{121} + N_{122} * e_{122}$$

2.2.3 Strategie di test

La frequenza di errore è l'indicatore più semplice della qualità di un classificatore. È la somma di ogni classe divisa per il numero di classi testate. Da ora in poi sarà usata la frequenza di errore empirica per semplicità, ma è chiaro che è semplice passare alla frequenza di errore massimo che è più simile a un caso reale. Un classificatore può essere calcolato anche a partire dall'indice di accuratezza, questa è il reciproco della frequenza di errore $(1 - f_e)$. Questo indicatore è più sofisticato ed è utile per comparare le differenti classi o le diverse impostazioni degli iperparametri. Quando vengono stimate le performance a run-time bisogna tenere conto anche del costo degli errori.

È possibile usare varie strategie per ottenere una valutazione accurata (vedi figura 2.8)

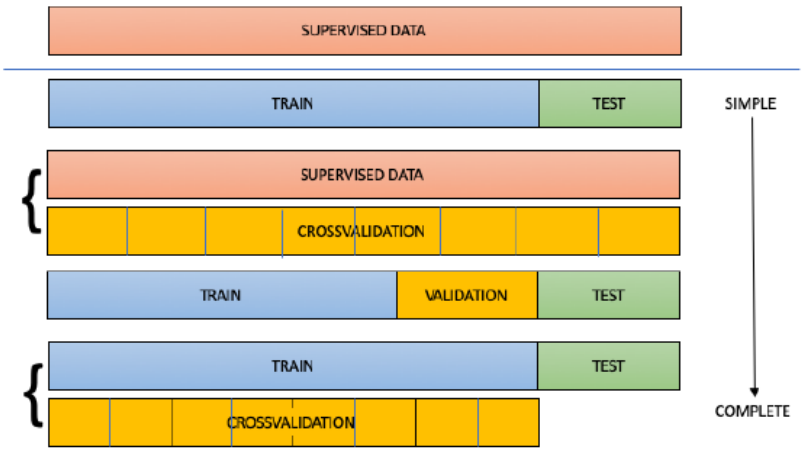


Figura 2.8

Holdout È un metodo standard che consiste nel dividere il dataset in un training set ed in un test set casualmente. Se il dataset è relativamente piccolo, potrebbe accadere che la proporzione delle classi nel dataset supervisionato è alterata nel training e nel test set, per prevenire questo caso si usa la tecnica derivata dalla statistica della **stratificazione**. Essa permette di avere train set e test set che riflettono le caratteristiche del dataset originale in termini di frequenza degli elementi.

Un'alternativa consiste nel dividere il dataset in: train, validation e test set ottenendo alcuni ulteriori vantaggi. In primo luogo questa tecnica è più veloce della cross validation (vedi dopo), inoltre l'ottimizzazione degli iperparametri è fatta sul validation set che è indipendente dal test set garantendo dunque un risultato finale più affidabile rispetto all'holdout semplice. Tuttavia questa tecnica è meno affidabile rispetto alla cross validation.

Cross Validation In questo caso il training set è partizionato in k subset, se necessario con il partizionamento stratificato. A questo punto vengono fatte k iterazioni usando uno dei subset come test e gli altri come training set. Infine, viene calcolato il risultato del test e generato il modello finale usando l'intero dataset.

In questo modo possiamo ottenere molti vantaggi dai dati supervisionati perché ogni record è usato $k-1$ volte per il training ed una volte per il test. Questo approccio ha diversi vantaggi:

- La stima delle performance è fatta su k iterazioni e l'errore finale è espresso come media sulle iterazioni permettendo di ottenere un risultato più preciso
- Tutti gli esempi sono usati una volta per il testing e diverse volte per il training
- Il modello finale è ottenuto usando tutti gli esempi, in questo modo si ha il miglior uso degli stessi

2.2.4 Misure delle prestazioni di un classificatore

Si consideri ora un esempio binario di previsione (saranno dette positivo e negativo le due classi da prevedere). Come è possibile osservare nella matrice di confusione (figura 2.9) il classificatore può predire:

- Veri positivi (TP): sono esempi positivi che il classificatore identifica come positivi
- Veri negativi (TN): sono esempio negativi che il classificatore identifica come negativi
- Falsi positivi (FP): sono esempi negativi che il classificatore identifica come positivi
- Falsi negativi (FN): sono esempi positivi che il classificatore identifica come negativi

È possibile quindi definire il success rate (ossia l'accuratezza) come:

$$s = \frac{TP + TN}{n_{test}}$$

		Predicted class	
		P	N
True class	P	TP	FN
	N	FP	TN

Figura 2.9

E l'error rate come il complemento a uno:

$$e = 1 - s$$

Oltre alla accuratezza esistono altri indicatori per valutare le prestazioni di un classificatore:

- Velocità: indica quanto il classificatore sia veloce a riconoscere un individuo, va valutato se tale velocità è sufficiente
- Robustezza al rumore
- Scalabilità al crescere del numero di individui da classificare
- Interpretabilità: significa in generale se è possibile capire perché un classificatore produce in output una determinata classificazione

Un errore di classificazione può avere diverse conseguenze a seconda del contesto di utilizzo. Infatti, se si stanno prevedendo malattie, ad esempio, un falso positivo può essere meno pericoloso di un falso negativo, se invece si vogliono fare previsioni se un veicolo militare è amico o nemico un falso positivo può avere conseguenze tragiche. Esistono diverse misure per valutare l'affidabilità di un classificatore:

- Precisione: il tasso di TP in relazione con le classificazioni positive

$$prec = \frac{TP}{TP + FP}$$

- Recall (sensibilità): il tasso di positivi che possono essere rilevati

$$rec = \frac{TP}{TP + FN}$$

- Specificità: il tasso di negativi che possono essere rilevati

$$spec = \frac{TN}{TN + FP}$$

- Accuratezza: la somma pesata della sensibilità e della specificità

$$acc = sens\frac{pos}{N} + spec\frac{neg}{N}$$

- F-measure (F1 o F-score): è la relazione armonica della precisione e della sensibilità. È importante perché esiste un bilanciamento che significa che una misura è ottimale se precisione e sensibilità sono correttamente bilanciate, se sono eguali si ottiene il miglior risultato

$$F1 = 2\frac{prec \cdot rec}{prec + rec}$$

Si analizza ora il caso multidimensionale (o per meglio dire multi-classe). Per valutare come funziona il classificatore vanno considerate le informazioni a priori, ciò significa che la valutazione di una previsione deve essere presa come base e si cerca, se possibile, di attuare miglioramenti. La matrice di confusione mostrata prima viene espansa nel seguente modo:

2.2.5 Il costo dell'errore

È stato prima spiegato come una previsione sbagliata corrisponda implicitamente ad un costo. Per questo motivo, a fianco della matrice di confusione, si può considerare anche la matrice del costo, per comparare due modelli e tenere in conto la differenza del costo dell'errore che uno ha rispetto ad un altro. Questa matrice ha valori 0 nella diagonale principale perché le posizioni della matrice che la formano sono i casi in cui il classificatore non sbaglia, nelle altre celle è indicato invece quanto una previsione sbagliata è più costosa rispetto ad un'altra.

		Pred class	
		P	N
True class	P	0	1
	N	5	0

Per questo motivo per inserire il costo dell'errore all'interno del processo di apprendimento possiamo usare due diversi approcci:

- Alterare la proporzione delle classi nei dati supervisionati duplicando gli esempi in cui il classificatore ha un errore più alto. In questo modo si forza il classificatore ad avere performance migliori nei casi più delicati
- Alcuni schemi di apprendimento consentono di assegnare pesi a determinate istanza, si da quindi un peso più alto alle istanze il cui errore di classificazione porta ad un costo dell'errore elevato

2.3 Valutazione dei classificatori probabilistici

Molti classificatori producono, anziché una valutazione diretta (valutazione *crisp*), una tupla di probabilità contenente un valore per ognuna delle classi possibili. Ciò consente di valutare l'affidabilità della misura e di vedere quali siano le possibili classificazioni alternative. Nel caso in cui il processo di classificazione includa varie fasi di valutazione questo tipo di classificatori risulta particolarmente in quanto fornisce via via tutti i dati necessari. Questo tipo di classificatori inoltre è particolarmente utile sia per classificare gli individui di un certo dataset sia per creare cluster di individui con caratteristiche simili. Esistono diversi strumenti per valutare i classificatori probabilistici.

I classificatori crisp possono nascondere le vere probabilità. Ad esempio, nel caso in cui un nodo fogli di un albero di classificazione non contenga elementi di una sola classe è possibile assegnare ad ognuna delle classi contenute nel nodo una certa probabilità. Tuttavia spesso capita che nei nodi foglia vi sia un piccolissimo numero di elementi con frequenza (e probabilità quindi) vicina allo zero, in questo caso è possibile usare tecniche di smoothing per aggiustare il valore delle probabilità. Infine occorre dire che è possibile passare da una valutazione probabilistica ad una crisp mediante l'uso di varie tecniche (a seconda del numero di classi) che in generale selezionano la classe avente la probabilità maggiore.

2.3.1 Lift Chart

Si consideri il caso binario, ovvero in cui un classificatore debba associare ad ogni individuo della popolazione una di due classi con la relativa probabilità.

Consideriamo un sample di individui da classificare. Ad ogni elemento sarà associata un probabilità di appartenere alla classe positiva ed una probabilità di appartenere alla classe negativa. Possiamo dunque ordinare l'intero sample sulla base della prima probabilità e creare un grafico (figura 2.10) in cui sulle ascisse vi sia la dimensione del campione (in termini percentuali) e sulle ordinate il numero di positivi trovati nella frazione di sample considerata. La linea del grafico conta cumulativamente quali degli individui etichettati come positivi sono realmente positivi.

La retta diagonale è il numero di positivi che sono ottenuti da un classificatore casuale, ovvero un classificatore che assegni le due classi con la medesima probabilità indipendentemente dall'individuo in esame. Con questo classificatore in media si classificherà correttamente sempre la metà degli individui. Se la curva del classificatore che si sta valutando scende sotto quella del classificatore casuale allora il classificatore in esame è di pessima qualità, al contrario più si trova al di sopra quella del classificatore casuale maggiore è la sua qualità. Per questo motivo maggiore è l'area tra le due curve migliore è il classificatore.

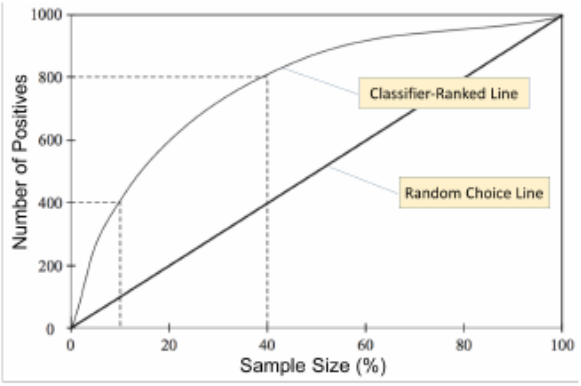


Figura 2.10

2.3.2 Curva ROC (Receive operator characteristic)

È uno strumento introdotto durante la seconda guerra mondiale per valutare se un segnale radio fosse nemico o causato da semplice rumore e per trovare un compromesso tra falsi allarmi e segnali interpretati correttamente. Il rumore altera il riconoscimento del segnale secondo una distribuzione gaussiana. Il problema può essere così riformulato: trovare la miglior soglia (vedi figura 2.11) tra le due curve per massimizzare il compromesso di cui prima. In questo grafico si hanno quattro aree:

- TN = blu e ciano = probabilità che un negativo sia riconosciuto correttamente
- FN = ciano = probabilità di riconoscere erroneamente un negativo
- FP = rosa e viola = probabilità di riconoscere erroneamente positivo
- TP = rosso e viola = probabilità di riconoscere correttamente un positivo

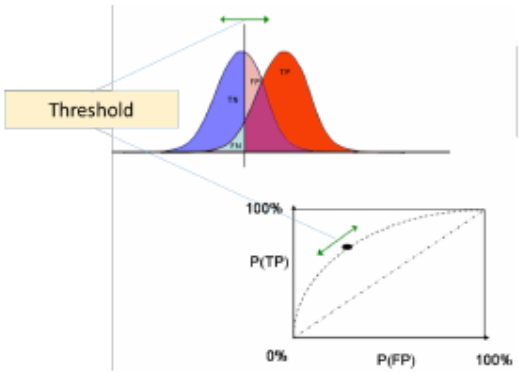


Figura 2.11

La soglia permette di aumentare o diminuire il tasso di riconoscimento dei segnali, spostandola verso destra incrementa sia il tasso dei falsi positivi che quello dei falsi positivi. Con un valore minore di rumore le due curve sono meglio separate. L'area tra le due curve (nel secondo grafico sotto) rappresenta la bontà della soglia scelta, avendo come massimo il triangolo sinistro del grafico.

Questo classificatore può essere convertito in un classificatore crisp scegliendo una soglia di probabilità oltre la quale la predizione è considerata come positiva.

Dato un classificatore probabilistico si può costruire la curva ROC in questo modo: per prima cosa si ordinano gli elementi di test in ordine di probabilità di essere positivi decrescente. Si imposta una soglia come la probabilità massima, si inizializzano TP e FP a zero. Fatto ciò, è necessario ripetere le azioni seguenti: aggiornare il numero di TP e FP con probabilità dalla soglia a 1, disegnare i punti sulla curva, muovere fino al prossimo valore di probabilità di positivi.

Abbassando la soglia aumenta sia il valore di TP che di FP, si hanno buone performance nel momento in

cui il tasso di TP aumenta più velocemente di quello di FP. Inoltre si nota che in questo caso aumenta il valore dell'indicatore recall e diminuisce quello di precision.

2.4 Classificatori multi-classe

Molti classificatori generano una classificazione binaria. Nel caso in cui vi siano più di due classi tuttavia è necessario adottare alcune tecniche specifiche. Per essere in grado di distinguere fra due o più situazioni è necessario eseguire la classificazione in più passi. Esistono diversi metodi per gestire la classificazione multi-classe:

- Modificare l'algoritmo di training e il modello
- Usare un insieme di classificatori binari e combinare il risultato: è una alternativa interessante perché possiamo utilizzare lo stesso classificatore in più passaggi. Quindi si hanno un numero maggiore di problemi da risolvere ma questi sono relativamente più semplici. Esistono due strategie:
  - One-vs-One (OVO). In questo caso vengono considerate tutte le possibili coppie di classi ($c(\frac{C-1}{2})$ nel caso vi siano C classi) e viene generato un classificatore binario per ogni coppia. Nel momento in cui è necessario effettuare la previsione si applica un modello a votazione. Ogni nuovo individuo viene valutato da tutti i classificatori binari. Ogni classe vincente riceve un +1 come punteggio. Alla fine della votazione la classe selezionata è quella con punteggio più elevato
  - One-vs-All (OVA o OVR). In questo caso si considerano C problemi binari dove la classe c è un esempio positivo e tutti gli altri sono esempi negativi. Quindi si ricostruisce il training set cambiando l'etichetta (c o $\neg c$). A questo punto si esegue il training di un classificatore binario e si ripete il processo per ogni classe, quindi si avranno C classificatori binari ognuno allenato a distinguere se un individuo appartiene o no a quella classe. Per effettuare la classificazione si applica uno schema a votazione. Ogni nuovo individuo viene sottoposto al giudizio di tutti i classificatori ottenendo un punteggio in termini di probabilità. La classe con la probabilità maggiore vince la votazione

  La strategia OVO deve risolvere molti più problemi, sebbene essi siano intrinsecamente più semplici, mentre la strategia OVA tende ad essere non bilanciata

2.5 Metodi Ensemble

I metodi Ensemble si basano su principi statistici. L'idea di base è di allenare un insieme di classificatori base e ottenere la predizione finale considerando i voti dei classificatori base. Ne consegue che il metodo Ensemble funziona meglio e la qualità del risultato è migliore rispetto a un classificatore unico. I metodi Ensemble sono utile solamente se i classificatori base sono indipendenti e se le performance dei classificatori base sono migliori di quelle del classificatore random.

Considerando un certo numero N di classificatori si può vedere per quale motivo, nel caso in cui essi siano indipendenti, l'errore complessivo dia minore dell'errore dei singoli classificatori. Per avere una previsione positiva occorrono almeno $k = \lfloor \frac{N}{2} \rfloor + 1$ votazioni positive.

$$e_{ensamble} = \sum_{i=k}^{N} \binom{N}{i} e^i (1-e)^{N-i}$$

2.5.1 Metodi per i classificatori ensemble

Esistono diversi metodi per i classificatori ensemble:

- Per manipolazione del training set: usa differenti subset dei dati e allena i classificatori su dati differenti. Questi classificatori sono estratti con una strategia di sampling:
  - Bagging: si ottengono training subset tutti della stessa dimensione a partire dal trading set di partenza. Per avere tutti la stessa dimensione alcuni individui vengono campionati più di una volta, può anche capitare che alcuni individui non vengano mai campionati

  - Boosting: viene eseguita una prima manche di addestramento. Viene associato un peso ad ogni istanza di addestramento. A questo punto una viene eseguito di nuovo l'addestramento dando un peso maggiore agli esempi falliti nella esecuzione precedente. Se un istanza fallisce viene aumentato il peso per renderla più rilevante alla prossima iterazione
  - Adaboost: è un esempio di Boosting, l'importanza di ogni classificatore base varia in relazione all'error rate, i voti dei classificatori più efficaci hanno maggiore importanza

- Per manipolazione delle caratteristiche di input: invece di campionare si possono generare classificatori con un sottoinsieme delle caratteristiche di input che possono essere scelte casualmente o con l'aiuto di un esperto del settore

2.6 Classificatore Naive Bayes

Il classificatore naive Bayes è un classificatore probabilistico, basato in particolare sul teorema di Bayes. Questo tipo di classificatore considera contemporaneamente il contributo di tutti gli attributi, al contrario dei DT che ne considerano uno per volta.

Per utilizzare questi classificatori è necessario che sia verificata un'assunzione particolarmente forte e vincolante, ovvero l'indipendenza statistica dei vari attributi. In generale essa non è mai verificata completamente, ma all'atto pratico se il grado di indipendenza degli attributi è sufficiente (anche se non totale) il metodo funziona lo stesso.

Esempio 2.9

Si consideri il dataset Weather/Play.

Aspetto			Temperatura			Umidità			Ventoso			Giocare	
	Sì	No		Sì	No		Sì	No		Sì	No	Sì	No
Assolato	2	3	Caldo	2	2	Alta	3	4	Vero	6	2	9	5
Nuvoloso	4	0	Tiepido	4	2	Norm.	6	1	Falso	3	3		
Piovoso	3	2	Freddo	3	1								

Vogliamo prevedere la probabilità di uscire a giocare sapendo che il tempo è assolato e ventoso, la temperatura fredda e l'umidità alta. Queste sono tutte informazioni indipendenti, il risultato complessivo è dato da:

$$Y = \frac{2}{9} \times \frac{3}{9} \times \frac{3}{9} \times \frac{3}{9} \times \frac{9}{14} = 0.0053$$
$$N = \frac{3}{5} \times \frac{1}{5} \times \frac{4}{5} \times \frac{3}{5} \times \frac{5}{14} = 0.0206$$

Valori che normalizzati ad uno corrispondono a:

$$P(Y) = \frac{0.0053}{0.0053 + 0.0206} = 20.5\% \quad P(N) = \frac{0.0206}{0.0053 + 0.0206} = 79.5\%$$

Si vede come il no sia molto più probabile del sì, quindi questa situazione verrà classificata come no.

2.6.1 Il metodo

Il classificatore è basato sul teorema di Bayes. Questo teorema viene normalmente impiegato per calcolare la probabilità di una causa che ha scatenato l'evento verificato. Per esempio, si può calcolare la probabilità che una certa persona soffra della malattia per cui ha eseguito il test diagnostico (nel caso in cui questo sia risultato negativo) o viceversa non sia affetta da tale malattia (nel caso in cui il test sia risultato positivo), conoscendo la frequenza con cui si presenta la malattia e la percentuale di efficacia del test diagnostico. Formalmente il teorema di Bayes è valido in tutte le interpretazioni della probabilità. In ogni caso, l'importanza di questo teorema per laăstatisticaăè tale che la divisione tra le due scuole (statistica bayesianaăeăstatistica frequentista) nasce dall'interpretazione che si dà al teorema stesso. Data un ipotesi

H e una prova E:

$$P(H|E) = \frac{P(E|H)P(H)}{P(E)}$$

Nel nostro caso l'ipotesi è la classe c, mentre la prova è una tupla di valori dell'elemento da classificare. Con la formula sopra si può dividere la prova, una per attributo, e, se gli attributi sono indipendenti allinterno della stessa classe:

$$P(c|E) = \frac{P(E_1|c)P(E_2|c)P(E_3|c)P(E_4|c)P(c)}{P(E)}$$

Riscrivere $P(c|E)$ in questo modo è utile perché così diventa possibile calcolare la probabilità di ogni valore di un attributo anziché la probabilità dell'intera tupla.

Il metodo Naive Bayes è semplice:

1. Si calcolano le probabilità condizionate dagli esempi
2. Si applica il teorema
3. Il denominatore è lo stesso per ogni classe, sarà eliminato con la normalizzazione
4. Si sceglie la classe che ha la probabilità più alta

2.6.2 Criticità

Questa tecnica è chiamata naive poiché assumere che gli attributi sian indipendenti è abbastanza semplicistico.

Attributi mancanti e smoothing Nei dataset reali è comune che ci siano valori mancanti. Occorre quindi interrogarsi su cosa succede se il valore v dell'attributo d non viene mai visualizzato negli elementi della classe c. In questo caso $P(d = v|c) = 0$. Nella pratica questo caso è abbastanza comune, in particolare in un dominio con molti attributi e molti valori distinti. È evidentemente necessaria una soluzione alternativa. Esiste una tecnica chiamata smoothing, questa tecnica può mitigare anche l'effetto di overfitting (è un iperparametro dei classificatori naive Bayes).

Definizione 2.10

Si introducono:

- α, coefficiente di smoothing
- $af_{d=v_i,c}$, frequenza assoluta del valore v_i nell'attributo d per la classe c
- V, numero di valori distinti nell'attributo d sull'intero dataset
- af_c, frequenza assoluta della classe c

Otteniamo la frequenza smooth (levigata) come:

$$sf_{d=v_i,c} = \frac{af_{d=v_i,c} + \alpha}{af_c + \alpha V}$$

Con $\alpha = 0$ si ottiene la frequenza relativa classica

Valori numerici Se il campione prevede valori numerici il metodo basto sulla frequenza è inapplicabile. In questo caso si applica un ulteriore assunto, ovvero che i dati seguano una distribuzione gaussiana. Al posto delle frequenze frazionarie possiamo calcolare in questo modo media e varianza per ogni attributo numerico relativamente ad ogni classe.

Esempio 2.10

Riprendendo l'esempio precedente, se le temperature fossero espresse come valori numerici sarebbe possibile calcolare media e deviazione standard delle temperature per le quali si è usciti a giocare.

Per ottenere la densità di probabilità associata ad un certo valore si può usare:

$$f(temp = v|c_i) = \frac{1}{\sqrt{2\pi}\sigma_{c_i}} e^{-\frac{(v-\mu_{c_i})^2}{2\sigma_{c_i}^2}}$$

Probabilità e densità di probabilità sono concetti strettamente legati ma sono comunque differenti. In un dominio continuo, la probabilità che un valore assuma esattamente un determinato valore reale è zero, tuttavia può essere calcolato il valore della densità di probabilità associata a quel valore, ovvero la probabilità che il valore in un intorno del valore considerato. Il valore considerato è ovviamente arrotondato a con un fattore di precisione. Questo fattore di precisione è lo stesso per tutte le classi, per questo motivo può non essere preso in considerazione. Se un valore numerico è mancante, la media e la deviazione standard sono basate solo sui valori presenti.

Esempio 2.11

Riprendendo l'esempio precedente e supponendo che temperatura ed umidità siano espresse tramite valori numerici si ha:

$$Y = \frac{2}{9} \times 0.0340 \times 0.0221 \times \frac{3}{9} \times \frac{9}{14} = 0.000036$$
$$N = \frac{3}{5} \times 0.0221 \times 0.0381 \times \frac{3}{5} \times \frac{5}{14} = 0.000108$$

Valori che normalizzati ad uno corrispondono a:

$$P(Y) == 25\% \quad P(N) = 75\%$$

Si vede come il no sia molto più probabile del sì, quindi questa situazione verrà classificata come no.

2.7 Classificazione lineare tramite percettroni

Un altro strumento per la classificazione sono i percettroni (perceptron, o anche neuroni artificiali), i quali consentono di effettuare classificazioni lineari. In pratica è una combinazione lineare di input pesati (w_0 consente di ottenere un iperpiano non passante per l'origine) che genera un iperpiano in grado di dividere (si spera al meglio) gli elementi appartenenti a due classi (chiamate positiva e negativa).

$$f(x) = w_0 + \sum_{i=1}^{N} w_i x$$

È possibile usare questa tecnica in presenza di attributi numerici. Nel caso $f(x_i) > 0$ l'elemento i-esimo appartiene alla classe positiva, appartiene alla classe negativa in caso contrario. Si deve inoltre scegliere una strategia da applicare nel caso in cui il valore restituito sia nullo, come ad esempio "predict negative".

Apprendimento Per la costruzione di un riconoscitore come descritto è evidente siano importantissimi i pesi che descrivono l'iperpiano. Per l'apprendimento dell'iperpiano si può usare il seguente algoritmo:

Algoritmo 2.1

```
set all weights to zero
while there are example incorrectly classified
do
    for each training instance e
    do
        if e is incorreclty classfied
        then
            if class of e is positive
```

```
                then
                    add e to weight vector
                else
                    sub e from weight vector
                fi
            fi
        done
    done
```

Convergenza Ogni modifica ai pesi sposta l'iperpiano attraverso le istanze non correttamente classificate verso la classificazione corretta. Nel caso di un'istanza positiva ad esempio si avrà:

$$(w_0 + e_0)e_0 + (w_1 + e_1)e_1 + \cdots + (w_N + e_N)e_N$$

che aggiunge rispetto all'iterazione precedente un valore pari a:

$$e_0^2 + \cdots + e_n^2$$

che rende il valore complessivo meno negativo rispetto al valore precedente.

Le correzioni sono incrementali e quindi possono interferire con modifiche precedenti. L'algoritmo pertanto converge solamente se il dataset è linearmente separabile (in caso contrario termina raggiunto un limite di iterazioni).

2.8 Support Vector Machines

I percettroni appena descritti funzionano solamente nel caso in cui i dati siano linearmente separabili. Nel caso in cui non lo siano è possibile utilizzare un una support vector machine. Essa rinuncia all'ipotesi della linearità per ottenere degli ipersuperfici più complesse di un iperpiano come ad esempio:

$$w_1 e_1^3 + w_2 e_1^2 e_2 + w_3 e_1 e_2^2 + w_4 e 2^3$$

In questo modo si hanno più gradi di libertà, ad esempio gli esponenti usati. Tuttavia questo metodo diventa impraticabile già per numeri relativamente bassi di variabili (ad esempio limitandosi al quinto grado con dieci variabili sarebbero necessari circa 2000 coefficienti). Inoltre questo metodo è particolarmente soggetto all'overfitting nel caso in cui il numero di parametri sia comparabile con il numero di esempi forniti.

Ci sono diversi approcci per superare questo limite:

- Teoria dell'apprendimento computazionale (un metodo specifico per imparare dagli esempi)
- Nuove funzioni efficienti per i domini non linearmente separabili che fanno uso delle funzioni kernel, sono famiglie di funzioni che possono essere usate per adattare la superficie decisionale a una computazionalmente trattabile
- Puntare sull'ottimizzazione anziché su una ricerca greedy: l'albero decisionale usa una ricerca di tipo greedy perché ad ogni livello decide quale sia il nodo migliore per effettuare la separazione del campione, in questo modo però viene scelto il miglior nodo non la migliore catena di nodi
- Apprendimento statistico: la ricerca di una funzione di predizione è modellata come una funzione che stima il problema

2.8.1 Il margine dell'iperpiano

Si consideri un iperpiano che separa perfettamente gli elementi di un dataset i cui valori appartengono a due classi distinte. Si nota che, nella maggior parte dei casi non esiste un solo iperpiano che separa perfettamente le due classi, è più corretto dire che esiste un fascio di iperpiani che soddisfa questa condizione. Si indica con margine dell'iperpiano la distanza minima tra uno dei valori e l'iperpiano stesso. Viene chiamato iperpiano con margine massimo quell'iperpiano, facente parte del fascio di piani che separa perfettamente le due classi, che massimizza il valore del margine.

Per calcolare l'iperpiano con margine massimo si considera l'inviluppo complesso di un insieme di punti. Se un dataset è linearmente separabile gli inviluppi complessi delle classi non si intersecano. L'iperpiano con margine massimo è quello più lontano possibile da entrambi gli inviluppi. In generale un sottoinsieme di punti è sufficiente per definire l'inviluppo e di conseguenza l'iperpiano con margine massimo. Dopo l'elaborazione di tutti i dati di apprendimento si otterranno diversi elementi (Support Vector) che sono sufficienti per trovare la superficie cercata. In un certo senso i support vector consentono di ridurre le rappresentazioni del training set al minimo indispensabile per la classificazione. Trovare i support vector e l'iperpiano con margine massimo sono problemi che è possibile risolvere in maniera efficiente.

Evidentemente è molto comune che vi sia la possibilità che non esista un iperpiano perché gli elementi di diversa classe sono molto mescolati tra di loro. In questo caso si può introdurre il concetto di soft margin, un'approssimazione in grado di separare il maggior numero di elementi, ma non tutti. Per fare questo si imposta un numero di eccezioni (ovvero di elementi diversi a quelli raggruppati) che si è disposti ad accettare.

2.8.2 Confini non lineari

Esistono casi in cui i metodi elencati sopra non sono efficaci. Per gestire questo tipo di situazioni l'idea è quella di trasformare lo spazio di input in un nuovo spazio (chiamato spazio delle caratteristiche) ottenuto trasformando lo spazio di input in modo da mantenere le caratteristiche dei dati ma facendo sì che esista un iperpiano in grado di separare le classi.

La procedura per la separazione degli iperpiani richiede una serie di prodotti scalari tra i vettori. Per definire il mapping tra i due spazi è necessario usare le funzioni kernel, in questo modo tutta l'elaborazione avviene nello spazio dell'input evitando di eseguire vari calcoli complessi. Con questo stratagemma si limita l'analisi ad un insieme di funzioni e tipicamente si parte dalle funzioni più semplici e poi si prova con quelle che hanno una maggiore complessità. Ovviamente, la support vector machine ha un costo computazionale non indifferente perché il la complessità temporale è influenzata dal livello di ottimizzazione della libreria che lo implementa. Per esempio, per la libreria più popolare chiamata libSVM la complessità varia da $O(D*N^2)$ a $O(D*N^3)$. È evidente come la complessità sia approssimabile ad una cubica ma influenza anche il valore D, che è il numero di colonne del dataset, spesso $D < N$ ma non è sempre vero, si possono avere dataset con un numero elevato di attributi ma pochi esempi.

Vi è un ultimo problema: questi metodi sono tutti per un tipo di classificazione binaria. Se è necessario distinguere fra più di due classi bisogna usare OVO o OVA, che significa che bisogna fare il training di più di un classificatore.

2.9 Reti neurali

Un'altra tecnica utilizzabile per superare il problema della separabilità lineare sono le reti neurali. Esse superano il problema mediante l'utilizzo di funzioni di valutazione non lineari, questo fornisce anche un secondo vantaggio, ovvero la resistenza al rumore. Se la funzione fosse lineare il rumore sarebbe completamente trasferito all'output, mediante funzioni non lineari invece il rumore può essere ridotto (auspicabilmente) o incrementato.

Ispirate alle connessioni tra i neuroni degli animali, sono costituite da un insieme di elementi, simili a perceptron, organizzati un una struttura gerarchica a strati. Ogni neurone è un elaboratore di segnale dotato di una funzione di soglia (generalmente non lineare, continua e differenziabile in termini di se stessa e limitata sia superiormente che inferiormente), vari input e un singolo output. Ogni neurone quindi effettua una somma pesata degli input (come un percettrone), sottopone il risultato alla funzione di soglia e restituisce come risultato questa elaborazione. Gli output di un neurone vengono poi usati come input da altri neuroni.

Durante l'apprendimento i pesi delle connessioni tra i neuroni possono cambiare in modo da modellare sempre meglio i dati in ingresso.

Funzioni di soglia È possibile usare varie funzioni, ma, come detto prima, si preferiscono quelle che sono continue, differenziabili in termini di se stesse e limitate. Sono esempi l'arcotangente e la sigmoide:

$$sgm(x) = \frac{1}{1+e^{-x}} \qquad \frac{d\ sgm(x)}{dx} = \frac{-e^{-x}}{(1+e^{-x})^2} = sgm(x) - sgm^2(x)$$

2.9.1 Regressione Multi-Livello

Consideriamo ora la struttura della rete:

- L'input arriva in uno strato di input che ha un nodo per ogni dimensione del training set
- L'input layer comunica (in maniera pesata) con l'hidden layer (il cui numero di nodi è un parametro associato alla costruzione della rete). Ognuno dei nodi nascosti calcola il proprio valore come:

$$v_j = g(\sum_{i=0}^{D_{in}} w_{j,i} e_i) \quad j \in [1, D_h]$$

- L'hidden layer comunica (in maniera pesata) con l'output layer. Il numero di nodi nell'output layer è in relazione con il numero di classi del dominio. Per ognuno di nodi di output viene applicato un calcolo simile a quello usato per l'hidden layer

La dimensione degli strati di input e di output derivano direttamente dalla struttura del dataset. Il numero di nodi di input è il numero di features (+1, il bias) e il numero di output è determinato dal numero di classi distinte. $g(.)$ è la funzione di trasferimento.

Il concetto di Feed-Forward consiste nel non avere rami in retroazione. Questo fa sì che il segnale scorra dall'input all'output, non vi è connessione intra-layer, ogni nodo di uno strato comunica solo con nodi dello strato successivo.

Il caso più semplice è costituito da una rete che ha un solo nodo per ogni hidden layer, tanti nodi quante le feature ed un solo nodi di output.

2.9.2 Training di una Rete Neurale

Ogni rete neurale ha necessità di essere allenata. L'algoritmo di base è così composto:

```
set alla weights to random values
while termination condition is not met
do
    for each training instance e
    do
        feed the network with e
        compute the output nn(e)
        compute the weight correction for nn(e) - e_out
        propagate back corrections
    done
done
```

In analogia con quello che avviene con l'apprendimento biologico, gli esempi devono essere riproposti più volte per migliorare la precisione della rete stessa. La convergenza non è garantita.

Per ogni nodo è possibile calcolare l'errore rispetto all'output desiderato come:

$$E(w) = \frac{1}{2}(y - g(w, x))^2$$

dove x è il vettore di input del nodo, y l'output desiderato e w i pesi in input. Questa funzione potrebbe essere sia convessa che non esserlo. Nel caso in cui sia convessa diviene possibile trovarne il minimo assoluto, ovvero quella configurazione di pesi tale da rendere l'errore minimo. Nel caso la funzione non sia convessa diviene possibile solamente trovare un minimo locale.

Per effettuare questa operazione si procede a sottrarre ad ogni peso la sua derivata parziale moltiplicata per il *learning rate* (una costante che influenza la velocità di apprendimento, occorre tarala correttamente per trovare il miglior compromesso tra velocità e precisione).

$$w_{i,j} = w_{i,j} - \lambda \frac{\partial E(w)}{\partial w_{i,j}}$$

Otteniamo così una nuova versione, migliorata, dell'algoritmo precedente:

```
set alla weights to random values
while termination condition is not met
do
    for each training instance e
    do
        feed the network with e and compute the output nn(e)
        compute the error prediction at output layer nn(e) - e_out
        compute derivatives and weight corrections for output layer
        compute derivatives and weight corrections for hidden layer
    done
done
```

Ogni iterazione del ciclo while è chiamata epoca, in generale servono varie epoche prima di ottenere un risultato accettabile, ognuna delle quali inizia avendo un valore dei pesi differente (quello dettato dall'epoca precedente, o dalle condizioni iniziali).

Scelte progettuali Sono possibili vari modi di apprendimento, tra cui:

- Stocastico: ad ogni propagazione in avanti corrisponde un aggiornamento dei pesi. QUesto introduce del rumore in quanto l'aggiornamento viene effettuato sulla base di un singolo esempio, tuttavia allo stesso tempo riduce le probabilità di rimanere bloccati in un minimo locale. Inoltre dato che gli aggiornamenti sono continui questa tecnica è di valore per l'apprendimento online
- A batch: occorrono molte propagazioni per avere un aggiornamento dei pesi, in modo da calcolare un valore dell'errore più preciso. Generalmente consente di raggiungere il minimo locale in maniera più veloce e stabile

È inoltre possibile selezionare il numero di strati nascosti ed il valore della costante λ (anche durante l'allenamento, magari valori maggiori durante le prime epoche e poi via via sempre minori in modo da migliorare la precisione).

Ovviamente, a dispetto della progettazione, rimane il rischio di avere minimi locali o di overfitting, se la rete è troppo complessa rispetto alla complessità del problema decisionale.

Regolarizzazione Per regolarizzazione si intende una tecnica usata in molte funzioni di machine learning per incrementare le capacità di generalizzazione di un modello. Modifica le funzioni di performance, che di solito sono scelte come la somma dei quadrati degli error rate sul training set. In sostanza:

- Il miglioramento delle prestazioni si ottiene riducendo una funzione di perdita (ad esempio la somma degli errori quadrati)
- La regolarizzazione corregge tale funzione per rendere più uniforme l'adattamento ai dati
- Il livello regolarizzazione deve essere regolato

Error rate In generale il tasso d'errore diminuisce all'aumentare del numero di nodi presenti nella rete e del numero di epoche usato per allenarla, in particolare all'aumentare del rapporto segnale/rumore.

2.10 Classificatore K-Nearest Neighbors

Nei casi precedenti si è costruito un modello del classificatore. Nel caso del classificatore K-Nearest Neighbours il modello è l'intero dataset. Con questo classificatore si opera confrontando nuovi oggetti con quelli già presenti nel dataset cercando di trovare i K più vicini (o simili secondo un certo criterio). Calcolate le distanze si ordinano e si mantengono le K istanze del dataset più vicine. Si contano le classi di queste istanze e si predicono le etichette dell'oggetto in esame per maggioranza, si da come assunzione che l'oggetto sia simile ai vicini più stretti. È immediato l'emergere di pro e contro:

- Pro: mantenendo l'intero training set non si perde alcuna informazione, l'intero dataset rimane disponibile nella sua interezza
- Contro: non è una tecnica efficiente perché mantenere tutta questa mole di dati per ogni classificazione non è semplice

I parametri principali sono:

- Il numero di vicini da controllare (K): non ha senso, infatti, mantenere un valore di K eccessivo perché se la lista di vicini è larga si rischia che all'atto pratico quelli più distanti siano profondamente diversi dall'individuo che si vuole classificare.
- La metrica usata per calcolare la distanza: è raccomandato usare la distanza di Mahalanobis perché tiene conto della distribuzione dei dati

Capitolo 3

Clustering

Consideriamo ora alcune tecniche di apprendimento non supervisionato. Con il clustering si parte dai dati grezzi, senza sapere cosa questi rappresentino. Nel seguito si considereranno solamente cluster disgiunti perché si è interessati a considerare il caso in cui l'unione dei cluster non copre l'intero database, in modo da mantenere il rumore al di fuori dei cluster stessi, dove con rumore si intendono dati non desiderati che hanno errori intrinsechi o letture sbagliate.

Il clustering può essere riassunto nel suo complesso come segue. Dati N oggetti, ognuno descritto da D valori (dove D è il numero delle dimensioni dello spazio che contiene i dati), lo scopo è quello di suddividere lo spazio in K regioni che raggruppino gli oggetti secondo un qualche criterio di somiglianza possibilmente riuscendo ad individuare anche i dati che causano rumore. Come risultato si ottiene uno *schema di clustering*,ovvero una funzione che mappa ogni oggetto nella sequenza $[1, \ldots, k]$ (oppure nell'insieme degli oggetti rumore). Nell'effettuare questa operazione si desidera che gli oggetti facenti parte del medesimo cluster siano il più simile (intuitivamente con simile si intende vicini in senso euclideo) possibile ed allo stesso tempo che gli oggetti contenuti in cluster differenti siano il più diversi possibile (vedi paragrafo 1.4).

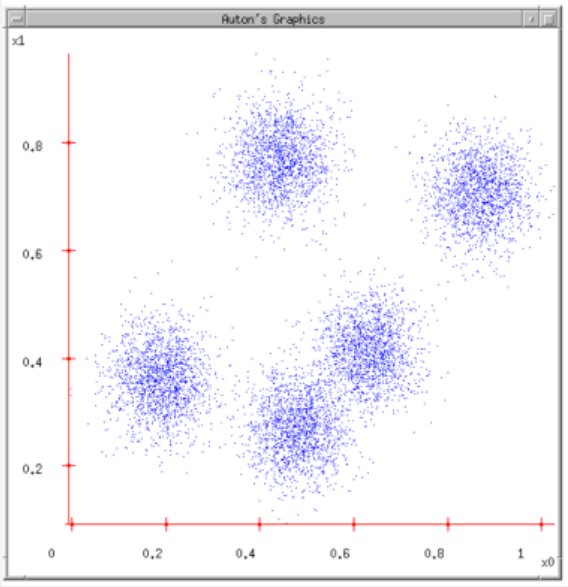
Esistono diverse tipologie di algoritmi di clustering:

- A partizioni: prevedono semplicemente di assegnare un'etichetta ad un oggetto, computare qualcosa e cambiare, se necessario, gli assegnamenti per migliorarli. Esempi sono K-Means, Expectation Maximization, Clarens
- Gerarchici: prevedono di effettuare una computazione ad un certo livello e poi di specializzarla al livello inferiore
- Basati su collegamenti
- Basati sulla densità. Ad esempio DBSCAN
- Basa sulla statistica

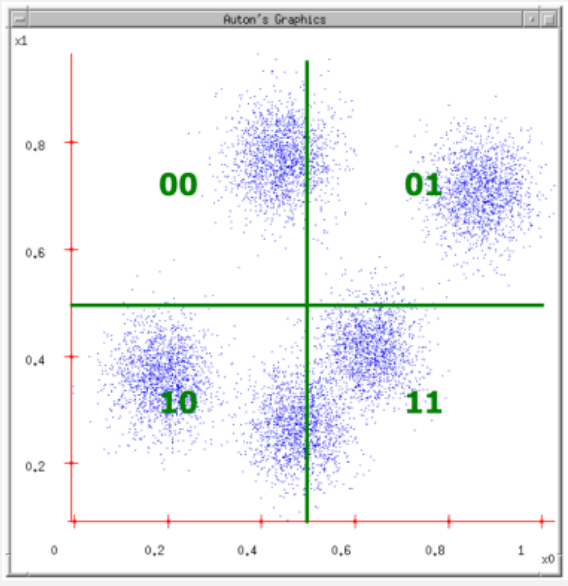
3.1 Algoritmo K-Means

Per spiegare il funzionamento dell'algoritmo K-Means prendiamo in considerazione un semplice esempio.

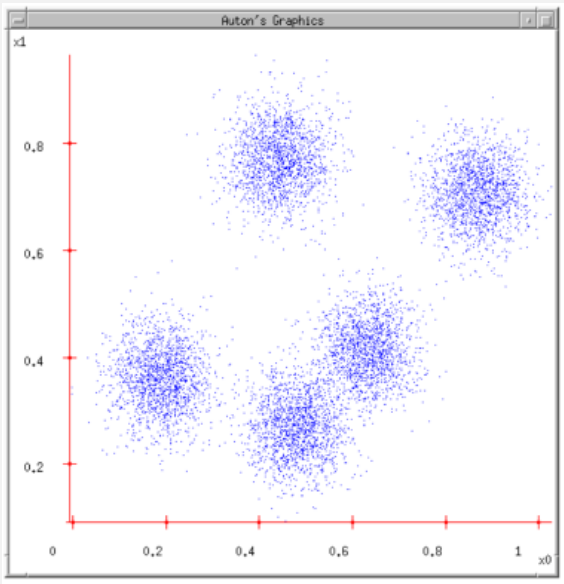
Esempio 3.1



Si può notare come vi siano cinque nuvole di punti (*clouds*). Lo scopo è quello di rappresentare il più fedelmente i dati senza però doverli elencare tutti. Si immagini ad esempio di doverli trasmettere su una linea avente poca banda, non è pensabile inviare tutti i dati, ma si possono inviare solamente n bit, che dunque devono essere rappresentativi dell'intero dataset mantenendo l'errore al minimo. Dunque è necessario un meccanismo di codifica/decodifica in grado da un lato di mappare ogni punto del dataset in un punto che lo rappresenti e dall'altra parte di ottenere i punti originali da quelli "trasmessi".
Una prima soluzione consiste nel dividere il piano in quattro quadranti ognuno individuato da una coppia di bit.



In questo modo codifichiamo ogni punto con il quadrante di appartenenza, e decodifichiamo ogni quadrante con il suo centro. Come si può vedere questa opzione introduce un errore non trascurabile, soprattutto per il quadrante 00. Possiamo migliorare questa idea iniziale ipotizzando di decodificare ogni quadrante con il centroide (*centroid*, centro di massa) dei punti che contiene.



K-Means parte dell'ultima intuizione iterandola fino a trovare i centri dei cluster.

Algoritmo 3.1: K-Means

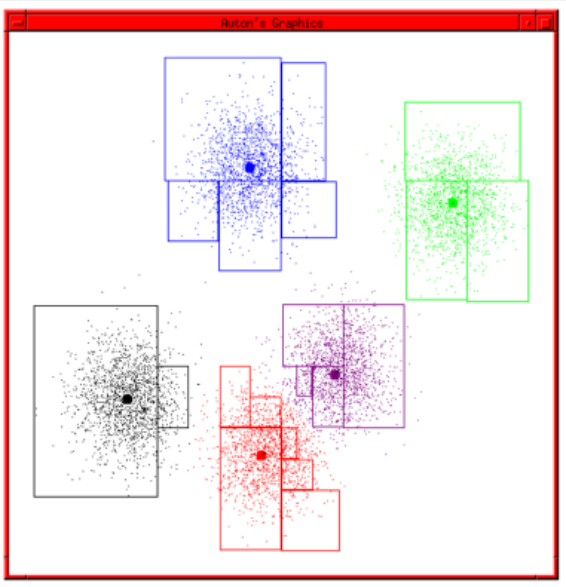
Dato un numero K di cluster si procede come segue:

1. Associare ogni punto del piano al centro più vicino tra i K possibili
2. Per ogni raggruppamento di punti trovare il centroide
3. Muovere il centro di ogni partizione nel corrispondente centroide
4. Se almeno un centro si è spostato ripetere dal punto 1, altrimenti terminare l'algoritmo

Può accadere che ad un certo punto un cluster risulti vuoto (nessun punto appartiene ad un certo centroide). Nel caso in cui ciò accada è possibile usare spostare il centro vuoto in una nuova posizione. O un punto casuale lontano da quello attuale, oppure un punto scelto tra quelli appartenenti al cluster avente distorsione maggiore (vedi definizione 3.1)

Esempio 3.2

Riprendendo l'esempio precedente possiamo applicare l'algoritmo K-Means in modo da trovare i centroidi di ognuna delle nuvole di punti.



3.1.1 Minimizzazione della distorsione

Definizione 3.1: Distorsione

Data una funzione di codifica E ed una funzione di decodifica D

$$E : \mathbb{R}^D \to [1..K] \quad D : [1..K] \to \mathbb{R}^D$$

possiamo definire la distorsione come

$$Distortion = \sum_{i=1}^{N} (e_i - D(E(e_i)))^2$$

È possibile inoltre usare una scorciatoia per chiamare la funzione D, ovvero $D(k) = c_k$. Perciò possiamo ridefinire la distorsione come:

$$Distortion = \sum_{i=1}^{N} (e_i - c_{E(e_i)})^2 = \sum_{j=1}^{K} \sum_{i \in OwnedBy(c_j)} (e_i - c_j)^2$$

Il nome ufficiale della distorsione è *Sum of Squared Errors* ovvero:

$$\begin{aligned} SSE &= \sum_{j=1}^{K} \sum_{i \in OwnedBy(c_j)} (e_i - c_j)^2 \\ &= \sum_{j=1}^{K} SSE_j \end{aligned}$$

Sulla base di questa ultima formula è possibile effettuare alcune considerazioni:

- Un cluster avente SSE elevato è di bassa qualità
- $SSE_j = 0$ se e solo se ogni punto coincide con il proprio centroide
- SSE è una funzione monotona il cui valore decresce all'aumentare di K

La distorsione rappresenta la somma dei quadrati tra le differenze tra i punti reali ed i punti trasmessi. Per fare in modo che la distorsione sia minima occorre che:

1. e_i sia codificato con il centro più vicino. Questo accade perché altrimenti la distorsione potrebbe essere semplicemente ridotta sostituendo $E(e_i)$ con il centro più vicino

$$c_{E(e_i)} = \underset{c_j \in \{c_1..c_k\}}{\operatorname{argmin}} (e_i - c_j)^2$$

2. La derivata parziale della distorsione rispetto alla posizione di ogni centro deve essere zero perché in questo caso la funzione ha un massimo o un minimo in tale punto

$$\begin{aligned} \frac{\partial Distortion}{\partial c_j} &= \frac{\partial}{\partial c_j} \sum_{i \in OwnedBy(c_j)} (e_i - c_j)^2 \\ &= -2 \sum_{i \in OwnedBy(c_j)} (e_i - c_j) = 0 \end{aligned}$$

   Affinché ciò sia vero deve accadere che

$$c_j = \frac{1}{|OwnedBy(c_j)|} \sum_{i \in OwnedBy(c_j)} e_i$$

   ovvero che ogni centro sia il centroide della propria nuvola di punti

Tenendo a mente queste considerazioni ed il fatto che stiamo cercando iterativamente il minimo di una funzione convessa si può dimostrare che l'algoritmo termina. Per quanto sia alto il numero di modi in cui è possibile partizionare un gruppo di N oggetti in K gruppi è finito. Gli stati dell'algoritmo sono dati dai centroidi delle partizioni, pertanto anche il numero di stati possibili è finito. Ogni cambiamento di stato produce una riduzione della distorsione pertanto ogni cambiamento di stato porta ad uno stato non ancora visitato. Questo accade fino a che non vi sono più stati visitabili, ovvero nel momento in cui è stato raggiunto un minimo della funzione che descrive la distorsione.

Va notato che l'algoritmo non permette di comprendere se il minimo raggiunto sia locale o globale (vedi Figura 3.1). Al fine di trovare un minimo globale si possono operare alcuni accorgimenti. Ad

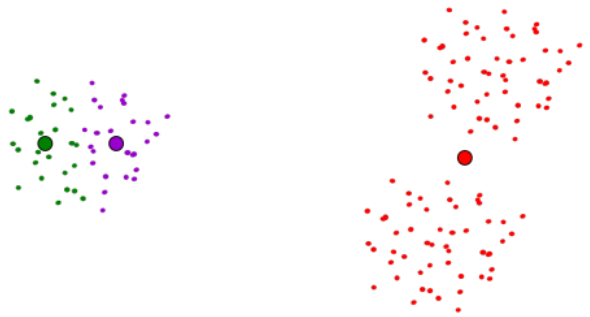


Figura 3.1: Esempio del raggiungimento di un minimo locale

esempio è possibile scegliere casualmente i centri iniziali lontani il più possibile gli uni dagli altri ed eseguire l'algoritmo. Una seconda possibile consiste nell'eseguire l'algoritmo più volte scegliendo ogni volta i centri iniziali in maniera diversa e scegliendo alla fine la configurazione dei centroidi avente distorsione minore, così da ridurre la possibilità di incappare in un minimo locale.

Numero di cluster Infine è importante scegliere il giusto numero iniziale di cluster. Una soluzione è quella di provare un certo numero di valori differenti e, mediante una valutazione quantitativa, scegliere il valore corretto (o comunque il migliore) di cluster. In generale il miglior valore rappresenta il miglior compromesso tra la minimizzazione delle distanze tra i punti interni al medesimo cluster e la massimizzazione della distanza tra i punti di cluster diversi. Tuttavia per quanto notato nella definizione 3.1 non è possibile utilizzare la minimizzazione di SSE in quanto usando $K = N$ si avrebbe una distorsione minima ma anche una classificazione pessima.

La soluzione più ovvia al problema è l'utilizzo della distanza euclidea, la quale rappresenta generalmente una buona soluzione negli spazi vettoriali. Tuttavia va tenuto a mente che esistono varie alternative più specifiche per alcuni tipi di dato e dataset (vedi capitolo 1.4).

3.1.2 Considerazioni generali

Otulier Gli outlier punti aventi un'elevata distanza dai relativi centroidi e che quindi portano un alto contributo al valore di della distorsione del relativo cluster. Per questo motivo hanno una cattiva influenza sul risultato finale. A tal proposito a volte può essere una buona idea rimuovere temporaneamente gli outlier prima dell'esecuzione dell'algoritmo, in modo da ottenere il risultato ottimale, e infine riaggiungerli allo schema.

Complessità La complessità dell'algoritmo è data da

$$O(TKND)$$

dove

- T è il numero di iterazioni
- K è il numero di cluster
- N è il numero di elementi nel dataset
- D è il numero di dimensioni

Casi d'uso È un algoritmo piuttosto efficiente, la cui complessità è quasi linear nel caso in cui $T, K, D << N$. Tuttavia l'uso è limitato ai casi in cui possono essere calcolati i centroidi delle varie nuvole di punti, ovvero i casi in cui i dati sono solamente numerici. Inoltre è piuttosto sensibile al rumore ed alla presenza degli outlier. Infine non è in grado di processare cluster non convessi.

Nonostante abbia questi difetti è un buon algoritmo per l'esplorazione iniziale dei dati e per la discretizzazione (in spazi unidimensionali) di valori in gruppi aventi dimensione non uniforme (vedi figura 1.6).

3.2 Valutazione dello schema di clustering

In questo paragrafo verranno descritti degli schemi per la valutazione del clustering su un insieme di dati generico. Va notato che questa operazione è unicamente collegata al risultato finale e non alla tecnica specifica per effettuare il clustering.

Il clustering non è una tecnica supervisionata, per cui la valutazione dei risultati è cruciale. Tuttavia nel caso in cui vi siano dei dati supervisionati (*supervised*) essi possono essere usati per affinare la valutazione del clustering. Ad esempio è possibile valutare quanto il clustering risulta vicini alla classificazione reale, ed usare questa informazione per aggiornare la valutazione complessiva.

In generale in spazi bidimensionali una visualizzazione grafica può essere utile ad esaminare i risultati tuttavia al di sopra delle tre dimensioni diventa difficile se non impossibile usare visualizzazioni grafiche, pertanto in questi casi si può ricorrere a delle proiezioni bidimensionali, sebbene sia meglio utilizzare metodo più formali.

Esistono diverse criticità da risolvere nella valutazione di uno schema di clustering:

- Distinguere pattern da regolarità apparenti e casuali
- Trovare il miglior numero di cluster (K)
- Valutazione non supervisionata
- Valutazione supervisionata
- Comparazione tra schemi di clustering

3.2.1 Coesione e Separazione

Coesione e separazione sono due criteri per la valutazione di uno schema di clustering:

Definizione 3.2: Coesione

Valore di prossimità degli oggetti appartenenti allo stesso cluster. Più è alta migliore risulta la valutazione del cluster.
Viene calcolata come la somma delle distanze tra gli elementi del cluster e il centro geometrico

$$Coh(k_i) = \sum_{x \in k_i} Prox(x, c_i)$$

Dove c_i può essere o il centroide o il medoide del cluster

Definizione 3.3: Centroide

Punto dello spazio le cui coordinate sono la media di quelle di un insieme di punti del dataset. Concetto analogo al centro di massa

Definizione 3.4: Medoide

Elemento del dataset la cui diversità media rispetto agli altri punti del cluster è minima. Non è necessariamente unico per un cluster, tipicamente viene utilizzato in contesti in cui non è possibile calcolare la media

Definizione 3.5: Separazione

Rappresenta il grado di separazione tra due cluster ed è data (in maniera esclusiva):

- dalla prossimità tra i due prototipi (centroide o medoide)
- dalla prossimità tra i due oggetti più vicini dei due cluster
- dalla prossimità tra i due oggetti più lontani dei due cluster

Definizione 3.6: Separazione globale

Detto c il centroide globale del dataset si definisce il grado di separazione globale di uno schema di clustering come:

$$SSB = \sum_{i=1}^{K} N_i Dist(c_i, c)^2$$

Definizione 3.7: Total Sum of Squares

Questo valore rappresenta il collegamento tra coesione e separazione ed è è una proprietà del dataset indipendente dallo schema di clustering. Essa è esprimibile come la somma dei quadrati delle distanze di ogni punto del dataset dal centroide globale

$$
\begin{aligned}
TSS &= \sum_{i=1}^{K} \sum_{x \in k_i} (x - c)^2 = \sum_{i=1}^{K} \sum_{x \in k_i} ((x - c_i) - (c - c_i))^2 \\
&= \sum_{i=1}^{K} \sum_{x \in k_i} (x - c_i)^2 - 2 \sum_{i=1}^{K} \sum_{x \in k_i} (x - c_i)(c - c_i) + \sum_{i=1}^{K} \sum_{x \in k_i} (c - c_i)^2 \\
&= \sum_{i=1}^{K} \sum_{x \in k_i} (x - c_i)^2 + \sum_{i=1}^{K} \sum_{x \in k_i} (c - c_i)^2 \\
&= \sum_{i=1}^{K} \sum_{x \in k_i} (x - c_i)^2 + \sum_{i=1}^{K} |k_i| (c - c_i)^2 = SSE + SSB
\end{aligned}
$$

Si ricordi che $\sum_{x \in k_i} (x - c_i) = 0$ per definizione di centroide.

Ogni cluster può avere la propria valutazione, inoltre si può considerare un ulteriore divisione per quello peggiore in termini di valutazione. Si possono anche riunire cluster precedentemente divisi nel caso in cui la loro separazione sia debole.

3.2.2 Silhouette

Un altro indice di valutazione dello schema di clustering è la **Silhouette**

Definizione 3.8: Silhouette

Dati i valori a_i e b_i

- Si calcola per l'i-esimo oggetto la distanza media rispetto gli altri oggetti appartenenti allo stesso cluster. Chiamiamo questo valore a_i
- Si calcola per l'i-esimo oggetto e per ogni cluster diverso da quello a cui appartiene l'oggetto la distanza media rispetto a tutti gli oggetti di quel cluster. Il minimo tra questi valori è b_i

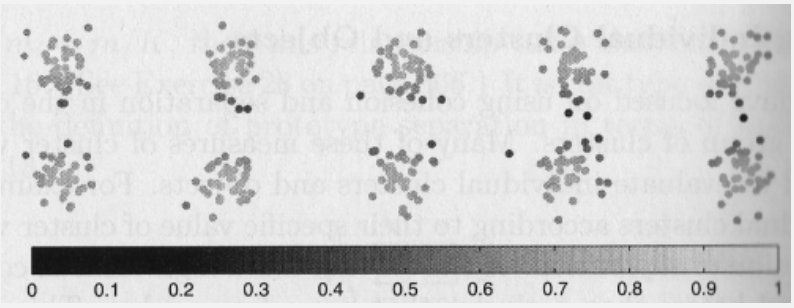
si calcola la silhouette dell'i-esimo oggetto come

$$
s_i = \frac{b_i - a_i}{\max(a_i, b_i)} \in [-1, 1]
$$

Intuitivamente se il valore di Silhouette è positivo l'oggetto considerato appartiene perfettamente al cluster (con uno si indica un'appartenenza perfetta), se invece è negativo significa che oggetti di altri cluster sono tendenzialmente gli sono più vicini rispetto ad oggetti del medesimo cluster.

Questo indice può essere calcolato per ogni oggetto oppure per l'intero dataset (o per un singolo cluster), in questo caso esso è la media degli indici di ogni oggetto.

Esempio 3.3: calcolo silhouette



In figura sono mostrati dieci cluster in uno spazio bidimensionale. I punti più scuri hanno un

valore di Silhouette più basso. Si può dire che la maggior parte di questi punti scuri si trovi sul bordo del cluster o comunque vicini a punti di altri cluster. Al contrario, al centro dei cluster il colore è più sbiadito (indica un valore di Silhouette più alto), questo perché l'influenza dei punti di cluster diversi è minore e quindi la certezza che il punto appartenga effettivamente al cluster è maggiore

Per quando riguarda la complessità, il calcolo dell'indice di Silhouette può essere approssimato a $O(N^2)$ mentre il calcolo del valore SSE può essere approssimato a $O(N)$. Ne deriva che il calcolo del valore di Silhouette è computazionalmente più oneroso.

3.2.3 Scelta del numero di cluster

Alcuni algoritmi, come anche K-means, richiedono il numero di cluster come iperparametro. Per questo motivo è necessario fare diversi tentativi per capire quale sia il miglior valore di K. Gli indici SSE e Silhouette sono influenzati dal numero di cluster, pertanto possono essere usati per calcolare questo valore.

SSE decresce monotonicamente (vedi figura 3.2) all'aumentare del numero di cluster e diventa nullo nel momento in cui si ha $K = N$, pertanto non può essere utilizzato.

Figura 3.2: Variazione dell'indice SSE al variare di K in riferimento all'esempio 3.3

Per quando riguarda il valore di Silhouette il comportamento è diverso (vedi figura 3.3). Al variare del numero di cluster si osserva un massimo in corrispondenza del miglior valore. Questo perché questo è il valore che consente di massimizzare la coesione dei valori interni ai cluster.

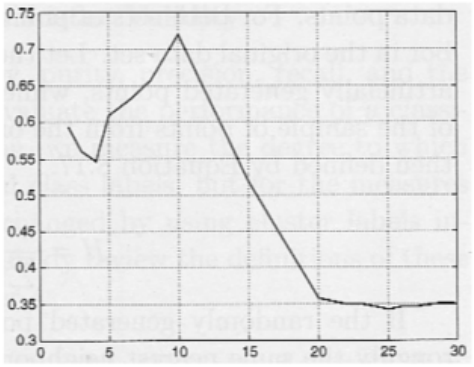


Figura 3.3: Variazione dell'indice silhouette al variare di K in riferimento all'esempio 3.3

3.2.4 Valutazione supervisionata

Ci sono casi in cui si hanno a disposizione alcuni dati supervisionati, questi dati sono chiamati gold standard. Si possono usare questi dati per allenare uno schema di clustering in grado di fare labeling anche sui dati non supervisionati. Anche in questo caso c'è interesse nel valutare lo schema di clustering, si vuole però generalizzare quanto appreso da un piccolo insieme di dati supervisionati all'intero dataset.

In questo caso si vuole usare un classificatore non uno schema di clustering. Per questo motivo si considera uno schema di clustering $K = \{k_1, \ldots, k_K\}$, in cui si usano i dati gold standard per validare lo schema di clustering. Si possono usare diverse tecniche:

- metodi orientati alla classificazione: misurano come le classi siano distribuite tra i cluster utilizzando misure come la *confusion matrix*, *precision*, *recall*, *f-measure*
- metodi orientati alla somiglianza: sono simili per certi versi ai confronti effettuati tra dati binari. Data una partizione $P = \{P_1, \ldots, P_L\}$ (che chiamiamo golden standard) consideriamo lo schema di clustering $K = \{k_1, \ldots, k_K\}$. Ogni coppia di oggetti può essere classificata come:

  a SS se appartengono allo stesso insieme in P e K

  b SD se appartengono allo stesso insieme in k_t ma non in P

  c DS se appartengono allo stesso insieme in P ma non in P

  d DD se appartengono a insiemi diversi sia in K che in P

  Dati questi valori possiamo definire gli indici

  - Indice rand $R = \frac{a+d}{a+b+c+d}$
  - Coefficiente di Jaccard $J = \frac{a}{a+b+c}$

La complessità di calcolo di questi due indici in entrambi i casi può essere approssimata con $O(N^2)$

3.3 Clustering gerarchico

Questo tipo di clustering genera strutture innestate di cluster. Si procede quindi in modo diverso rispetto ad altri algoritmi come K-Means che dato un insieme di punti ed un valore di K divide i punti in K insiemi offre un risultato migliorabile modificando il valore di K e rieseguendo il clustering.

In questo caso invece la suddivisione può essere fatta o top-down o bottom-up:

- Modalità agglomerativa (bottom-up): all'inizio della procedura ogni punto è un cluster a sé. Ad ogni passi si individua la coppia di punti più vicini e li si inserisce nel medesimo cluster. Questa operazione potenzialmente unisce ad ogni passo un punto ad un cluster esistente oppure unisce du cluster esistenti. Per poter operare questo tipo di clustering è necessaria una misura del grado di separazione tra due cluster
- Modalità divisiva (top-down): all'inizio l'intero dataset è un unico cluster. Ad ogni passo il cluster avente il grado di coesione minore viene diviso in due in modo da ottenere un massimo di coesione per entrambi i nuovi sottocluster. Per poter operare questo tipo di clustering è necessaria una misura del grado di coesione dei cluster. Tipicamente questa tecnica è più difficile da applicare rispetto alla precedente, pertanto è meno usata

Il risultato dell'operazione di clustering è o un dendrogramma (figura 3.4 sinistra) o un diagramma dei cluster innestati (figura 3.4 destra) sia per i metodi agglomerativi che per i metodi divisivi.

3.3.1 Separazione tra cluster

Ci sono almeno tre modi per calcolare la separazione tra cluster usando grafi dei punti (figura 3.5):

- Single link: $Sep(k_i, k_j) = \min_{x \in k_i, y \in k_j} Dist(x, y)$
- Complete link: $Sep(k_i, k_j) = \max_{x \in k_i, y \in k_j} Dist(x, y)$
- Link medio: $Sep(k_i, k_j) = \frac{1}{|k_i||k_j|} \sum_{x \in k_i, y \in k_j} Dist(x, y)$

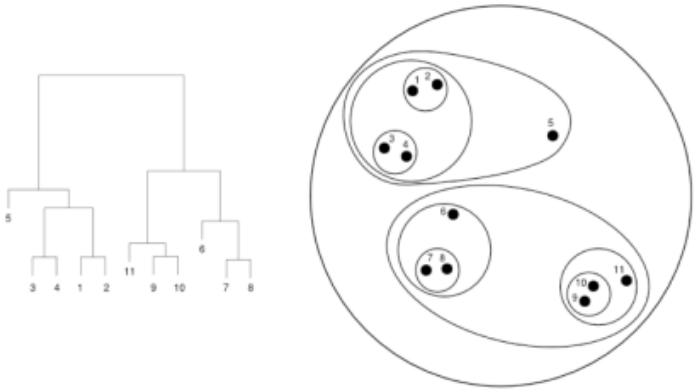


Figura 3.4: Risultato del clustering gerarchico

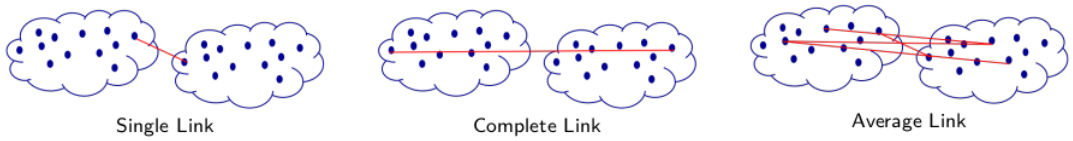


Figura 3.5: Separazione tra cluster basata su grafi

Ci sono inoltre metodi per calcolare il grado di separazione tra cluster a partire dalla posizione dei centroidi:

- Distanza tra i centroidi
- Metodo di Ward: dati due insiemi con i rispettivi SSE, la separazione tra di essi è data dalla differenza dell'SSE totale risultante in caso di fusione dei sue cluster stessi e la somma degli SSE iniziali. In questo modo un grado di separazione minore implica un minore incremento di SSE a seguito della fusione

3.3.2 Clustering single linkage

Algoritmo 3.2: Clustering gerarchico single linkage

Rappresenta un esempio di modello di clustering agglomerativo.

```
inizializza i cluster, uno per ogni oggetto
calcola la matrice della distanza tra tutti i cluster
while (numero di cluster > 1)
    trova i due cluster meno distanti
    uniscili in un unico cluster
    aggiorna la matrice rimuovendo i due cluster uniti ed aggiornando le
        distanze
```

La matrice delle distanze è una matrice quadrata simmetrica con N righe e diagonale nulla. Nel momento in cui avviene la fusione tra due cluster la nuova distanza computata sarà:

$$Dist(k_k, k_{(r+s)}) = \min\left(Dist(k_k, k_{(r)}), Dist(k_k, k_{(s)})\right) \forall k \in [1, K]$$

La complessità temporale è complessivamente $O(N^3)$ che può essere ridotto a $O(N^2 \log(N))$ nel caso in cui si usino strutture indicizzate

3.3.3 Ottenere uno schema di clustering

Utilizzando un dendrogramma per rappresentare il risultato degli algoritmi consente infine di ottenere lo schema di clustering desiderato. Per fare ciò è sufficiente tagliare verticalmente il diagramma nel punto

desiderato (vedi figura 3.6), in quanto il numero di cluster dipende dal caso specifico (è possibile aiutarsi con degli indici per scoprire quale sia il valore corretto, come accade con K-Means). L'asse orizzontale

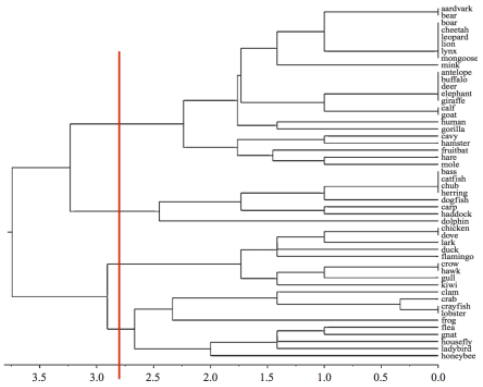


Figura 3.6: Sezionamento di un dendrogramma al fine di ottenere uno schema di clustering

del dendrogramma rappresenta la diversità totale interna ai cluster, la quale diminuisce all'aumentare del numero di cluster ed al diminuire contemporaneo della loro dimensione. Il diametro di un cluster invece è dato dalla distanza tra gli oggetti più distanti interni al cluster. La metodologia *single linkage* tende a generare cluster con diametri maggiori anche ai livelli più bassi, mentre la metodologia *complete linkage* tende a generare cluster più compatti.

Complessivamente però queste soluzioni sono poco scalabili a causa dell'alta complessità computazionale, inoltre non essendoci una funzione obiettivo globale ogni decisione viene presa in termini locali e non può essere annullata. Nonostante ciò la struttura a dendrogramma è particolarmente utili a fini di visualizzazione e di interpretazione dei risultati e porta tipicamente a buoni risultati.

3.4 Clustering basato su densità

Questo tipo di clustering parte da una semplice osservazione: i cluster non sono altro che regioni ad alta densità circondate da regioni a bassa densità. Questa osservazione non tiene conto di concavità e convessità delle nuvole di punti, né della forma specifica e quindi consente in teoria di individuare cluster di ogni forma e dimensione.

Per calcolare la densità di una regione di spazio si possono adottare due soluzioni di base. La prima è basata sull'utilizzo di una griglia che divide l'iperspazio in un certo numero di celle regolari che contengono oggetti. Questa soluzione è piuttosto semplice da realizzare e misura la densità come numero di oggetti all'interno della singola cella, tuttavia tutto dipende dalle dimensioni della cella stessa. Infatti se la cella è troppo piccola siamo in grado di osservare moltissimi dettagli e piccole variazioni in densità, tuttavia il calcolo diventa estremamente esoso nel suo complesso; se invece le celle sono troppo grandi si perde la cognizione dei dettagli e si tende a confondere il rumore con i dati facenti parte dei cluster. La seconda soluzione consiste nell'utilizzare delle ipersfere a partire da ognuno dei punti. Ogni oggetto viene associato agli altri oggetti che rientrano nella sua ipersfera. In questo caso il raggio della sfera è un iperparametro dell'algoritmo e consente di regolare il livello di dettaglio desiderato.

3.4.1 DBSCAN

DBSCAN sta per *Density Based Spatial Clustering of Applications with Noise*. Intuitivamente consente di costruire cluster aventi un centro (*core*) ed un bordo (*border*) sulla base del concetto di vicinato (*neighborhood*). Intuitivamente il punto p è un punto "border", mentre q è un punto "core" (vedi figura 3.7) A questo punto per ogni punto dello spazio viene definito il concetto di "neighborhood" di un punto.

Definizione 3.9: Neighborhood

Dato un punto p definiamo neighborhood del punto p come l'insieme dei punti all'interno dell'ipersfera avente raggio ϵ centrata in p

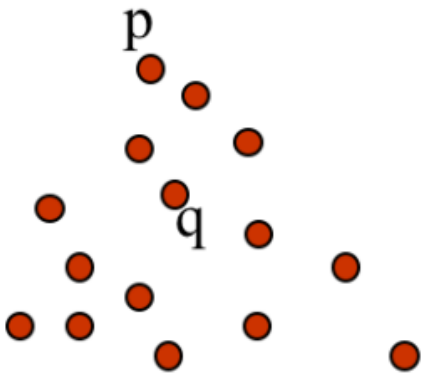
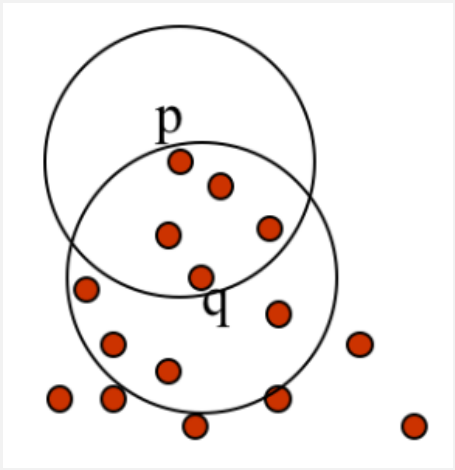


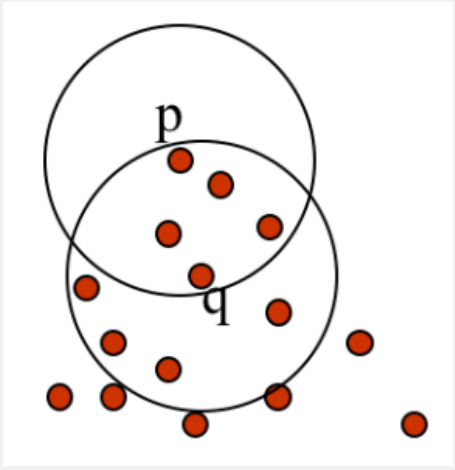
Figura 3.7



Come si può vedere dalla figura la relazione di vicinato è simmetrica

Definizione 3.10: Punto core, punto border

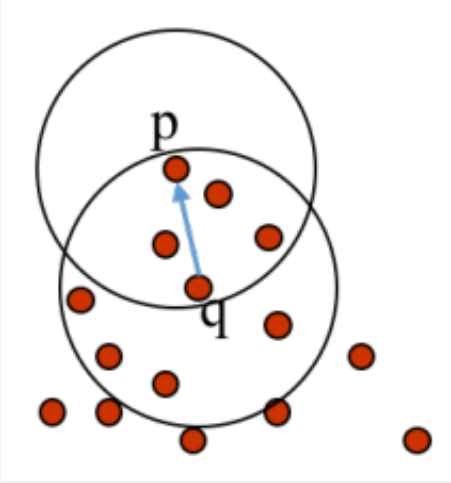
Data una soglia *minPoints*, un punto q è definito core se il suo vicinato comprende almeno *minPoints* oggetti, mentre un punto p è border se il suo vicinato comprende meno di *minPoints* oggetti



Definizione 3.11: Direct Density Reachability

Un punto p è direttamente raggiungibile per densità da un punto q se e solo se:

- q è un punto core
- q è all'interno del vicinato di p

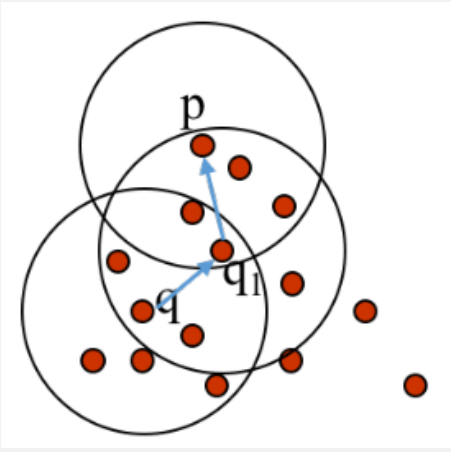


La raggiungibilità diretta non è simmetrica

Definizione 3.12: Density Reachability

Un punto p è raggiungibile per densità da un punto q se e solo se:

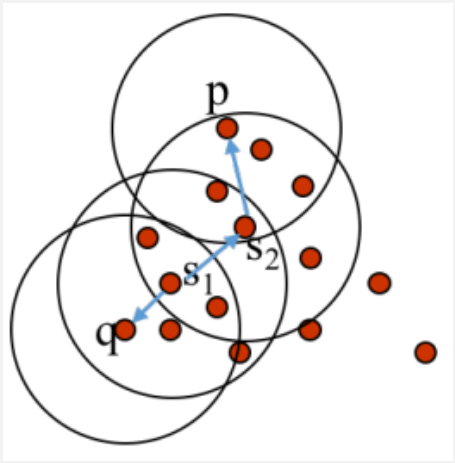
- q è un punto core
- esiste una sequenza di punti $(q, q_1, \ldots, q_{nq})$ tale per cui q_1 è raggiungibile direttamente da q, ogni punto q_{n+1} è direttamente raggiungibile da q_n e p è direttamente raggiungibile da q_{nq}



La raggiungibilità non è simmetrica

Definizione 3.13: Density Connection

Un punto p è connesso per densità ad un punto q se e solo se esiste un punto s tale per cui sia p che q sono raggiungibili per densità a partire da s



La connessione per densità è simmetrica

Definizione 3.14: Clustering con DBSCAN

Un cluster è l'insieme massimo di punti connessi per densità dati un valore di ϵ e $minPoints$ iniziali. I punti "border" che non appartengono ad alcun cluster sono etichettati come rumore

Usando questa definizione siamo in grado di distinguere le arre ad alta densità dalle aree a bassa densità.

Proprietà

L'algoritmo gode delle seguenti proprietà:

- DBSCAN è in grado di trovare cluster di ogni forma. Infatti usando il concetto di punti "border" l'algoritmo è in grado di seguire facilmente la forma di qualsiasi cluster. Questo non accade ad esempio con K-Means il quale invece considera le distanze dal centroide della nuvola di punti e quindi è in grado di trovare unicamente cluster convessi
- In questo algoritmo è inclusa l'idea di rumore, rendendo l'algoritmo particolarmente robusto anche in caso di rumore ed in presenza di outlier
- I parametri ϵ e $minPoints$ sono molto sensibili, anche piccole variazioni producono grandi mutamenti nei risultati del clustering. I due parametri lavorano in sensi opposti. All'aumentare di ϵ aumenta il grado di inclusione, quindi tendenzialmente si avrà un numero minore di cluster i quali avranno maggiori dimensioni, al contrario al calare di questo parametro si avrà un numero maggiore di cluster aventi però dimensioni minori. All'aumentare di $minPoints$ invece diminuisce il grado di aggregazione in quanto questo parametro rappresenta la soglia oltre la quale un punto è considerato "core", quindi si avranno cluster più piccoli
- Non ha un buon comportamento nel caso in cui vi sino grandi differenze di densità tra i vari cluster. Questa proprietà è in qualche modo collegata alla precedente, in quanto i parametri sono i medesimi per la divisione dell'intero dataset. Nel caso in cui i parametri favoriscano l'aggregazione saranno riconosciuti come cluster anche le nuvole di punti meno dense, tuttavia potrebbero essere riconosciute com un unico cluster nuvole molto dense che in realtà sono nettamente separate. Allo stesso modo se i parametri sono tarati per riconoscere nuvole dense aumenterà il grado di rumore e di conseguenza le nuvole meno dense verranno riconosciute come rumore e non come cluster
- La complessità computazionale è pari a $O(N^2)$, che si riduce a $O(Nlog(N))$ se sono presenti strutture indicizzate

3.5 Clustering model based

Uno dei primi approcci al clustering è quello del clustering model based. Esso è un approccio parametrico che parte dall'ipotesi che i dati siano generati da un insieme di distribuzioni di cui è ignoto però il numero. Tale numero rappresenta il numero di cluster. Si suppone in questo caso che gli attributi siano indipendenti tra loro.

Poiché le varie distribuzioni si trovano all'interno di un spazio n-dimensionale ognuna di esse avrà un insieme di parametri per ognuna delle dimensioni dello spazio. Nel caso in cui i dati possono essere approssimati con un'unica distribuzione ricavarne i parametri è triviale, nel caso in cui vi siano più distribuzioni (anche diverse tra loro) viene utilizzato l'algoritmo **Expectation Maximization**

Supponiamo ora che la distribuzione seguita dai dati sia gaussiana.

3.5.1 Algoritmo EM

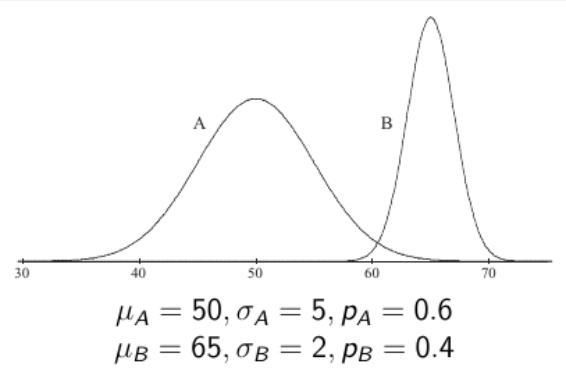
Algoritmo 3.3: Expectation Maximization

```
seleziona un insieme iniziale di parametri
while (i parametri non cambiano)
    per ogni oggetto calcola la probabilita' che appartenga ad ogni
        distribuzione
    date le probabilita' stimate trova una nuova stima dei parametri
        che massimizzi la verosimiglianza attesa
```

Il primo passo all'interno del ciclo è il cosiddetto passo "Expectation", il secondo invece è il passo "Maximization"

Esempio 3.4

In figura è rappresentato un possibile risultato dell'algoritmo EM



In questo caso occorre calcolare cinque parametri (media e deviazione standard per A, media e deviazione standard per B, probabilità per A). Diciamo:

$$P(A|x) = \frac{P(x|A)P(A)}{P(x)} = \frac{f(x; \mu_A, \sigma_A)p_A}{P(x)}$$

dove

$$f(x; \mu_A, \sigma_A) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{(x-\mu)^2}{2\sigma^2}}$$

Per il calcolo della media e della deviazione standard si procede come segue:

$$\mu_A = \frac{w_1 e_1 + w_2 e_2 + \cdots + w_N e_N}{w_1 + w_2 + \cdots + w_N}$$

$$\sigma_A = \frac{w_1(e_1 - \mu_A)^2 + w_2(e_2 - \mu_A)^2 + \cdots + w_N(e_N - \mu_A)^2}{w_1 + w_2 + \cdots + w_N}$$

3.6 Considerazioni finali

In generale l'efficacia di un algoritmo di clustering diminuisce all'aumentare del numero di dimensioni D dello spazio in cui si trovano i dati e con l'aumentare del rumore di fondo. Il costo computazionale invece cresce all'aumentare delle dimensioni N del dataset ed all'aumentare del numero delle dimensioni D in cui si trovano i dati.

Capitolo 4

Regole di associazione

Le regole di associazione sono un'altra metodologia di apprendimento non supervisionata che mira ad imparare relazioni tra i dati a partire dai dati stessi. In questi casi diventa fondamentale una misura della qualità dei risultati

4.1 Introduzione alla "analisi del carrello"

Questo problema è stato il capostipite dell'analisi delle regole di associazione. Dato un insieme di transazioni commerciali, questa analisi si pone il problema di trovare regole che possano predire la presenza di elementi in carrelli futuri basate sul contenuto dei carrelli passati.

In questo caso possiamo considerare l'intero inventario del supermercato come se fosse l'inventario (spesso chiamato *universo*) ed una transazione riguarda un piccolo sottoinsieme degli elementi facenti parte dell'insieme universo. Come semplificazione si considera solamente la presenza degli elementi all'interno del carrello e non la loro quantità.

Possiamo descrivere una transazione come un vettore di bit la cui cardinalità coincide con quella dell'insieme universo. Solamente i bit associati agli elementi presenti nel carrello saranno posti ad 1, mentre gli altri sono posti a 0. Un'altra modalità per rappresentare una transazione è quella che vede l'uso di una tabella delle transazioni (vedi figura 4.1). In questo caso ogni riga è associata ad una transazione avente un proprio TID. Si noti che questa tabella non è una tabella relazionale, in quanto la seconda colonna (quella contenente la lista degli elementi) non è in forma normale.

TID	*Items*
1	Bread, Milk
2	Beer, Bread, Diaper, Eggs
3	Beer, Coke, Diaper, Milk
4	Beer, Bread, Diaper, Milk
5	Bread, Coke, Diaper, Milk

Figura 4.1: Insieme di transazioni

Dalle transazioni in figura 4.1 si possono derivare alcune regole di associazione:

- $\{Diaper\} \rightarrow \{Beer\}$
- $\{Bread, Milk\} \rightarrow \{Coke, Eggs\}$
- $\{Beer, Bread\} \rightarrow \{Milk\}$

Queste regole possono essere interpretate come "se nel carrello sono presenti gli elementi sulla sinistra allora è probabile che siano presenti anche quelli sulla destra". Si noti l'espressione "*è probabile che*",

infatti queste regole sono diverse dall'implicazione logica, sono regole vere entro un certo livello di tolleranza. Infine sono regole che indicano la co-occorrenza di due eventi e non implicano in alcun modo causalità di alcun tipo.

Definizione 4.1: Itemset

Definiamo itemset un insieme di uno o più elementi

Esempio 4.1: itemset

$$\{Milk, Bread, Diaper\}$$

Definizione 4.2: k-Itemset

Definiamo itemset un insieme di k elementi

Definizione 4.3: Regola di associazione

Definiamo regola di associazione una regola della forma $A \Rightarrow C$ dove A e C sono itemset detti rispettivamente antecedente e conseguente

Esempio 4.2: regola di associazione

Riprendendo l'esempio in figura 4.1 si ha:

$$\{Milk, Diaper\} \Rightarrow \{Beer\}$$

Definizione 4.4: Support count

Definiamo support count (σ) come la frequenza di occorrenza di un itemset all'interno dell'intero insieme delle transazioni

Esempio 4.3: support count

Riprendendo l'esempio in figura 4.1 si ha: $\sigma(\{Milk, Bread, Diaper\}) = 2$
Questo indicatore è antimonotono, ovvero può solo diminuire all'aumentare della cardinalità dell'insieme valutato

Definizione 4.5: Support

Definiamo support (sup) come la frequenza relativa di un itemset all'interno dell'intero insieme delle transazioni

Esempio 4.4: support

Riprendendo l'esempio in figura 4.1 si ha: $sup(\{Milk, Bread, Diaper\}) = 2/5$

Definizione 4.6: Confidence

Definiamo confidence ($conf$) di una regola di associazione come

$$\frac{\sigma(A \cup C)}{\sigma(A)}$$

Esempio 4.5: confidence

Riprendendo l'esempio in figura 4.1 si ha:

$$conf(\{Milk, Diaper\} \Rightarrow \{Beer\}) = \frac{\sigma(\{Milk, Diaper, Beer\})}{\sigma(\{Milk, Diaper\})}$$

Definizione 4.7: Itemset frequente

Definiamo itemset frequente un itemset il cui support è superiore ad un certo valore *minsup*

I valori di *sup* e *conf* sono importanti indicatori di interesse nei confronti di regole. Nel caso in cui una regola abbia un basso livello di support potrebbe essere stata prodotta da un'associazione casuale e quindi poco interessante ai fini della previsione di comportamenti futuri. Regole aventi un baso livello di confidence invece sono poco affidabili in quanto è vero che sono situazioni successe e reali, ma sono probabili. Un caso diverso riguarda quelle regole aventi un basso valore di support, ma al contempo un elevato livello di confidence. Queste regole rappresentano eventi rari ma piuttosto interessanti.

Si può notare infine che le regole aventi origine nel medesimo itemset hanno il medesimo livello di support ma differiscono per il valore di confidence.

Esempio 4.6

Riprendendo l'esempio in figura 4.1 si ha:

$$\begin{aligned}
\{Diaper, Milk\} &\Rightarrow \{Beer\} && sup = 0.4 \quad conf = 0.67 \\
\{Beer, Milk\} &\Rightarrow \{Diaper\} && sup = 0.4 \quad conf = 1 \\
\{Beer, Diaper\} &\Rightarrow \{Milk\} && sup = 0.4 \quad conf = 0.67 \\
\{Beer\} &\Rightarrow \{Diaper, Milk\} && sup = 0.4 \quad conf = 0.67 \\
\{Diaper\} &\Rightarrow \{Beer, Milk\} && sup = 0.4 \quad conf = 0.5 \\
\{Milk\} &\Rightarrow \{Beer, Diaper\} && sup = 0.4 \quad conf = 0.5
\end{aligned}$$

Scopo di un algoritmo è quindi trovare tutte quelle regole che sono interessanti, ovvero aventi dei valori minimi di support e confidence di almeno *minsup* e *minconf* rispettivamente. Questi sono iperparametri dell'algoritmo, pertanto il risultato dipende dalla loro corretta valutazione. In generale regole aventi support troppo alto non sono di particolare interesse in quanto sono talmente tanto frequenti da essere quasi scontate.

Per prima cosa si potrebbe pensare ad un approccio a forza bruta, ovvero generare tutte le possibili regole di associazione e calcolare i valori di support e confidence per ognuna di esse eliminando tutte quelle che non soddisfano i criteri imposti. In realtà questo approccio è proibitivo in quanto il numero di regole possibile cresce esponenzialmente con $O(3^D)$ (in realtà il numero esatto è $3^D - 2^{D+1} + 1$) dove D è la cardinalità dell'inventario.

È evidente la necessità di dotarsi di un approccio a due fasi

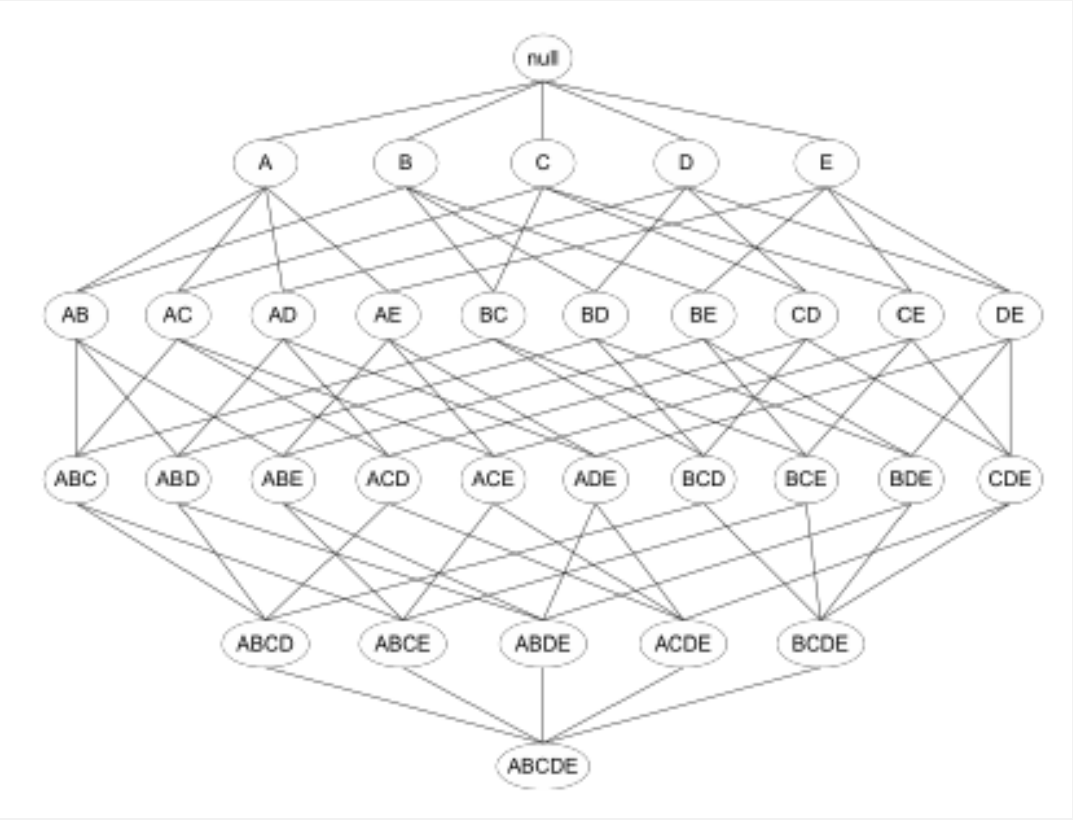
1. **Frequent Itemset Generation** genera tutti gli itemset aventi support maggiore di *minsup*
2. **Rule Generation** genera regole aventi alti livelli di confidence a partire da itemset particolarmente frequenti mediante partizionamento binario dell'itemset stesso

4.2 Generazione del frequent itemset

Introduciamo un esempio che accompagnerà il resto di questo sotto paragrafo.

Esempio 4.7

Supponiamo di avere un inventario contenente D oggetti. Da ciò si evince che vi sono 2^D possibili itemset differenti.



L'approccio a forza bruta procede a generare ogni itemset nel grafo nell'esempio 4.7 e quindi a calcolare il valore degli indici *sup* e *conf* per ognuna delle regole di associazione derivanti da ognuno degli itemset. La complessità computazionale diviene così dell'ordine di $O(NWM)$ dove N è il numero complessivo di transazioni, W è il numero medio di elementi all'interno di una transazione e M è il numero di candidati. È possibile calcolare il numero complessivo di regole di associazione come

$$R = \sum_{k=1}^{D-1} (\binom{D}{k} \times \sum_{j=1}^{D-k} \binom{D-k}{j}) = 3^D - 2^{D+1} + 1$$

Per quanto riguarda la generazione degli itemset frequenti è possibile rifarsi ad alcune strategie di base. In primo luogo è possibile diminuire il numero di candidati complessivo M dal valore iniziale 2^M mediante tecniche di pruning. In secondo luogo è possibile ridurre il numero di confronti dall'originario valore NM usando strutture dati efficienti per memorizzare i candidati e le transazioni in modo tale de rendere non necessario confrontare ogni candidato possibile con ogni transazione.

4.2.1 Algoritmo Apriori

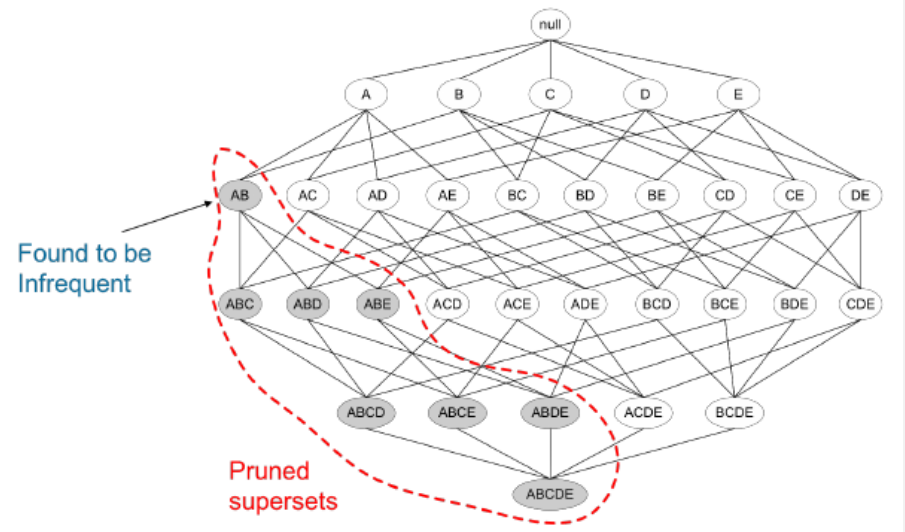
Per ridurre il numero di candidati è possibile usare il principio **Apriori**. Esso sfrutta il fatto che la funzione *sup* è antimonotona, quindi dato un itemset frequente tutti i suoi sottoinsiemi devono essere di conseguenza frequenti a loro volta (questo perché al diminuire degli elementi all'interno dell'itemset si ottiene un insieme avente meno condizioni da soddisfare la cui frequenza è quindi maggiore). Formalmente può essere descritto come:

$$\forall X, Y : (X \subseteq Y) \Rightarrow sup(X) \geq sup(Y)$$

Risulta vera anche l'affermazione opposta, ovvero dato un itemset se esso risulta poco frequente allora anche tutti gli insiemi di cui esso risulta un sottoinsieme saranno poco frequenti a loro volta. Questo fatto consente di potare parte del grafo precedente.

Esempio 4.8

Supponiamo di aver capito che l'itemset $\{AB\}$ sia poco frequente, ne consegue che tutti gli itemset racchiusi nel cerchio rosso (ovvero itemset aventi tra i sottoinsiemi proprio $\{AB\}$) sono poco frequenti e quindi possono essere potati.



Algoritmo 4.1: Apriori

Definiamo C_k come l'insieme dei candidati di dimensione k, L_k l'insieme degli itemset frequenti di dimensione k e $subset_k(c)$ l'insieme dei sottoinsiemi di c aventi k elementi.
L'algoritmo è diviso in due fasi

1. **Join**. Rappresentiamo L_k come una tabella avente k colonne ed in cui ogni riga contiene un itemset frequente. Supponiamo che gli elementi in ogni riga siano ordinati in ordine lessicografico. Il valore di C_{k+1} è generato come un self join su L_k del tipo:

```
insert into C_{k+1}
select item p.item_1, p.item_2, ..., p.item_k, q.item_k
from L_k p, L_k q
where p.item_1 = q.item_1 and ... and p.item_{k-1} = q.item_{
    k-1} and p.item_k < q.item_k
```

2. **Pruning**. Ogni itemset $k+1$-esimo che include un itemset che non è incluso in L_k è rimosso da C_{k+1}

L'algoritmo può quindi essere riassunto come:

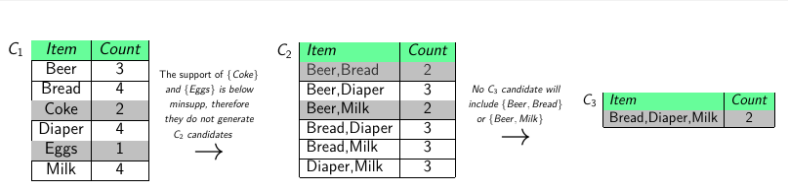
```
L_1 <- itemset di dimensione k frequenti
k <- 1

while L_k != {} do
    C_{k+1} = candidati generati da L_k
    calcola support per ogni elemento di C_{k+1}
    L_{k+1} <- {c} in C_{k+1}: sup(c) >= minsup
    k <- k + 1

return k, L_k
```

Il nome dell'algoritmo deriva dal fatto che ogni calcolo è fatto sulla base della conoscenza pregressa (a priori) acquisita dalla valutazione del livello precedente.
La complessità dell'algoritmo è influenzata

- dal valore di *minsup*. Al diminuire di questo valore aumenta il numero di itemset frequenti e diminuisce l'efficacia del pruning

- dalle dimensioni dell'inventario che si riflette nella necessità di spazio per memorizzare il conteggio di ognuno degli item e nell'incremento del numero di item frequenti
- dalle dimensioni del database ovvero dal numero di transazioni
- dalla dimensione media di una transazione che si traduce nell'incremento della lunghezza media degli itemset frequenti e quindi nel numero di sotto insiemi frequenti

Esempio 4.9: pruning



Utilizzando il pruning si nota che il numero di itemset da valutare passa da 41 a 13 dato che vengono eliminati tutti quelli discendenti da un itemset non abbastanza frequente.

Anche dopo aver filtrato il dataset sulla base del valore di *sup* il numero di itemset frequenti può essere imponente, per questo motivo risulta utile identificare un piccolo numero di itemset frequenti e rappresentativi:

- **Maximal frequent itemset** ovvero il più piccolo insieme di itemset da cui è possibile derivare per intero gli itemset frequenti. Un MFI non ha alcun sovra insieme immediato frequente (vedi figura 4.2) e si trova dunque vicino al bordo divisivo tra itemset frequenti e non frequenti. Questo insieme da una rappresentazione compatta di tutti gli itemset frequenti. Esistono algoritmi in grado di trovarli senza effettuare l'enumerazione dei relativi sottoinsiemi
- **Closed itemset** ovvero la minima rappresentazione degli itemset senza perdete informazioni riguardo al valore di *sup*. Un itemset è **chiuso** se nessuno dei suoi sovra insiemi ha il medesimo valore di support.

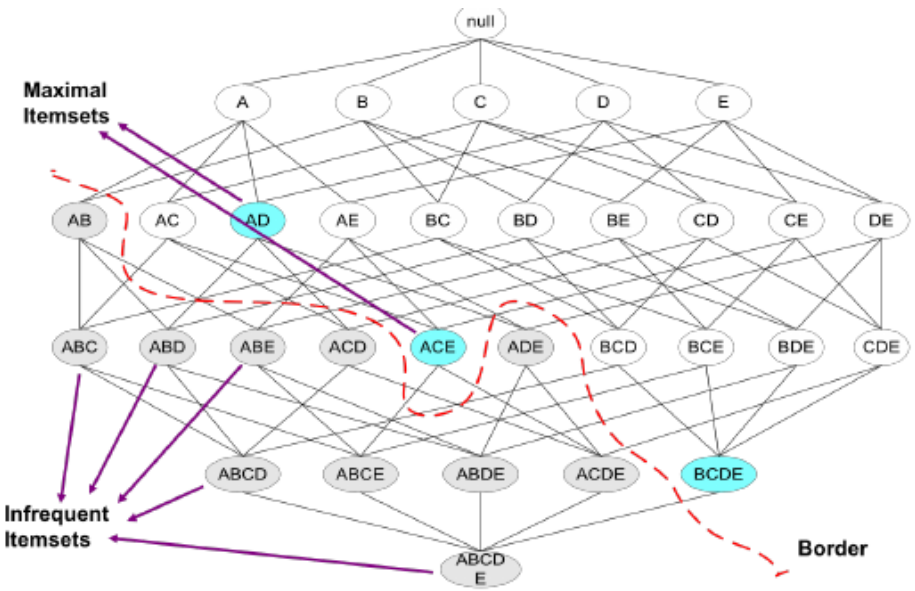


Figura 4.2: MFI relativamente all'esempio 4.7

Si può dimostrare che i **maximal frequent itemset** sono un sottoinsieme dei **closed frequent itemset** che a loro volta sono un sottoinsieme dei **frequent itemset**.

4.2.2 Algoritmo FP-Growth

Il grosso problema dell'algoritmo apriori è il fatto che ha la necessità di generare gli insiemi dei candidati (C_k), il cui numero può essere anche elevatissimo, e per far ciò ha la necessità di scorrere il database numerose volte per effettuare i calcoli relativi al valore di support per ogni candidato.

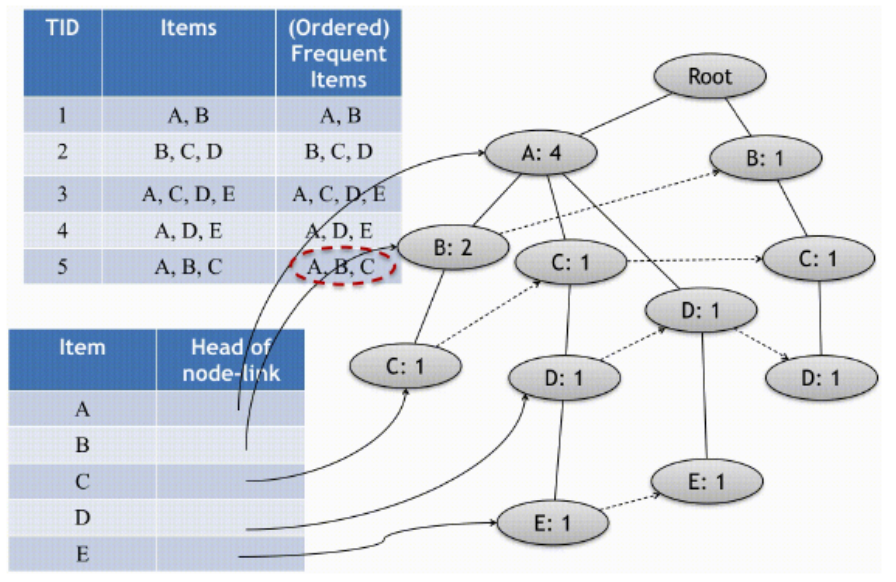
L'algoritmo FP-Growth invece trasforma il problema da trovare pattern lunghi e frequenti a trovare pattern più brevi e concatenare questi con dei suffissi. L'algoritmo usa una rappresentazione compatta del database mediante un FP-Tree, sul quale una volta costruito viene usato un approccio ricorsivo del tipo divide et impera per estrarre gli itemset frequenti.

Algoritmo 4.2: FP-Growth

a fase di costruzione dell'albero può essere riassunta come una serie di passi:

1. Scansiona il database per calcolare il valore di support di ogni itemset contenente un singolo elemento
2. Crea la radice dell'albero
3. Scansiona il database e per ogni transazione:
   (a) Riordina gli elementi secondo valori di support discendenti
   (b) Spostati alla radice dell'albero
   (c) Per ogni item della transazione ordinata
   - Se coincide con uno dei figli del nodo attuale spostati su di esso ed aumentane il contatore associato di 1
   - Altrimenti crea un nuovo nodo figlio avente contatore inizializzato ad 1

Al contempo occorre tener traccia delle posizioni di ogni item mediante una lista linkata



Un fatto interessante è che la struttura dell'albero dipende unicamente dall'ordine in cui vengono valutate le transazioni. Di conseguenza nel caso in cui le transazioni vengano valutate secondo l'ordine dato dal valore di support si ottiene la rappresentazione più compatta possibile dell'albero.

Dato un valore minimo per il valore di support e un albero FP è possibile generare l'insieme degli itemset frequenti. Utilizzando le liste linkate è possibile trovare tutte le catene che terminano con un dato item. Per ognuna di esse occorre risalire alla radice a partire dal nodo trovato seguendo la lista e nel fare questo calcolare il minimo valore del contatore. Se il valore finale ottenuto è superiore alla soglia prestabilita allora la catena rappresenta un itemset frequente

Se con l'algoritmo apriori si ha la necessità di fare la scansione di tutto il database contando il support e il numero di volte che compare la dimensione del più grande Itemsets + 1. Con FP-tree c'è bisogno solo di scansionare una sola volta perché basta un solo passaggio per costruire l'intero albero. Ovviamente sono interessanti le performance di questo algoritmo: cambiano in base a come cambia la soglia, più la soglia di support è piccola più pattern si genereranno (vedi 4.3).

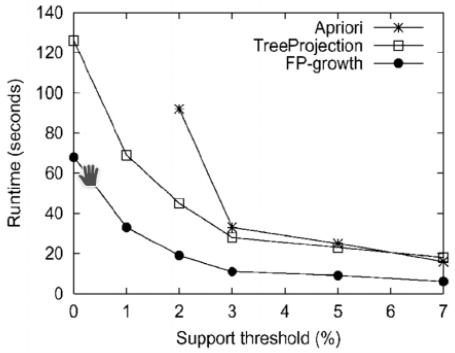


Figura 4.3: Confronto dei tempi di esecuzione al variare della soglia su un dataset sparso

4.3 Generazione delle regole

Trovati i gli itemset frequenti bisogna estrarre le regole. Gli itemset frequenti sono generatori di regole perché le regole sono interessanti solo se sono generate dal Frequent itemset. Per generare le regole ci si appoggia ai valori di support e confidence.

$$conf(A \rightarrow C) = \frac{sup(A \rightarrow C)}{sup(A)}$$

Per il pruning basato sulla confidenza è sufficiente sapere il valore di support degli Itemset frequenti. Dato un itemset frequente L, il problema è trovare tutti i subset non vuoti f per cui il valore di $conf(f \Rightarrow L - f)$ non è minore della confidenza minima. Per questo motivo dato $\{Beer, Diaper, Milk\}$ le possibili regole sono:

$$\{Beer, Diaper\} \rightarrow \{Milk\}$$
$$\{Beer, Milk\} \rightarrow \{Diaper\}$$
$$\{Milk, Diaper\} \rightarrow \{Beer\}$$
$$\{Beer\} \rightarrow \{Milk, Diaper\}$$
$$\{Diaper\} \rightarrow \{Milk, Beer\}$$
$$\{Milk\} \rightarrow \{Beer, Diaper\}$$

Se la dimensione di $|L| = k$ allora ci sono $2^k - 2$ regole possibili (ogni elemento può trovarsi a destra o a sinistra della freccia ed escludiamo il cado in cui siano tutti a destra o tutti a sinistra).

Ci si pone ora il problema circa la generazione efficiente delle regole a partire da un itemset frequente. In generale però la confidenza non è antimonotona, ma gode di una proprietà simile. Se si considerano le regole generate dallo stesso itemset $i = \{A, B, C, D\} appartenentia L$ si ha:

$$conf(ABC \rightarrow D) \geq conf(AB \rightarrow CD) \geq conf(A \rightarrow BCD)$$

quando si sposta un oggetto da sinistra a destra si cambia il denominatore della formula per il calcolo della confidenza, si incrementa il support e si diminuisce il valore della confidenza. Questo rende possibile la potature del grafo contenente tutte le possibili regole. Osservando la figura 4.4, si parte dal grafo che contiene tutte le regole generate partendo da un itemset. Se una regola ha un valore di confidenza sotto la soglia allora anche i suoi discendenti, che sono ottenuti spostando un oggetto da sinistra a destra, avranno indice inferiore alla soglia.

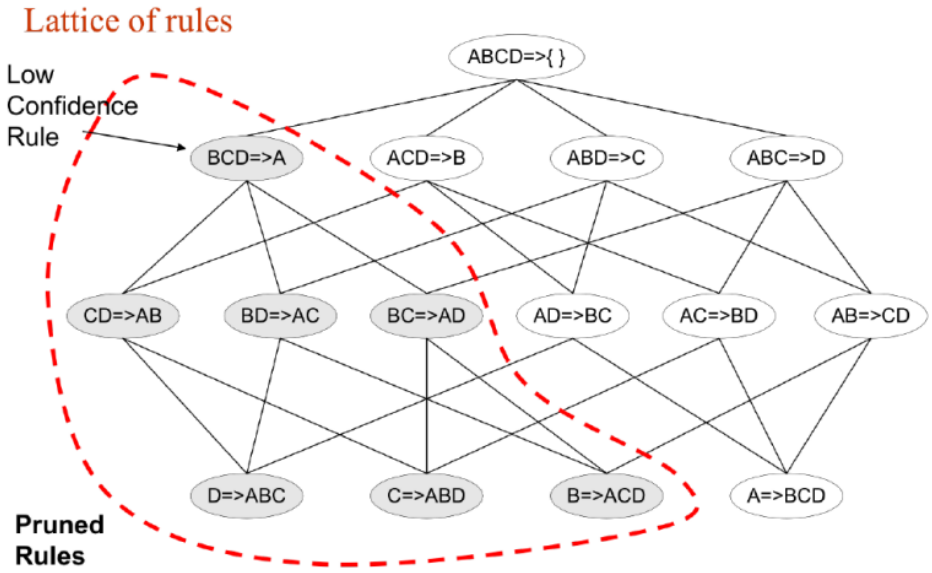


Figure 94: Pruning rules

Figura 4.4: Pruning del grafo contenente tutte le regole ottenibili dall'itemset $\{A, B, C, D\}$

Anche utilizzando l'algoritmo apriori si possono generare regole in modo equivalente mediante join. La regola candidata è generata dal merging di due regole che condividono lo stesso prefisso nella regola conseguente.

- Il join $(CD \rightarrow AB,\ BD \rightarrow AC)$ produrrà la regola $D \rightarrow ABC$
- La regola $D \rightarrow ABC$ sarà potata se il subset $AD \rightarrow BC$ non ha valore di confidenza alta

Ovviamente è necessario impostare una soglia per il supporto *minsup* e una soglia per la confidenza $minconf$ in maniera sensata. In generale la distribuzione del support del singolo oggetto è un indice asimmetrico. Osservando la figura sotto se si ordinano gli oggetti in ordine decrescente rispetto al support e si rappresenta sull'asse y il support count è molto comune avere una distribuzione come quella in figura che implica che ci sono molti item con valore di support molto alto e molti con valore molto basso. La scelta delle soglie è cruciale perché se minsup è troppo bassa, il calcolo dell'algoritmo diventa computazionalmente costoso e la dimensione degli itemset è molto grande. Per questo motivo si vuole essere in grado di usare diversi valori di soglia minsup.

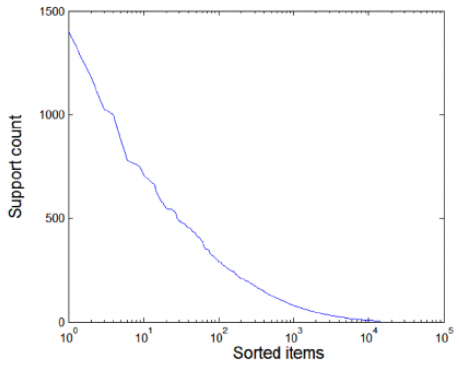


Figura 4.5: Molti dataset reali hanno una distribuzione dell'indice support asimmetrica

Multiple Minimum Support

Ci si può chiedere come scegliere il valore corretto di *minsup* o se occorra sceglierne per forza solamente uno. In alcuni casi potrebbe non essere sufficiente usare un singolo valore in quanto la situazione da

modellare potrebbe essere complessa, in questo caso è possibile definire più valore minimi di support, anche uno per ogni item in modo da poter aggiustare la soglia di support minimo basandosi sulla conoscenza pregressa. In questo ultimo caso tale valore può essere relazionato all'oggetto stesso.

Come maneggiare però quelle transazioni che includono oggetti con soglie differenti? Si può certamente scegliere il minimo delle soglie *minsup* tra tutti gli oggetti, ma in questo modo il support non è più antimonotono. Si può progettare una variante dell'algoritmo per generare gli itemset frequenti che tenga in conto questo support minimo. In questo caso l'idea di base è quella di ordinare gli oggetti secondo il loro minimo support (in ordine ascendente) e quindi modificare l'algoritmo Apriori nel seguente modo:

Algoritmo 4.3: Apriori con support variabile

Detti L_1 l'insieme degli item frequenti, F_1 l'insieme degli item il cui valore di support è $MS(1) = \min_i (MS(i))$ e C_2 l'insieme degli itemset candidati di dimensione 2 generato da F_1 e L_1
Nell'algoritmo Apriori tradizionale, un elemento candidato $(k+1)$-esimo viene generato unendo due itemset frequenti di dimensione k e il candidato viene eliminato se contiene sottoinsiemi poco frequenti di dimensione k. In questo caso va modificato la fase di pruning, nella quale il pruning avviene solo se si trova un sottoinsieme non frequente che contiene il primo elemento dell'itemset stesso.

Valutazione dei pattern

Vediamo ora come valutare i pattern ottenuti a partire dagli algoritmi precedenti. Essi infatti generano un numero elevato di regole, delle quali spesso molte sono ridondanti o poco interessanti.

Esempio 4.10

Le regole

$$\{A, B, C\} \to \{D\} \quad \{A, B\} \to \{D\}$$

sono ridonandoti nel caso abbiano i medesimo valori di support e confidence

L'obiettivo è, tramite parametri specifici, di potare/valutare i pattern derivati. Le misure di interesse sono applicate in diversi passaggi della derivazione della conoscenza a partire dai dati grezzi e non solamente in fase finale. Infatti dopo ogni fase è possibile valutare il risultato intermedio ottenuto per ottenere la miglior valutazione finale.

Per ottenere queste misure di interesse è necessario costruire una *contingency table* basata sulle regole ottenute dall'esecuzione degli algoritmi.
Dalla regola $A \to C$ si ottiene la contingency table in figura 4.6 i cui elementi sono le basi per la maggior parte delle misure di interesse.

	C	$\overline{C}$	
A	f_{11}	f_{10}	f_{1+}
$\overline{A}$	f_{01}	f_{00}	f_{0+}
	f_{+1}	f_{+0}	

Figura 4.6: Contingency table

Esempio 4.11

Supponiamo di avere la seguente contingency table

	$Coffee$	$\overline{Coffee}$	
Tea	15	5	20
$\overline{Tea}$	75	5	80
	90	10	100

$$Tea \Rightarrow Coffee$$

Potrebbe sembrare che la regola $\{Tea\} \rightarrow \{Coffee\}$ abbia un valore piuttosto alto di confidence

$$conf(\{Tea\} \rightarrow \{Coffee\}) = \frac{sup(Tea, Coffee)}{sup(Tea)} = \frac{15}{20} = 0.75$$

Ma dalla tabella si nota che la probabilità di trovare caffè nel carrello ammonta a 0.9 e che la probabilità di trovare caffè ma non tè ammonta a circa il 0.94

$$P(Coffee) = 0.9, \quad P(Coffee|\bar{Tea}) = \frac{75}{80} = 0.9375$$

Nonostante l'alto valore di *conf* l'alta probabilità dell'assenza di tè incrementa la probabilità del caffè, dunque il valore iniziale di confidence è fuorviante

Sono necessarie nuove misure perché a volte la confidenza potrebbe essere ingannevole. Per questo motivo si usa il concetto di **indipendenza statistica**. Il problema insito nel considerare la probabilità invece della confidenza è che la nozione di indipendenza statistica è molto forte e comporta l'assunzione forte che il valore trovato sia la probabilità vera e non la frequenza. Considerando la nozione di intervallo di confidenza, si può esprimere la probabilità come la frequenza più o meno qualcosa ma in questo modo quando il numero diventa piccolo questa probabilità ha un'alta incertezza e se si moltiplicano le probabilità con l'incertezza si ha un'amplificazione della stessa.
Per questo motivo, l'utilizzo della probabilità solo per trovare regole interessanti non è una soluzione valida. È più utile filtrare le regole usando support e confidenza, quindi con una fase di post-elaborazione possiamo usare la nozione di indipendenza statistica (o alcune nuove misure relative a questa) per fare un ulteriore filtraggio mediante:

- **Lift**: è la confidenza di una regola diviso il support del conseguente. Potrebbe anche essere espressa come la probabilità degli eventi congiunti diviso per il prodotto della probabilità

  $$lift(A \rightarrow C) = \frac{conf(A \rightarrow C)}{sup(C)} = \frac{P(A \cap C)}{P(A)P(C)}$$

  Questo indice ha valore $[0, 1]$ dove 1 si raggiunge solo in caso di completa indipendenza. È un valore che non tiene conto della direzione delle regole. Maggiore è il valore maggiore è la probabilità che la regola **non** sia una coincidenza

- **Laverage**: vale 0 nel caso di completa indipendenza, è positiva per correlazione positiva e negativa per correlation correlazione. Non è sensibile alla direzione della regola. Rappresenta il numero di casi in relazione alla indipendenza.

  $$leve(A \rightarrow C) = sup(A \cup C) - sup(A)sup(C) = P(A \cap C) - P(A)P(C)$$

- **Conviction**: rappresenta il rapporto della frequenza aspettata che A si presenti senza C, è chiamata anche *novelty*, è sensibile alla direzione della regola e risulta infinita se la regola è sempre vera

  $$conv(A \rightarrow C) = \frac{1 - sup(C)}{1 - conf(A \rightarrow C)} = \frac{P(A)(1 - P(C))}{P(A) - P(A \cap C)}$$

  Maggiore è il valore meno la regola viene violata rispetto ai casi in cui antecedente e conseguente sono scorrelati

Esempio 4.12

Riprendendo l'esempio precedente si ottengono i seguenti valori:

- $conf(\{Tea\} \rightarrow \{Coffee\}) = \frac{15}{20} = 0.75$ questo valore è apparentemente alto
- $lift(\{Tea\} \rightarrow \{Coffee\}) = \frac{0.15}{0.9 \cdot 0.2} = 0.83$ il valore è minore di 1 quindi non è interessante
- $leve(\{Tea\} \rightarrow \{Coffee\}) = 0.15 - 0.9 \cdot 0.2 = -0.03$ il valore è minore di zero quindi la regola non è interessante
- $conv(\{Tea\} \rightarrow \{Coffee\}) = \frac{1-0.9}{1-0.75} = 0.4$ questo valore è basso (il valore massimo è $+\infty$)

Per concludere, le proprietà che una buona misura M deve soddisfare sono:

- $M(A, B) = 0$ (or 1) se A e B sono statisticamente indipendenti
- $M(A, B)$ cresce monotonamente con $P(A, B)$ quando $P(A)$ e $P(B)$ rimangono invariate
- $M(A, B)$ decrescono monotonamente con $P(A)$ (o $P(B)$) quando $P(A, B)$ e $P(B)$ (o $P(A)$) rimangono invariate

Ci sono varie misure proposte in letteratura oltre a quelle presentate. La confidence è solitamente la misura di base, le altre misure possono essere usate per testare il risultato dato dalla confidenza e per filtraggi ulteriori.

Regole di associazione multidimensionali

Nei paragrafi precedenti è stato descritto come ottenere regole relazionali da database relazionali. Espandiamo ora questi concetti anche alle tabelle relazionali in generale. Questo significa non avere più regole binarie, ovvero che coinvolgono solo due attributi per volta.

Esempio 4.13

TID	nationality	age	income
1	Italian	50	low
2	French	40	high
3	French	30	high
4	Italian	50	medium
5	Italian	45	high
6	French	35	high

In tabella sono presenti tre attributi. Ogni componente può essere interpretato come un evento, per questo le possibili regole di associazione tra valori e attributi possono essere:

$$\begin{aligned} nationality = French \rightarrow income = high \quad & [sup = 0.5, \quad conf = 1] \\ income = high \rightarrow nationality = French \quad & [sup = 0.5, \quad conf = 0.75] \\ age = 50 \rightarrow nationality = Italian \quad & [sup = 0.33, \quad conf = 1] \end{aligned}$$

Da ciò si deduce che, anche una tabella relazionale standard, può avere rappresentate informazioni che possono essere usate per ricavare regole di associazione. A tal proposito si può ragionare in due modi:

- **Mono-Dimensionale**: gli attributi sono gli oggetti nella transazione. L'evento è la transazione che è descritta come: gli oggetti A, B e C sono insieme in una transazione
- **Multi-Dimensionale**: l'evento è la tupla ed è descritto come: l'attributo A ha valore a, l'attributo B ha valore b e l'attributo C ha valore c

In generale è possibile passare da una rappresentazione monodimensionale ad una multidimensionale tramite regole di trasformazione.

L'approccio multidimensionale però soffre in caso di attributi quantitativi (vedi figura 4.7). Per esempio, si considerino gli attributi della tabella sotto, ci son troppi valori distinti per trasformazione (multi/mono). Molti software per le regole di associazione non riescono a maneggiare attributi quantitativi. Per risolvere questo problema si procede con la tecnica della discretizzazione perdendo però la possibilità di fare confronti e ordinare gli attributi.

TID	height	weight	income
1	168	75.4	30.5
2	175	80.0	20.3
3	174	70.3	25.8
4	170	65.2	27.0

Figura 4.7: Transazione con dati numerici

Regole di associazione multilivello

Uno step ulteriore è quello di includere all'interno delle regole di associazione della conoscenza pregressa. Ciò permette di fare confronti tra gli oggetti del dataset a su più livelli (vedi figura 4.8), permettendo di organizzare gli oggetti in strutture gerarchiche.

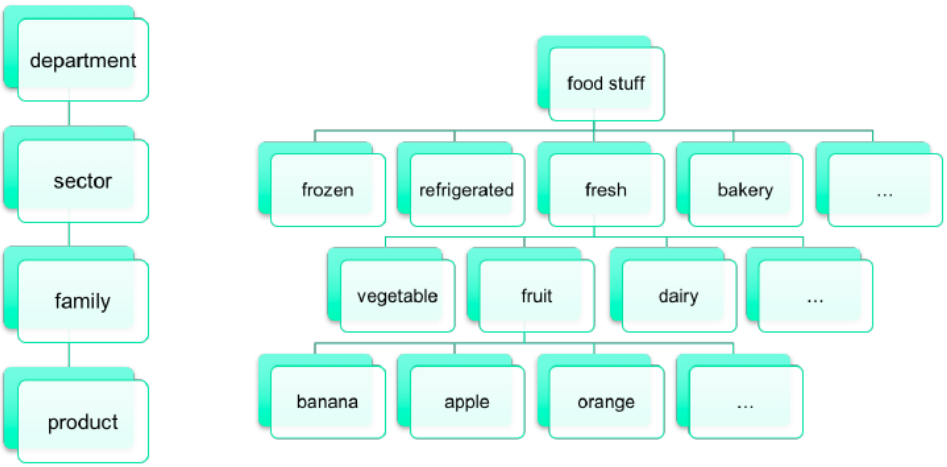


Figura 4.8: Gerarchia concettuale dei prodotti di un supermercato

Questo permette di osservare le regole su più livelli. Spostandosi da regole specifiche a regole generali si guadagna una visione d'insieme che generalmente consente di avere un'idea più precisa di come le cose stiano andando globalmente. In questo caso in generale il valore di support delle regole cresce in quanto le regole includono categorie più ampie. Al contrario, spostandosi da regole generali a regole specifiche, il valore di support diminuisce (scendendo potenzialmente sotto la soglia minima).

Per quanto riguarda il valore di confidence non è possibile trovare una regola specifica, in quanto in generale cambia sia il numeratore che il denominatore delle regole. Tuttavia è possibile dire che se una regola più specifica ha circa lo stesso valore di confidenza della sua corrispettiva più generale allora essa è ridondante.

Parte II

Data Mining

Capitolo 5

Introduzione

I dati esistono indipendentemente dal Data Mining e dal Machine Learning, ma questi argomenti sono spesso correlati. Sebbene queste branche siano nate ben prima dell'avvento dei big data, esse sono infatti necessarie per analizzare in maniera efficace tali moli di dati, fatto che non fa altro che accrescerne l'importanza. A partire dai dati grezzi è infatti possibile ricavare varie informazioni mediante l'osservazione di relazioni tra i dati stessi. Grazie a queste informazioni è poi possibile ricavare nuova conoscenza oppure prendere decisioni migliori o semplicemente più mirate.

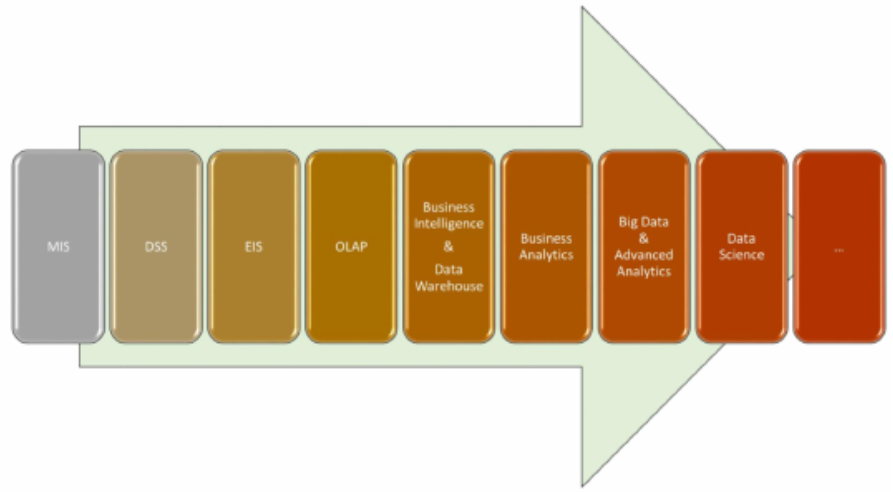


Figura 5.1: Inizialmente i dati sono raccolti in forma grezza, tramite selezione, organizzazione ed analisi è poi possibile ricavare informazioni utili

La Data Science (vedi figura 5.1) è la scienza che partendo dai dati grezzi riesce a capire quali siano le decisioni migliori con tecniche di Machine Learning e Data Mining.

Bisogna ora capire da dove derivino i dati. Un *business process* è un insieme di attività umane volte a realizzare un *business goal*. Ogni attività di un business process genera dati in un'organizzazione moderna. Quando nel mondo reale un evento cambia stato dell'organizzazione, una transazione riflette questo cambiamento all'interno del database. In pratica una transazione è la controparte digitale di ciò che accade nel mondo reale.

5.1 Sistemi e software

On-Line Transaction Processing (OLPT) I software OLTP sono una classe di applicativi in grado di supportare applicazioni orientate alle transazioni e alla memorizzazione dei dati. Questo genere di software è pensato per tener traccia delle transazioni necessarie alla conduzione di un'azienda. Di conseguenza caratteristiche chiave di questi software sono la disponibilità, la velocità, il grado di concorrenza e la robustezza ai danni.

Enterprise Resource Planning (ERP) È un sistema integrato evoluzione dei DBMS, un software che riesce a gestire tutti gli aspetti dei business process:

- Possono gestire tutti i business process di un'azienda con un singolo applicativo
- Contengono un database comune a tutte le applicazioni
- Operano quasi in real-time

Management Information System (MIS) MIS è uno strumento usato per ottenere report periodici standard costruito su un OLTP. Viene usato per dare informazioni sul significato dei dati e per cercare di dare una rapida visone globale ai manager aiutandoli a prendere le decisioni migliori.

Decision Support System (DSS) Per DSS si intende un sistema interattivo e user-friendly che garantisce supporto nel gestire complesse decisioni non strutturate (ovvero decisioni che non possono essere descritte nei dettagli se non dopo che sono state prese). Il DDS cerca di combinare l'uso di modelli e di analytics con l'accesso tradizionale ai dati.

Executive Information System (EIS) EIS è usato per formulare decisioni strategiche che impattano l'operato dell'azienda. Per esempio, dato il trend dei ricavi dell'ultimo periodo, quale sia il compenso per i lavoratori.

Business Intelligence (BI) Con la locuzione business intelligence (BI) ci si può solitamente riferire a:

- un insieme di processi aziendali per raccogliere dati ed analizzare informazioni strategiche
- la tecnologia utilizzata per realizzare questi processi
- le informazioni ottenute come risultato di questi processi

Lo scopo principale della BI è quello di dare, con complesse architetture hardware e software, visioni differenti dei dati grezzi in maniera real-time, cosa che un comune DBMS non può garantire. Ai piedi

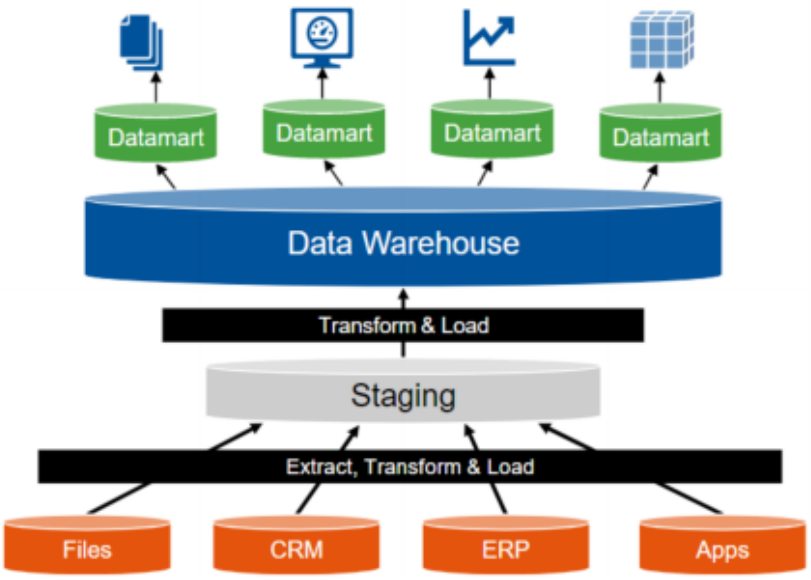


Figura 5.2: Schema architetturale della business intelligence

dell'architettura (vedi figura 5.2) vi è il database. Appena sopra al database vi è uno strato software che estrae i dati grezzi immagazzinati e li colleziona in un'area di staging. Questi dati sono trasformati e collezionati nel data warehouse. Il data warehouse è una vista uniforme di tutti i dati di un'azienda, sul data warehouse possono essere fatte operazioni specifiche per l'amministrazione dell'azienda stessa.

OnLine Analytical Processing (OLAP) Con OLAP si esegue un'analisi globale dell'azienda, si analizzano dati di tipo multidimensionale da più prospettive diverse. Nella pratica equivale a fare una enorme quantità di GROUP BY, per fare questo in maniera più efficiente si usano strutture dati ed algoritmi appositi (ad es. Data Cube).

Structured		*Unstructured*	
Description	***Example***	***Description***	***Example***
Made under an established situation	Hiring a new employee	Made under an emergent situation	Fire breakout
Programmed	Start the monthly payment of salaries	Unplanned	Opportunity for financial investment
Fully understood	When a bank customer makes huge fund movements ask him the reason	Unclear or uncertain	Necessary to acquire information to understand which operation is to be performed
Routine task	Hiring new personnel in a given sector	Suddeen 0ne-shot situation	Dealing with a labor strike
Specified process	Manufacturing something	General processes	Managing security for IT equipment
Well defined methodology	Possible withdraw of funds from international accounts according to currency rates	Decisions relying on knowledge and/or expertise and on analysis of information	What new market segment could be targeted

Figura 5.3: Confronto tra decisioni strutturate e non strutturate

Decisioni strutturate e non strutturate

5.2 Introduzione ai Big Data

Durante l'epoca moderna si è visto un aumento esponenziale dell'utilizzo di internet. Questo ha portato con sé la crescita esponenziale della mole di dati scambiati e immagazzinati. Se all'inizio bastavano semplici file per poter gestire l'immagazzinamento e la gestione dei dati, al giorno d'oggi servono complesse infrastrutture con notevoli capacità di calcolo. A compendio di ciò si sono rese necessarie tecniche in grado di estrarre informazioni dall'enorme quantità di dati perché questi non possono più essere analizzati a mano da un essere umano. Per questo motivo, anche se sono argomenti ben distinti, Big Data, Data Mining e Cloud Computing sono fortemente legati tra di loro.

Cloud Computing La crescente domanda di capacità di calcolo ha portato alla creazione di servizi di cloud computing per la gestione di task impegnativi che non possono essere svolti con le risorse di più largo uso. Il modello del Cloud Computing si basa su:

- Accesso a pool di calcolatori con capacità computazionali anche molto elevati
- On-demand
- Pay-per-use
- Fruizione veloce e affidabile

Il cloud computing prevede diversi tipi di servizi offerti:

- Software as a Service (SaaS)
- Platform as a Service (PaaS)
- Infrastructure as a Service (IaaS)

Nell'uso del Cloud Computing si hanno vantaggi in più aspetti come: scalabilità, non è più necessario investire nell'hardware, il pagamento è a fronte del solo utilizzo, gli aggiornamenti sono automatici, si possono avere politiche di disaster recovery e sicurezza e si ha più flessibilità di utilizzo in quanto un sistema cloud può essere utilizzato anche da dispositivi con bassa capacità computazionale.

Big Data In statistica e informatica, la locuzione inglese big data ("grandi [masse di] dati", o in italiano megadati) indica genericamente una raccolta di dati informativi così estesa in termini di volume, velocità e varietà da richiedere tecnologie e metodi analitici specifici per l'estrazione di valore o conoscenza. Il termine è utilizzato dunque in riferimento alla capacità (propria della scienza dei dati) di analizzare ovvero estrapolare e mettere in relazione un'enorme mole di dati eterogenei, strutturati e non strutturati (grazie a sofisticati metodi statistici e informatici di elaborazione), allo scopo di scoprire i legami tra fenomeni diversi (ad esempio correlazioni) e prevedere quelli futuri.

Volendo dare una tassonomia ad alto livello dei Big Data, questi possono essere divisi in:

- Strutturati: tabelle relazionali, spreadsheet, etc...
- Non strutturati: non hanno un modello di associazione tra i dati
- Semi-strutturati: hanno una forma lieve di struttura tra i dati, un esempio sono i dati XLM e JSON

Capitolo 6

Introduzione alla Business Intelligence ed alla Data Warehouse

I data warehouse sono il punto di inizio per i Big Data. Sono importanti perché i Big Data non possono esistere senza un data warehouse e viceversa. All'interno di un data warehouse si ha l'unione di diversi tipi di dato perché unendo più sorgenti diversi si hanno consigli sulle azioni da compiere più precisi.

6.1 Business intelligence

In generale la Business Intelligence comincia con i dati grezzi, li aggrega e li pulisce per poter dare supporto alle scelte strategiche aziendali, comunicando le giuste informazioni alle giuste persone nel momento giusto attraverso il giusto canale. Oltre a questo, possono essere usate tecniche di Machine Learning e Data Mining per avere una visione ancora più profonda di quale strategia sia meglio adottare.

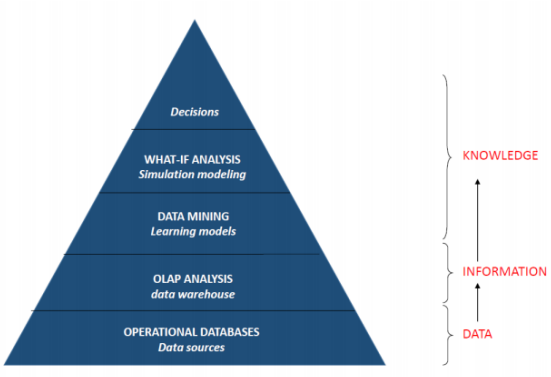


Figura 6.1: Struttura piramidale della business intelligence

Per supportare la business intelligence occorrono vari componenti come hardware ad hoc, infrastrutture di rete, database, data warehouse e software frontend adatto. Il tutto può essere organizzato secondo una chiara struttura.

Un passaggio fondamentale della business intelligence è il *Data Staging* (o ETL). In questo strato i dati sono copiati e messi nel data warehouse in modo da non interferire con i dati originali e riuscire a lavorare in real-time.

Sul data warehouse si possono fare diverse importanti operazioni:

- OLAP Analysis
- Reporting: stato zero del Data Mining, semplice aggregazione di dati (SELECT, GROUP BY)
- Data Mining
- Reconciliation: nel mondo reale i dati sono spesso sporchi. Quando vengono posti tutti insieme spesso si hanno condizioni di inconsistenza, è necessario risolvere queste situazioni

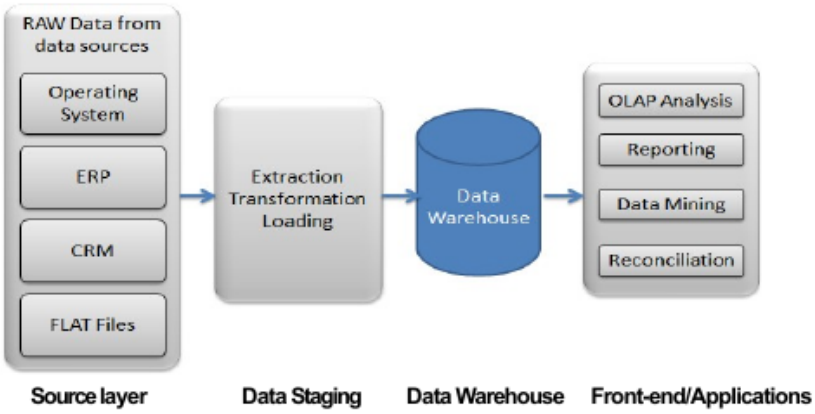


Figura 6.2: Un data warehouse è uno dei support principali alla business intelligence

6.2 Data Warehouse

Il data warehouse è lo strumento più importante per il supporto alle operazioni di business intelligence, in quanto l'incremento delle informazioni che un compagnia deve esaminare per formulare una strategia efficace implica la necessità di soluzioni più complesse di un semplice database relazionale. Questo componente non è altro che un repository di dati estremamente ottimizzato alle operazioni di decision-making ed ha diversi vantaggi:

- La possibilità di gestire lo storico dei dati
- La possibilità di formulare delle analisi multidimensionali. Le dimensioni non sono altro che le colonne di una tabella relazionale. Fare un'analisi multidimensionale significa vedere come le colonne interagiscono tra loro
- È basato su un modello semplificato che può essere facilmente appreso dagli utenti

Quindi un data warehouse è una collezione di dati al supporto dei processi di decision-making. Questo componente ha le seguenti caratteristiche:

- È subject-oriented, ovvero è focalizzato sugli aspetti di interesse dell'azienda come: clienti, fornitori, prodotti, vendite, ...
- È integrato e consistente: data warehouse integra i dati da diverse sorgenti eterogenee e fornisce una forma uniforme di questi
- Fornisce una visione dell'evoluzione dei dati e non è volatile. Viene tenuto traccia dei cambiamenti dei dati avvenuti all'interno del database, producendo un resoconto dei cambiamenti in senso temporale. Una volta inseriti, o dati all'interno di un DWH non sono mai modificati o sovrascritti, in quanto vi è la sola possibilità di lettura

6.2.1 Modello multidimensionale

Si ipotizzi di avere delle transazioni commerciali (data, prodotto e negozio). Ciò significa che ogni transazione è l'intersezione dei tre attributi. Sui dati raccolti poi possono essere svolte operazioni di raggruppamento come ad esempio i prodotti di una determinata categoria o il profitto dato da un singolo cliente con le sue transazioni. Questi aspetti possono essere visualizzati in un grafico tridimensionale. La figura che si denota è un cubo formato da tanti piccoli cubetti: ogni cubetto è un raggruppamento di dati transazionali (vedi figura 6.3).

6.2.2 OLTP vs OLAP

Possiamo riassumere qui le differenze tra la metodologia OLTP e OLAP:

- OLTP

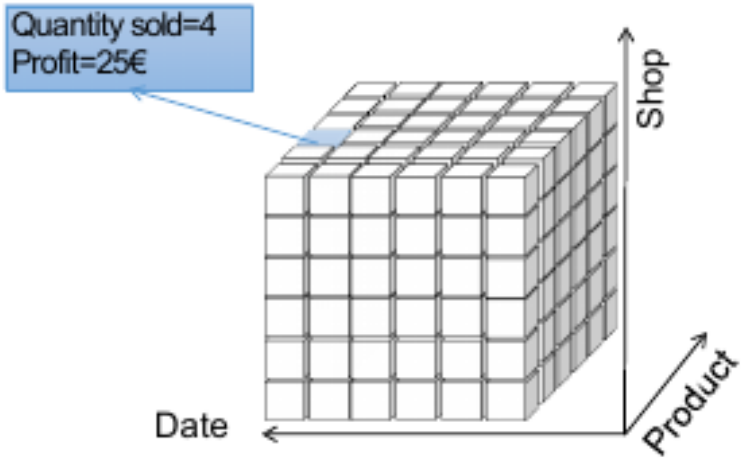


Figura 6.3: Ogni cubetto contiene gruppi di quattro prodotti

  - Sistema interattivo per il l'elaborazione dei dati basato su transazioni
  - Ogni transazione legge e scrive un piccolo numero di record da tabelle caratterizzate da semplici relazioni
  - È dotato di un core congelato all'interno dei programmi applicativi. Non vi sono necessità di cambiare le query effettuate

- OLAP
  - Sistema interattivo per l'elaborazione dei dati utile in caso di analisi multidimensionali
  - Ogni query coinvolge un enorme numero di record necessarie ad elaborare un insieme di valori numerici per ottenere un'indicazione circa le performance del business
  - Il carico di lavoro cambia col tempo

6.2.3 Database relazionali vs data warehouse

Riassumiamo qui le differenze principali tra un database relazionale ed un data warehouse:

Caratteristiche	**Database**	**Data Warehouse**
Utenti	Migliaia	Centinaia
Tipo di lavoro	Transazioni prestabilite	Analisi specifiche
Accesso	Centinaia di record (rw)	Milioni di record (r)
Scopo	Dipende dall'applicazione	Supporto alle decisioni
Dati	Dettagliati, sia numerici che testuali	Dati riassuntivi, principalmente numerici
Qualità	In termini di integrabilità	In termini di coerenza
Temporalità	Dati correnti	Dati correnti e visione storica
Aggiornamenti	Continui	Periodici
Modello	Normalizzato	Non normalizzato, multidimensionale
Ottimizzazioni	Per l'accesso OLTP	Per l'accesso OLAP

6.2.4 Data Mart

Il concetto di Data Mart è simile a quello di view nei database relazionali. In sintesi, è una vista dal data warehouse per uno specifico scopo, tralasciando i dati non di interesse. Esistono tuttavia delle differenze tra Data Mart e una vista del modello relazionale:

- Nel modello relazionale la vista viene creata e archiviata. All'atto della query, la query è combinata con la definizione della vista ed eseguita
- Nel Data Mart viene copiato ogni sottoinsieme di dati utili per una singola attività. Quando viene progettato il data mart occorre definire anche la frequenza di aggiornamento dei dati

6.3 OLAP

L'analisi OLAP consente agli utenti di navigare interattivamente le informazioni contenute all'interno di un data warehouse (o data mart). Tipicamente i dati vengono analizzati a diversi livelli di aggregazione utilizzando una serie di operatori OLAP (ognuno dei quali può lanciare una o più query).

Durante una sessione OLAP l'utente può esplorare il modello multidimensionale e scegliere il prossimo operatore in base ai risultati ottenuti dall'applicazione del precedente. In questo modo, l'utente genera un percorso di navigazione che corrisponde ad un'analisi del processo.

L'idea chiave di OLAP è la flessibilità e la capacità di cambiare vista molto velocemente. Il punto di partenza per ogni operatore è il data cube, che contiene i dati raggruppati inizialmente. Gli operatori OLAP principali sono:

- Roll-up: consente di raggruppare ulteriormente i dati contenuti all'interno del cubo (idealmente diminuendo il numero di cubetto che lo compongono, ma aumentandone la dimensione)
- Drill-down: è l'operazione duale rispetto al roll-up. Riduce il livello di aggregazione aggiungendo dettagli
- Slice-and-dice: riduce il numero di dimensioni del cubo dopo aver impostato una delle dimensioni ad uno specifico valore. L'operazione di dicing riduce il set di dati che sono stati analizzati da una criterio di selezione
- Pivot: comporta un cambio di layout, rappresenta una visione differente dell'insieme dei dati
- Drill-across: permette di creare collegamenti tra i concetti tra cubi correlati, per confrontarli
- Drill-through: trasforma la vista aggregata multidimensionale in una tabella operazionale

6.4 Extraction, Transformation and Loading (ETL)

La lunga pipeline nella figura sotto, che permette di passare dai dati ai Data Cube, inizia con il processo di ETL. Il processo di ETL estrae, integra e pulisce i dati dai dati operazionali per alimentare lo strato data warehouse. Si distinguono le seguenti fasi:

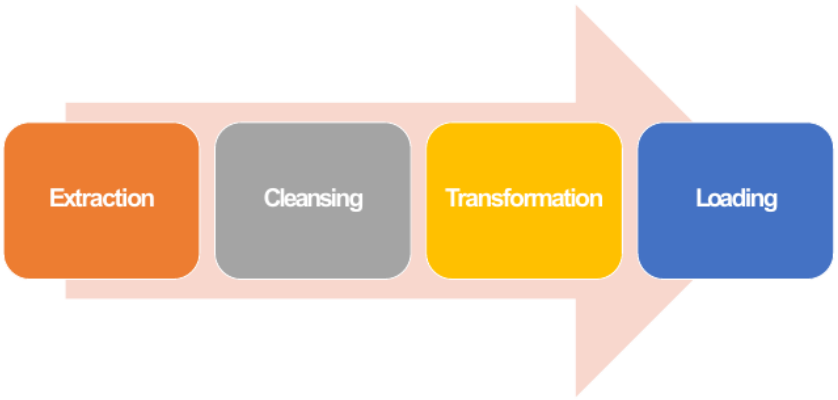


Figura 6.4: Processo ETL nel suo completo

- Estrazione: è il processo di estrazione dei dati. Si possono avere dati strutturati (ad es. dei csv o dei database relazionali) e dati non strutturati (ad.es provenienti da social network). Il processo di estrazione può essere statico, usato per popolare la DWH per la prima volta, o incrementale, usato per aggiornarne il contenuto con regolarità. Nel secondo caso ogni aggiornamento è associato ad un timestamp che lo identifica ed è attivato da dei trigger specifici

- Cleansing: è il processo che permette di pulire i dati da eventuali errori o incongruenze, specialmente se si tratta di dati inseriti a mano dall'uomo. Questo processo in alcuni casi può rivelarsi molto complicato. I principali motivi di errore sono:
  - Dati duplicati
  - Dati mancanti
  - Uso dei campi di compilazione errato
  - Valori impossibili o palesemente errati
  - Valori inconsistenti per un singola entità a causa di errori di battitura o dell'utilizzo di convenzioni diverse nella nomenclatura

  Per risolvere i casi di inconsistenza si possono usare diverse tecniche:
  - Tecniche basate su dizionari: sono utilizzate per verificare la correttezza dell'attributo basandosi su tabelle di ricerca e dizionari per cercare abbreviazioni e sinonimi. Si possono applicare se il dominio è noto e limitato. Queste tecniche sono adatte per risolvere problemi come errori di battitura e discrepanze di formato
  - Unione approssimata: questa tecnica è usata quando è necessario deve unire i dati provenienti da fonti diverse, ma non si ha una chiave comune per identificare le tuple corrispondenti. Si possono usare quindi:
    * Join approssimato: l'unione verrà eseguita sulla base degli attributi comuni. Questi attributi non sono identificatori per il cliente, quindi non sono soggetti a procedure di controllo per garantire vincoli di integrità e assenza di errori di immissione
    * Funzioni di similitudine: viene usato il concetto di similitudine per identificare diverse istanze delle stesse informazioni. Gli strumenti ETL hanno al loro interno molti algoritmi che ispezionano i dati e cercano di capire quale trasformazione e strategia per migliorare i dati sia la migliore per un dato caso specifico. Quindi si può usare le funzioni di affinità per calcolare la somiglianza tra due parole. Se la somiglianza è maggiore / minore di una determinata soglia, le due parole sono le stesse e le righe possono essere unite
  - Algoritmi ad-hoc: algoritmi personalizzati basati su regole aziendali specifiche (es. riguardo al contesto finanziario deve sempre essere verificata la seguente regola, $profitto = entrate - spese$)
- Trasformazione: durante questa fase i dati vengono trasformati affinché rispettino il formato desiderato (ad es. tutte le spese nella medesima valuta ecc.). Per risolvere questi problemi si possono usare le seguenti tecniche:
  - Conversione: si cambia il dato in accordo con le regole correnti
  - Arricchimento: combinazione di più attributi per creare nuove informazioni
  - Separazione/concatenazione
- Caricamento: è il processo finale che serve a riempire il data warehouse con dati corretti e puliti. Per farlo ci sono due modi differenti:
  - Refresh: ovvero il data warehouse è completamente riscritto
  - Update: vengono modificati solo i campi che realmente cambiano lasciando invariati gli altri

6.5 Architettura di un data warehouse

L'architettura di un data warehouse ha determinati requisiti:

- Separazione
- Scalabilità
- Estensibilità
- Sicurezza
- Facilità di amministrazione

Per raggiungere tutti questi punti si possono usare architetture differenti, ognuna con i suoi pro e contro.

Single-Layer Architecture L'obbiettivo di questa architettura è minimizzare la quantità di dati immagazzinati, rimuovendo i dati ridondanti. Lo strato sorgente è l'unico strato fisicamente disponibile. Il data warehouse è implementato come una view multidimensionale dei dati operazionali creata da uno specifico middleware.

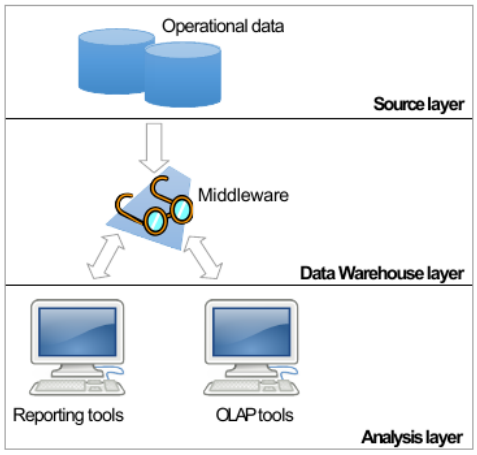


Figura 6.5: Single Layer Architecture

Two-Layer Architetture In questo caso c'è separazione fisica fra sorgente dei dati e data warehouse. Lo strato sorgente dei dati include un insieme eterogeneo di sorgenti. Lo strato di Data Staging immagazzina i dati estratti dalla sorgente per rimuovere inconsistenze ed errori e integra i dati provenienti dalle sorgenti eterogenee in un unico schema comune. Questo strato include le procedure ETL. Lo stato di data warehouse è dove sono immagazzinate le informazioni e sono distribuite secondo un modello logico centralizzato, una repository. Questo strato fornisce un punto di accesso comune per interrogare i dati o creare Data Mart. Lo stato di analisi infine è uno strato che permette un approccio più user-friendly per interrogare i dati, creare report, dashboard, etc ...

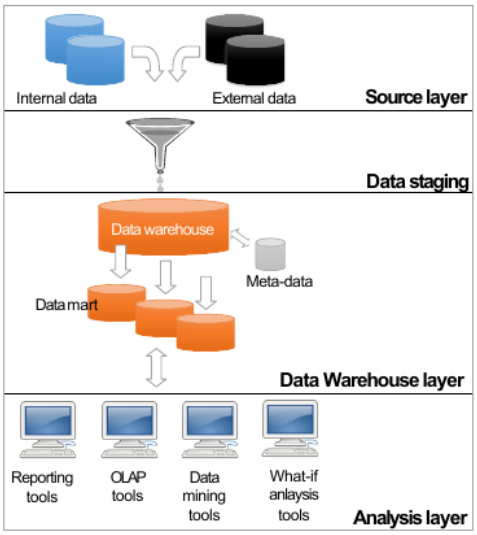


Figura 6.6: Two Layer Architecture

Tree-Layer Architetture Nella base questa architettura condivide lo schema dell'architettura a due strati, ma aggiunge uno strato chiamato Reconciled Layer. Questo strato materializza i dati operazionali ottenuti dopo le fasi di integrazione e pulizia. Il risultato è un dato integro, consistenti, corretti, aggiornati e dettagliati. Lo strato aggiunto crea un riferimento comune al modello dei dati per l'intera azienda.

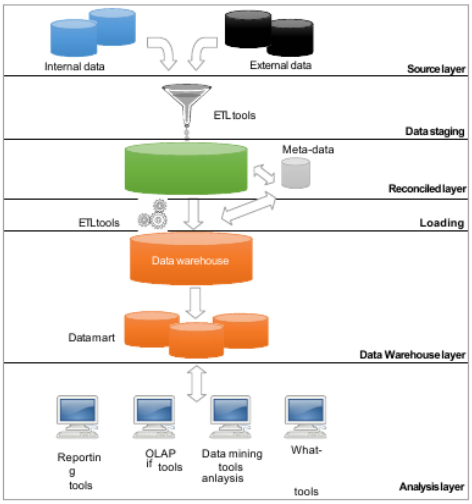


Figura 6.7: Three Layer Architecture

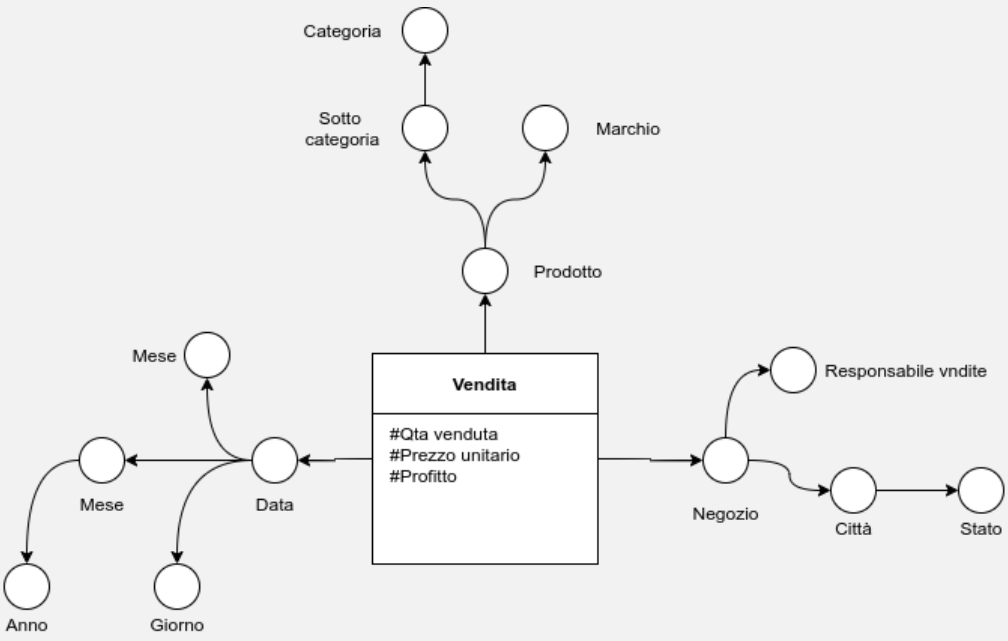
6.6 Modellazione concettuale DWH: The Dimensional Fact Model (DFM)

Con The Dimensional Fact Model (DFM) si intende un modello concettuale creato a supporto della progettazione del data mart. Questo modello concettuale ha come obbiettivi:

- Fornire supporto al design concettuale
- Creare un ambiente in cui gli utenti possano interagire in maniera comoda con la DWH
- Favorire la comunicazione tra utenti e designer per formalizzare i requisiti richiesti
- Costruire una piattaforma stabile per il design logico
- Fornire una chiara e efficace documentazione del design della DWH

Introduciamo un esempio di DFM

Esempio 6.1



Il Dimensional Fact Model (DFM) ha i seguenti concetti base:

- Fatto: i concetti rilevanti nel processo decisionale. Tipicamente modella un insieme di eventi in avvenimento all'interno dell'azienda. Alcuni esempi sono le vendite, gli acquisti o gli ordini
- Misura: è una proprietà numerica che descrive un fatto in maniera quantitativa relativamente ad un qualche aspetto analizzabile. Alcuni esempi sono la quantità venduta e lo sconto applicato
- Dimensione: è una proprietà di un fatto dotata di un dominio finito e descrive un punto di coordinazione per l'analisi. Alcuni esempi sono le date, i prodotti, i negozi
- Attributi dimensionali: sono altre dimensioni o attributi in grado di descrivere una dimensione. Alcuni esempi sono le categorie di prodotti, i mesi
- Gerarchia: è un albero i cui nodi sono attributi dimensionali ed i cui archi modellano relazioni molti ad uno tra gli elementi di una coppia di attributi dimensionali. La radice dell'albero è sempre una dimensione
- Evento primario: è una particolare occorrenza di un fatto, identificata da una ennupla contenente un valore per ognuna delle dimensioni. Ad ogni evento primario è associato un valore per ognuna delle misure

Esempio 6.2

In riferimento all'esempio precedente si ha il seguente evento primario

Data	Negozio	Prodotto	Qta venduta	Guadagno	Prezzo unitario
01/03/2015	Negozio Centrale	Caffe	54	100	5

- Evento secondario: dato un insieme di attributi dimensionali, ogni ennupla di valori identifica un evento secondario che aggrega tutti gli eventi primari corrispondenti. Ogni evento secondario è associato ad un valore per ogni misura che riassume tutti i valori della misura stessa in corrispondenza dell'evento primario

Esempio 6.3

In riferimento all'esempio precedente si hanno i seguenti eventi primari:

Data	Negozio	Prodotto	Qta venduta	Guadagno
01/03/2015	Negozio Centrale	Caffe	20	60
01/03/2015	Negozio Centrale	Latte	25	50
02/03/2015	Negozio Centrale	Pane	40	70
10/03/2015	Negozio Centrale	Vino	15	150

Da cui deriva il seguente evento secondario:

Data	Negozio	Prodotto	Qta venduta	Guadagno
Marzo 2015	Negozio Centrale	Cibo e bevande	100	330

L'aggregazione richiede di definire un operatore adatto a comporre i valori delle misure che caratterizzano gli eventi primari in valori da abbinare a ciascun evento secondario.

- Misure di flusso: fanno riferimento ad un intervallo temporale al termine del quale esse sono valutate cumulativamente (ad es. quantità di beni venduta)
- Misure di livello: sono valutate in momenti particolari (ad es. numero di prodotti in inventario)

- Misure unitarie: sono valutate in momenti particolari ma sono espresse in termini relativi (ad es. il prezzo unitario)

Una misura viene chiamata additiva lungo una dimensione quando è possibile utilizzare l'operatore SOMMA per aggregare i suoi valori lungo la gerarchia delle dimensioni. Se questo non è il caso, viene chiamata non additivo. Una misura non additiva non è aggregabile quando non è possibile utilizzare nessun operatore di aggregazione.

	Gerarchie temporali	Gerarchie non temporali
Misure di flusso	Somma, Media, Minimo, Massimo	Somma, Media, Minimo, Massimo
Misure di livello	Media, Minimo, Massimo	Somma, Media, Minimo, Massimo
Misure unitarie	Media, Minimo, Massimo	Media, Minimo, Massimo

Gli operatori di aggregazione seguono una precisa classificazione:

- Distributivi: calcolano aggregando aggregati parziali (somma, min, max)
- Algebrici: necessitano l'utilizzo di informazioni addizionali sotto forma di un numero finito di misure di supporto per calcolare correttamente gli aggregati da aggregati parziali (es. Media)
- Olistici: calcolano aggregando aggregati parziali solamente con un numero infinito di misure di supporto (es. Rango)

Queste proprietà sono utili quando si utilizza gli operatori Roll-up e Drill-down perché se si pre-calcola la SOMMA nel cubo, quando si applica l'operatore Roll-up poi semplicemente si calcola la somma delle somme.

6.6.1 DFM avanzate

All'interno dei grafici è poi possibile usare vari altri tipi di connessioni:

- Attributi descrittivi, usati per fornire ulteriori informazioni ma non usati per le fasi di aggregazione
- Attributi inter-dimensionali, ovvero attributi dimensionali il valore è definito dalla combinazione di due o più attributi dimensionali (che possono anche appartenere a gerarchie differenti)
- Convergenza, caso in cui due o più archi della stessa gerarchia terminano nel medesimo attributo dimensionale
- Gerarchie condivise
- Archi multipli, usati per rappresentare relazioni molti a molti
- Archi opzionali, usati per modellare scenari che non sono la norma (ad es. la taglia di un prodotto è valida solamente per i vestiti e non per il tonno)
- Gerarchie incomplete, ovvero quelle gerarchie all'interno delle quali uno o più attributi sono mancanti in alcune istanze
- Gerarchie ricorsive, usate per rappresentare relazioni genitore-figlio tra i livelli

6.6.2 Progettazione logica

La fase di progettazione logica definisce le strutture dati, ovvero l'insieme di tabelle e le relazioni tra tabelle. Questo perché si vogliono rappresentare i Data Mart secondo il modello logico selezionato in modo da ottimizzare le prestazioni mettendo a punto queste strutture. Quindi, partendo dallo schema concettuale si vuole definire lo schema logico di un Data Mart. Esistono tre passaggi principali per implementare uno schema logico in un DBMS relazionale:

- Tradurre i fatti in schemi logici: star o snowflake
- Viste materializzanti: insieme di viste secondarie che aggregano i dati della vista primari per migliorarne le prestazioni durante le query
- Frammentazione dei fatti: sia orizzontale che verticale

Lo schema Star è caratterizzato da fact tables e dimension tables. Una fact table contiene tutte le misure e gli attributi descrittivi legati a un fatto. Per ogni dimensione viene creata una tabella dimensionale che include tutti gli attributi della gerarchia:

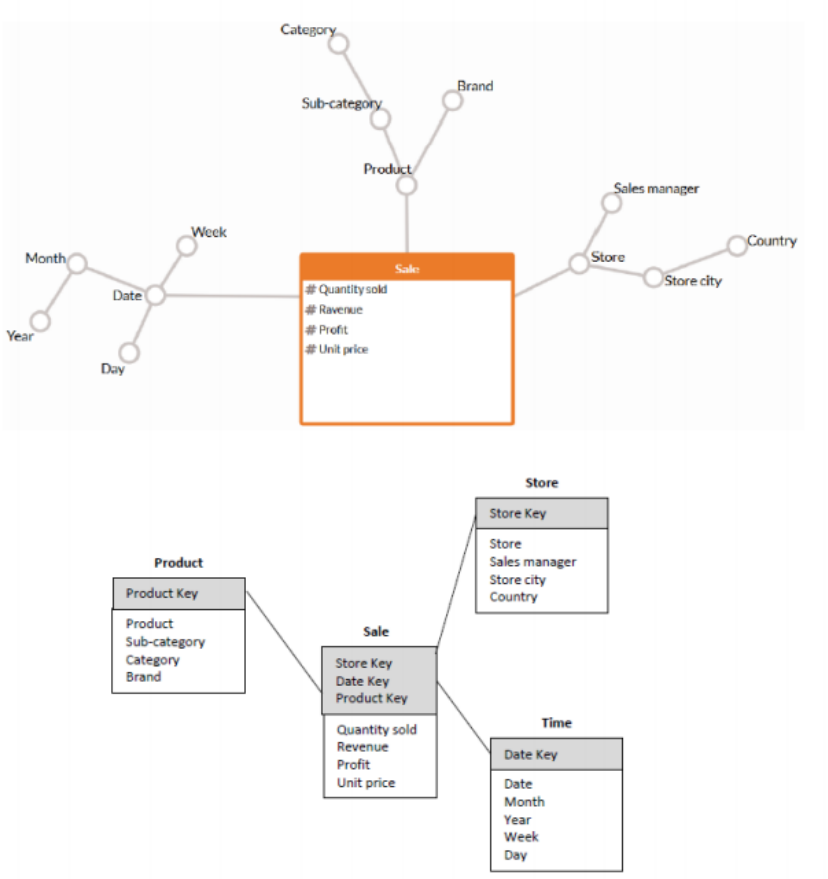


Figura 6.8: Schema a stella

Una schema snowflake è uno schema Star che varia con la dimensione delle tabelle parzialmente normalizzate.

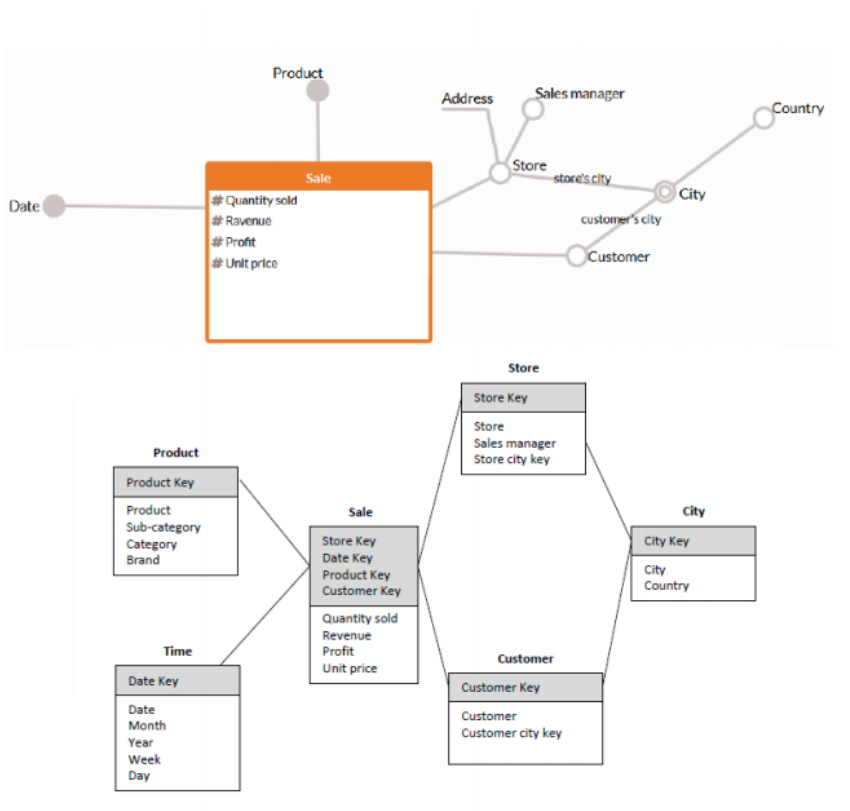


Figura 6.9: Schema a fiocco di neve