Ringraziamenti

Il primo ringraziamento va alla mia famiglia, a mio padre, mia madre ed i miei fratelli, senza il cui sostegno tutto questo non sarebbe stato possibile.

La mia riconoscenza va al prof. D'Acierno, per la professionalità e la cordialità con le quali ha diretto il presente lavoro.

Desidero rivolgere un pensiero a tutte le persone che ho avuto il piacere di incontrare durante questo lungo cammino, i colleghi universitari, gli amici extrauniversitari, i compagni di appartamento. Ognuno di loro ha avuto un ruolo…elencarli tutti sarebbe impossibile…

Una carezza a Sonia, che mi ha supportato e sopportato…

Nicola Calandriello

13/02/2006

A mamma Caterina…

"Unum scire, Nulla scire"

Socrate.

SOMMARIO

PREFAZIONE

Il presente lavoro si colloca nel campo del Data Mining, il settore disciplinare delle tecnologie per basi di dati che affronta il problema della scoperta e costruzione di conoscenza utile per il supporto alle attività decisionali critiche nell'attività di Business di aziende medio - grandi a partire dal proprio patrimonio informativo. Le metodologie adottate dalla disciplina intendono fornire strumenti per il controllo e l'analisi dei dati, che spesso "sommergono" le organizzazioni che collezionano dati in formato elettronico, al fine di dedurre da essi elementi preziosi che si possano aggiungere al "know how" già acquisito.

Cap. I

IL PROCESSO DI SCOPERTA DELLA CONOSCENZA NELLE BASI DI DATI

1.1 Introduzione

Oggigiorno, in svariati campi, i dati digitali sono collezionati e immagazzinati a passi vertiginosi. Nasce quindi un urgente bisogno di nuove teorie computazionali e strumenti applicativi che assistano gli uomini ad estrarre informazioni utili, ovvero conoscenza, da collezioni massive di dati. E' stato infatti stimato che il volume di dati mondiale raddoppia ogni 20 mesi ed il numero di database diffusi cresce anche più velocemente (Fig.1).

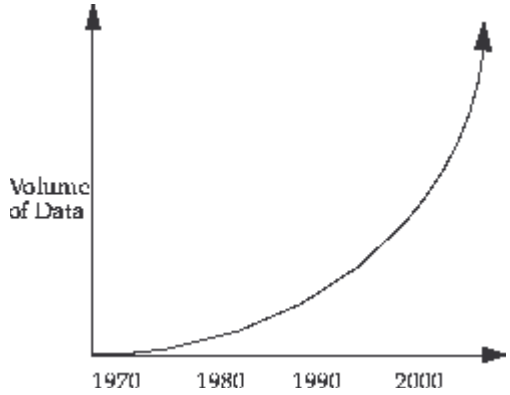


Figura 1: Andamento qualitativo della collezione di dati elettronici nell'ultimo trentennio.

Ad un alto livello di astrazione il campo del KDD (Knowledge Discovery in Databases) può essere visto come il processo di sviluppo di metodi e tecniche che possano "dare senso" ai dati. Il problema di fondo di tale processo è quello di rendere possibile un mapping fra i dati di basso livello, tipicamente troppo voluminosi da essere umanamente intelligibili, ed altre forme di rappresentazione più compatte, come un report, più astratte, come una approssimazione del modello che ha generato i dati, o più utili, come un modello predittivo che aiuti nella previsione di casi futuri. Nel cuore di questo processo si colloca l'applicazione dei metodi tipici del Data Mining (DM) per la scoperta e l'estrazione dei modelli nascosti, dall'inglese patterns, nei dati.

La metodologia tradizionale per trasformare dati in conoscenza consiste nell'analisi manuale e nell'interpretazione di analisti esperti che abbiano grande confidenza con il problema da

trattare. Ad esempio è comune nella Sanità che gli specialisti analizzino andamenti e mutazioni dei dati con cadenza trimestrale e riportino poi le osservazioni fatte al direttivo gestionale dell'organizzazione, questo report diventa la base sulla quale operare scelte strategiche e pianificazioni future. In un campo applicativo totalmente diverso i geologi possono, avendo a disposizione immagini satellitari di pianeti od asteroidi, localizzare e catalogare oggetti d'interesse come i crateri di impatto.

Quindi sia in Scienza, Finanza, Commercio, Sanità nell'approccio classico è l'analista che funge da interfaccia fra i dati e gli utilizzatori dei risultati.

Si intende subito che questa indagine manuale sui dati risulta essere lenta, faticosa, dispendiosa ed altamente soggettiva, nasce quindi la necessità di automatizzare, almeno parzialmente, il processo specialmente laddove la crescita esponenziale della mole di dati renderebbe l'approccio classico completamente impraticabile.

Infatti i database crescono in due direzioni: il numero N di record ed oggetti ed il numero d di campi ed attributi di un oggetto. Basi di dati con $N = 10^9$ oggetti sono sempre più comuni nelle scienze astronomiche ed il numero di campi può facilmente raggiungere l'ordine delle centinaia nelle applicazioni di diagnosi mediche.

La possibilità di scalare le capacità umane di analisi su grandi moli di dati è una questione sia economica che scientifica. Le aziende osservano i dati per ottenere vantaggi competitivi sul mercato, incrementare l'efficienza delle proprie attività ed offrire servizi di qualità ai clienti. Siccome i calcolatori hanno reso possibile la collezione di molti più dati di quanto un essere umano possa assimilarne, è naturale pensare allo sviluppo di metodologie che possano ricavare da essi strutture e modelli significativi per il supporto alle decisioni. Quindi il processo KDD fa fronte ad uno dei problemi principali dell'era digitale: il sovraccarico di dati.

1.2 L'evoluzione della disciplina

Storicamente il concetto di ritrovamento di pattern utili nei dati ha avuto una vasta nomenclatura: data mining, knowledge extraction, information discovery, information harvesting, data archaeology, data pattern processing ecc. Il termine Data Mining è stato in origine adoperato dagli statistici, dagli analisti di dati e dalle comunità MIS (Management Information System) mentre la frase Knowledge Discovery in Databases è stata coniata nel primo seminario che si tenne nel 1989 (Piatetsky – Shapiro 1991) per enfatizzare che è la conoscenza il prodotto finale di un processo di scoperta guidato dai dati (data – driven). Nel

seguito ha acquisito popolarità anche nel campo dell'intelligenza artificiale e del machine learning.

Una delle definizioni più accettate e famose in letteratura è quella fornita da William J. Frawley, Gregory Piatetsky-Shapiro e Christopher J. Matheus :
"Il Data Mining, o Knowlegde Discovery in Databases come è anche conosciuto, è l'estrazione non banale di informazioni implicite, in precedenza nascoste e potenzialmente utili dai dati. Questo comprende una varietà di differenti approcci tecnici come il raggruppamento, riassunto dei dati, regole di classificazione, legami di dipendenza, analisi dei cambiamenti e rivelazione di anomalie."

Un'altra definizione più esemplificativa è quella di Marcel Holshemier e Arno Siebes (1994):
"Il Data Mining è la ricerca delle relazioni e dei pattern globali che esistono nei grandi database ma che sono "nascosti" nel vasto ammontare di dati come una relazione fra i pazienti e le loro diagnosi mediche. Queste relazioni rappresentano una conoscenza preziosa circa il database e, se il database è uno specchio fedele, il mondo da esso registrato."

L'analogia con il processo di estrazione mineraria è descritto come:
"L'uso di una varietà di tecniche per identificare "pepite" di informazioni o conoscenza per il decision-making in volumi di dati ed estrarle in modo che possano essere usate in contesti come il supporto alle decisioni, previsioni, pronostici e stime" (Guida utente di Clementine, SPSS Inc.)

Ad ogni modo il KDD si riferisce al processo complessivo di scoperta di informazioni utili dai dati mentre il DM riguarda un passo particolare del processo. Il DM è l'applicazione di algoritmi specifici al fine di estrarre i patterns dai dati. La distinzione fra il procedimento globale di KDD e il passo di DM come attività interna al processo è una questione fondamentale. Gli ulteriori passi nel processo come la preparazione dei dati, la selezione delle sorgenti, la pulizia dei dati, l'incorporamento di conoscenze anteriori ed appropriate, l'opportuna interpretazione dei risultati dell'attività di estrazione sono essenziali per assicurare che conoscenza utile sia ricavata dai dati grezzi. L'applicazione cieca dei metodi di DM, giustamente criticata e classificata come *data dredging* in letteratura, è una attività pericolosa oltre che non proficua, che porta facilmente alla scoperta di patterns invalidi o privi

di significato; ciò può risultare dannoso al tentativo di modellizzazione del fenomeno sotto osservazione e confondere chi lo sta studiando.

1.3 La natura interdisciplinare del KDD

Il processo di KDD si è evoluto, e continua ad evolvere, dalla originaria concezione che lo vedeva come intersezione di campi di ricerca come l'apprendimento delle macchine, riconoscimento di strutture (pattern recognition), basi di dati, statistica, intelligenza artificiale, acquisizione di conoscenza per sistemi esperti. L'obiettivo comune è estrarre conoscenza di alto livello da dati di basso livello nel contesto di estesi volumi di dati (Fig.2).

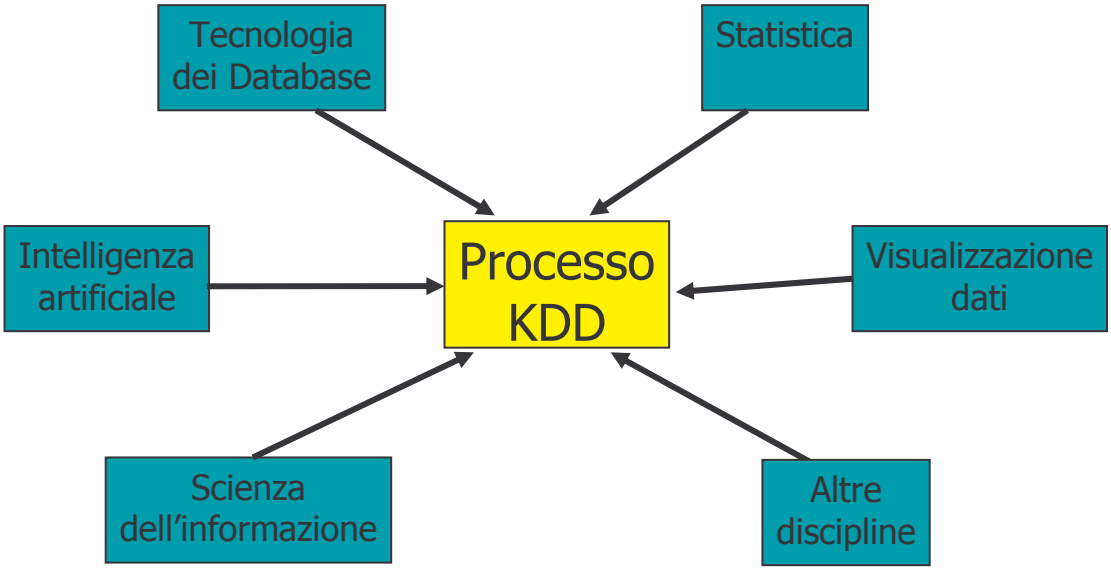


Figura 2: I contributi delle discipline al processo KDD.

L'attività di DM fa un grande affidamento su ben note tecniche che provengono dal machine learning, dal pattern recognition, dalla statistica ecc. Allora viene spontaneo chiedersi in cosa differiscono questi campi dal sottoprocesso di DM. La risposta sta nel fatto che questi campi di studio forniscono al DM metodi e tecniche propri mentre il processo di KDD focalizza la sua attenzione sulle dinamiche complessive ivi inclusi i metodi di memorizzazione dei dati (data storage), i metodi di accesso, come gli algoritmi possono essere scalati verso moli di dati crescenti senza inficiarne le prestazioni, come i risultati possono essere interpretati e visualizzati, come l'interazione uomo-macchina (MMI) può essere modellata e supportata.
Quindi il KDD deve essere visto come un'attività multidisciplinare che racchiude tecniche che vanno oltre lo scopo delle particolari discipline da cui provengono.
In questo contesto vi sono chiare opportunità applicative per l'intelligenza artificiale, ad esempio l'uso delle reti neurali; esse costituiscono un potente strumento di modellizzazione ma risultano di difficile comprensione se comparate con gli alberi decisionali.

La statistica in particolare lavora in grande sinergia con il processo KDD ed ha con esso molto in comune. La scoperta di conoscenza nei dati è sostanzialmente uno sforzo statistico. La statistica fornice un linguaggio ed un quadro di lavoro (framework) per quantificare il grado di incertezza quando si tenta di inferire modelli generali da campioni particolari in una popolazione complessiva.

Il termine Data Mining ha avuto una connotazione negativa dal punto di vista delle scienze statistiche sin dagli anni 60, quando furono introdotte per la prima volta le tecniche di analisi dei dati basate su calcolatori. La faccenda si presentò sulla base dell'osservazione che se si cerca abbastanza in profondità su qualsiasi set di dati (anche generato casualmente) si possono trovare patterns statisticamente rilevanti ma che, in realtà, non lo sono praticamente. Chiaramente, questa questione è di fondamentale importanza per il processo KDD. Molti progressi sono stati fatti negli anni recenti per comprendere queste problematiche in statistica ma la necessità ineluttabile è che per applicare proficuamente le metodologie proposte dal DM bisogna capire come farlo nel modo corretto, conoscendo la natura del problema che si vuole analizzare e il dominio applicativo in cui essa si colloca.

Il KDD quindi mira ad automatizzare, finchè è possibile, il processo di analisi dei dati e supportare la selezione statistica delle ipotesi.

La forza motrice che sta dietro tale processo è, evidentemente, il campo delle basi di dati, in quanto la maggior parte degli algoritmi di DM suppone che tutti i dati di input risiedano in memoria centrale e non prestano attenzione a come decadono le prestazioni se è possibile accedere a solo una parte dei dati. Conseguentemente, tecniche efficienti di accesso ai dati, operazioni di ordinamento e raggruppamento durante l'accesso, interrogazioni ottimizzate costituiscono le basi per scalare gli algoritmi verso set di dati più ampi.

1.4 Il ruolo dei sistemi OLAP

Un campo applicativo in stretta relazione con il Data Mining è quello dei cosiddetti sistemi *OLAP* (On Line Analitical Processing), nomenclatura proposta da Codd (1993) che li definì come:

"la sintesi, l'analisi e l'unificazione dinamica di grandi volumi di dati".

La componente principale è una speciale base di dati chiamata *Data Warehouse* (magazzino dei dati) che svolge il ruolo di sorgente dei dati da sottoporre ad elaborazioni. Questi sistemi fungono da interfaccia tra gli utenti, tipicamente analisti, e la enorme mole di dati contenuta nel magazzino offrendo, grazie ad opportuni motori, strumenti e metodi di manipolazione dei dati che si basano sulla rappresentazione *multidimensionale* (cfr. par. 1.4.3).

Questo magazzino ha una struttura peculiare che è diversa da una base di dati tradizionale, ovvero:

- E' una base di dati integrata: i dati provengono da sorgenti diverse sia per natura (struttura delle relazioni, tabelle, campi, record) che per localizzazione sul territorio (architetture federate interconnesse da reti a banda larga).
- Archivia informazioni storico - temporali: a differenza degli OLTP che mantengono solo lo stato corrente delle transazioni, i DW sono interessati alla totale evoluzione temporale delle informazioni.
- Contiene dati aggregati: i dati collezionati sono organizzati sulla base di precise coordinate (tempo, locazione geografica, tipologia di prodotto ecc.)
- E' autonoma: Essa viene tenuta separata sia dalle sorgenti informative (un'aggregazione in linea sarebbe improponibile) che dagli applicativi che la sfruttano (che richiedono strutture e metodi di accesso specifici).
- E' fuori linea: Il caricamento dei dati viene effettuato periodicamente per non influire negativamente sulle prestazioni di un OLTP. Sovente è permesso un lieve disallineamento fra i dati che non risultano quindi perfettamente aggiornati.

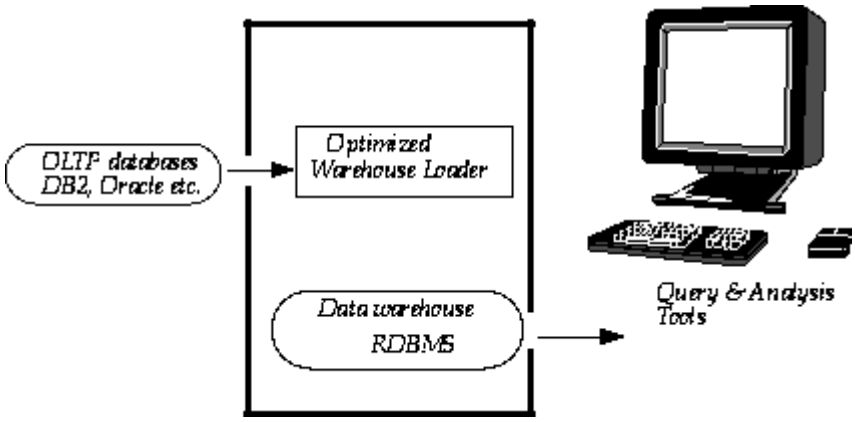


Figura 3: Input e output di un magazzino di dati.

1.4.1 Architettura di un Data Warehouse

Diamo ora uno sguardo ad un possibile modello architetturale di massima sul quale può essere progettato un magazzino dati. Lo schema è riportato in Fig. 4.

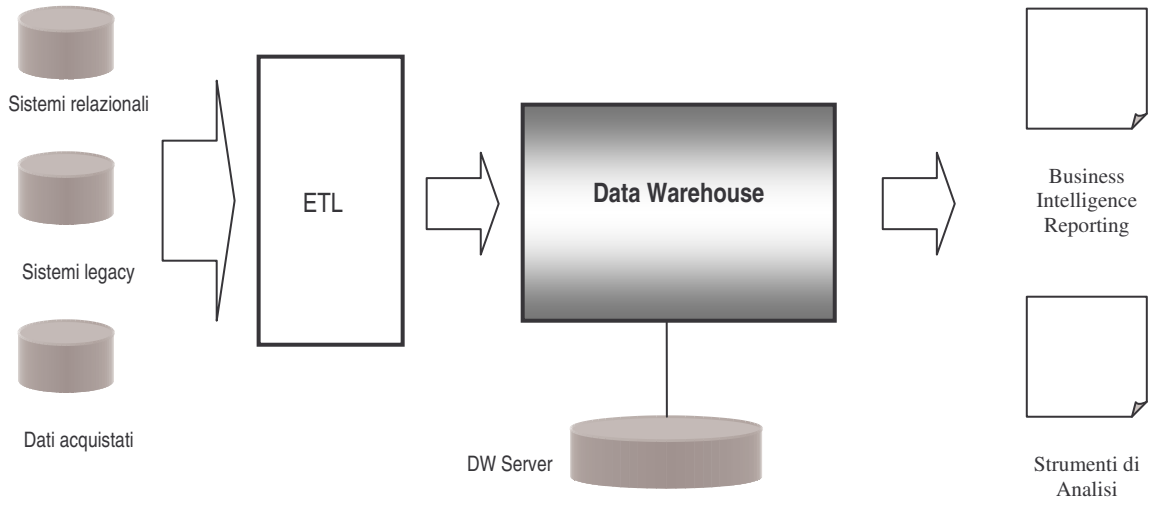


Figura 4: Architettura generica di un sistema DW.

- **Sorgenti informative**: Si è detto che possono essere, e per lo più lo sono, di varia natura: sistemi transazionali aziendali, sistemi ereditati (legacy system) od anche fonti non gestite tramite DBMS come archivi o semplici file.
- **DW Server**: E' l'unità dedicata alla gestione e manutenzione del magazzino, memorizza fisicamente i dati e permette di effettuare operazioni complesse di interrogazione ed operazioni speciali come il roll-up e il drill-down.
- **Sistema ETL** (Extraction, Transfomation and Loading)
    - L'estrazione dei dati dalle sorgenti avviene in maniera incrementale: il modulo che se ne occupa rileva solo le modifiche come inserzioni o cancellazioni.
    - La pulizia dei dati avviene controllando sulle singole sorgenti evidenti errori o inconsistenze (come ad esempio la gestione dei valori mancanti).Ciò è fondamentale ai fini della qualità dei dati.
    - La fase di trasformazione serve ad omogeneizzare le strutture dati provenienti da ambienti diversi, vengono quindi effettuate conversioni di scala, associazioni e formattazioni di campi, denormalizzazioni delle tabelle, ordinamento dei valori ecc.
    - Il caricamento e l'acquisizione dei dati viene effettuato principalmente in maniera incrementale ovvero propagando periodicamente le modifiche subite dalle sorgenti (refresh).Si preferisce relegare questa attività nelle ore notturne o nei fine settimana organizzandola in lotti (batch).

- **Strumenti di analisi**: Tali strumenti permettono di analizzare i dati sfruttando i servizi offerti dal DW offrendo interfacce amichevoli e di facile comprensione per l'utente. In questo settore si collocano gli strumenti di Data Mining.
- **Metadati**: rappresentano un archivio dove sono contenute tutte le informazioni sui dati contenuti, le operazioni possibili su di essi, i concetti secondo i quali essi sono organizzati e secondo quali il DW è stato progettato.

1.4.2 Piattaforme Hardware

Non è raro il caso in cui una grande azienda odierna collezioni dati nell'ordine delle centinaia di Gigabyte o Terabyte in poche settimane, ciò dovuto alla grande crescita dell'e-business e reso possibile dall'evoluzione di tecnologie come, per citarne una, la comunicazione senza fili (wireless).

Ad esempio un grosso venditore on line di prodotti, che registra ogni click di milioni di clienti per conoscerne le abitudini di shopping, ha necessità di analizzare profondamente milioni di righe di dati per attuare campagne promozionali mirate ed efficaci.

D'altro canto molti DW di oggi sono basati su vecchie architetture general purpose che sono inadatte a gestire enormi moli di dati cosicchè molte applicazioni di BI (Business Intelligence) sono sottoutilizzate o addirittura abbandonate.

Un DW di una grossa azienda (EDW, Enterprise Data Warehouse) utilizzerà verosimilmente qualche sorta di architettura multiprocessore visto che le informazioni da gestire sono troppe per una singola CPU o per un singolo backplane.

Le due forme principali di questo approccio sono il *Symmetrical Multiprocessing (SMP)* ed il *Multiple Parallel Processing (MMP).*

L'SMP consiste in uno svariato numero di processori, ognuno con la propria memoria cache. Essi costituiscono un pool di risorse computazionali sul quale è uniformemente distribuito il carico dei threads in modo che nessuna unità risulti sovraccarica mentre ve ne è un'altra inattiva. Tale compito è svolto dal Sistema Operativo. Le risorse come la memoria ed i canali di I/O sono condivise ed accessibili ad ogni singola CPU. Questa applicazione trova il suo punto di forza quando vi è una quantità rilevante di dati in memoria centrale sui quali operano più task ma è limitativa se grandi moli di dati devono essere prelevate da altre unità, come la memoria di massa.

In generale, l'MMP costiste in un notevole numero di processori lascamente accoppiati. Ogni CPU ha la propria memoria RAM, la propria scheda madre, la propria memoria di massa ed il proprio sistema operativo. Questo approccio di non condivisione delle risorse, tipico

dell'MMP puro, permette una scalabilità semplice e lineare, con l'obiettivo che il software possa trarne vantaggio ed essere parallelizzabile.
L'MMP puro è abbastanza raro in pratica in quanto i costi di espansione del sistema (memoria addizionale e dispositivi di I/O) possono essere non trascurabili e le difficoltà di amministrazione e gestioni di unità indipendenti possono essere notevoli.
Tipicamente un'architettura MMP viene implementata connettendo virtualmente cluster di SMP che spesso condividono risorse di I/O.L'intento è quello di preservare le performance e la scalabilità dell'MMP riducendo i costi di realizzazione e manutenzione.

1.4.2.1 Varianti per il Data Warehousing

La maggior parte dei grandi sistemi DW reali sono basati sulle seguenti varianti architetturali delle forme SMP ed MPP.

1. ***SMP su larga scala (large scale SMP)*:** un piccolo sistema SMP non è capace di processare interrogazioni complesse. Daltronde, sistemi su larga scala con processori supplementari ed una memoria di taglio consistente rendono disponibile una grande risorsa di calcolo e spesso sistemi del genere sono impiegati per il data warehousing.

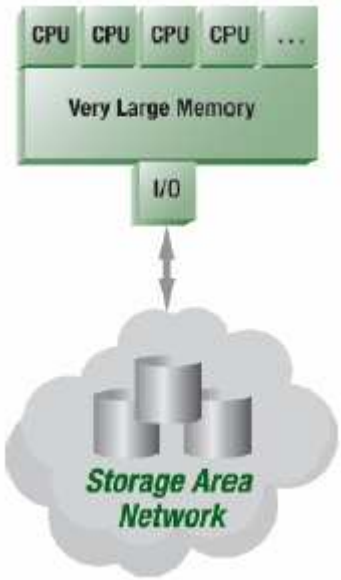


Figura 5: schema di un unità SMP.

Come si può vedere dalla Fig. 5 dozzine di processori condividono memoria centrale e memoria di massa. Quando il volume di dati cresce tali sistemi tendono ad espandere memoria, backplane e capacità di I/O ma quando i processori necessitano di accedere a grandi moli di dati il bus di memoria diventa facilmente un collo di bottiglia. Siccome la banda passante del bus è limitata, aumentare il numero di CPU o le dimensioni della

memoria diventa futile. Il bus di memoria può essere congestionato al crescere del volume di dati che viene richiesto alla SAN (Storage Area Network) per processare una interrogazione articolata.

Quindi, un tradizionale svantaggio di questo approccio è la scalabilità meno che lineare oltre al declino delle prestazioni alla crescita del sistema.

2. ***MPP on clustered SMP*:** Lo schema rappresentato in fig. 6 consiste in piccoli cluster SMP che operano in parallelo condividendo una SAN e la relativa struttura di gestione.

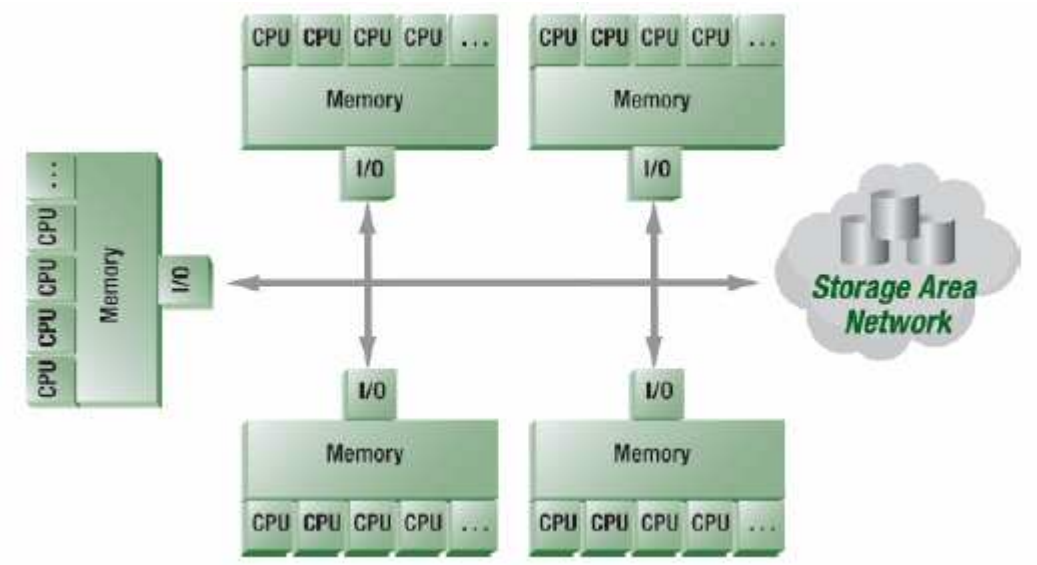


Figura 6: schema di nodi SMP.

Ogni CPU condivide la memoria RAM con i suoi vicini e l'accesso al bus con gli altri cluster.

Molti vendors propongono soluzioni basate su questa architettura, come IBM DB2 Integrated Cluster Environment (ICE). Il collo di bottiglia diventa la possibilità di accesso al bus comune per raggiungere la SAN.

3. ***Shared Nothing MPP*:** In questo tipo di architettura, coppie processore – disco operano in parallelo dividendo il carico di lavoro su un vasto insieme di dati.Ogni CPU dialoga con il disco associato per ottenere i dati grezzi e svolgere le operazioni. Un processore è invece dedicato a collezionare i risultati intermedi ed all'assemblaggio del responso delle interrogazioni da ritornare all'applicazione richiedente.

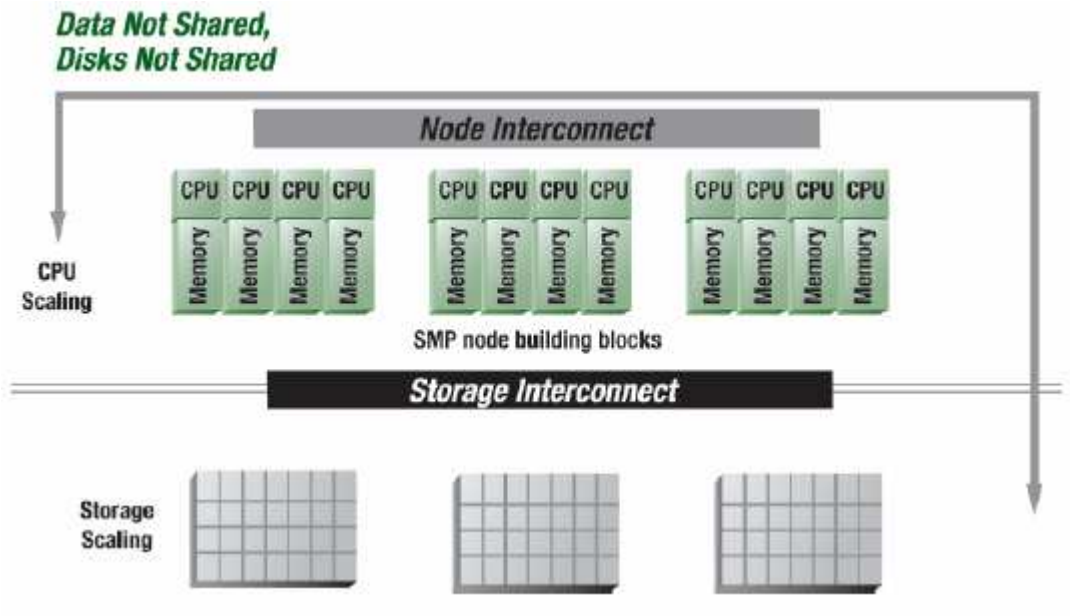


Figura 7: Architettura a condivisione nulla.

Evitando così la contesa delle riscorse fra i nodi MPP questa soluzione permette di scalare verso dimensioni nell'ordine dei terabyte. Una delle sue debolezze però è la necessità di muovere grandi quantità di dati fra disco e CPU. Infatti, mentre le coppie processore – disco operano esse condividono una interconnessione proprietaria e dedicata che, se intasata, influisce rapidamente sul tempo di risposta. In un tipico scenario i dati vengono letti a porzioni di 64K blocchi quando ne sono sufficienti 1K per eseguire la query, il processore, quindi, si deve far carico di filtrare tutti i dati inutili; ciò si tramuta in carico ulteriore per il sistema. Questo è il classico problema dei sistemi MPP, i backplane interni, i bus e le connessioni di I/O trovano serie difficoltà a gestire volumi di dati considerevoli. L'incapacità delle velocità di trasferimento dati di andare di pari passo con la crescente disponibilità di memoria di massa inibisce le possibilità di scalatura, come ammette la stessa Teradata che adotta questa metodologia[1].

4. ***Separate data – Shared Storage*:** Il supporto di memorizzazione di massa è condiviso ma partizionato logicamente secondo i processori, affinchè essi non si contendano gli stessi dati. Questi sistemi sono più economici di un'architettura "shared nothing" dove ogni unità di elaborazione ha il proprio dispositivo fisico di memorizzazione. Per contro, tale soluzione soffre degli svantaggi tipici di un'architettura MPP e cioè il

[1] "With capacity growing more quickly than disk bandwidth, the bandwidth per GB of storage capacity has actually decreased by 50%. Given this trend, it is challenging to take advantage of abundant storage capacity while maintaining required performance levels." – Ron Yellin, Director of Storage Product Management, Teradata (Teradata Magazine Online, Vol. 4, No. 1, 2004).

sovraccarico delle linee di comunicazione con il dispositivo allorché una query complessa necessita di dati residenti su memoria di massa.

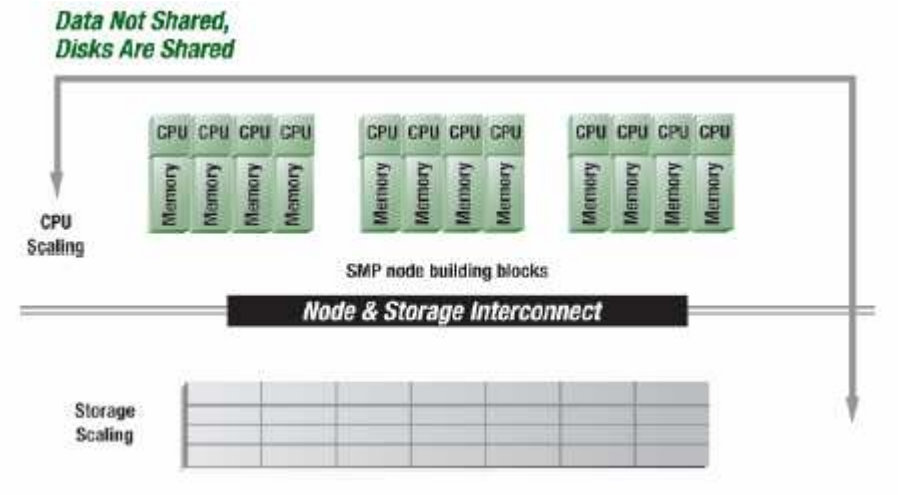


Figura 8: Architettura a separazione logica di dati.

IBM ha sviluppato un soluzione data warehouse secondo questo schema. Essa è indipendente dall'hardware ed ha come nucleo l'RDBMS (Relational DBMS) DB2 e può essere supportata da una grande varietà di server ed opzioni di memorizzazione. Chiaramente tale indipendenza ha un prezzo, che si riflette nelle delicate scelte da compiere nella vasta gamma di prodotti in commercio allorché si voglia realizzare un sistema del genere. Ciò necessita da parte dei clienti la comprensione delle dipendenze di I/O del DW da architetture di memorizzazione completamente differenti.

Un DBMS maturo come DB2 è dotato di opportuni tool atti a regolare ed ottimizzare i diversi sistemi componenti l'architettura per eseguire query articolate con buone performance. E' previsto, quindi, un pesante lavoro da parte degli amministratori, di complessità crescente all'aggiunzione di nuovi nodi SMP.

Un modo per far fronte ai vincoli di performance è l'uso dell'indicizzazione per limitare i dati esaminati in una query ma la combinazione di tecniche di partizionamento e indicizzazione su cluster separati rende la configurazione ed il caricamento dati molto complesso.

Il pesante carico di configurazione e gestione , in questi casi, spaventa l'utente e richiede spesso l'intervento di servizi professionali specializzati ad alto costo, alterando così il rapporto costi / benefici.

5. ***Shared data – Shared storage*:** in questo schema, processori multipli operano in parallelo accedendo agli stessi dati residenti su un dispositivo di massa comune. Viene usato un lock manager per prevenire accessi simultanei agli stessi dati da parte di

processi diversi. Tale accesso a dati condivisi è coordinato via messaggi tra il lock manager e i processi in esecuzione.

Oracle ha costruito una soluzione secondo questo approccio usando il suo RDBMS RAC 10g (Real Application Cluster).

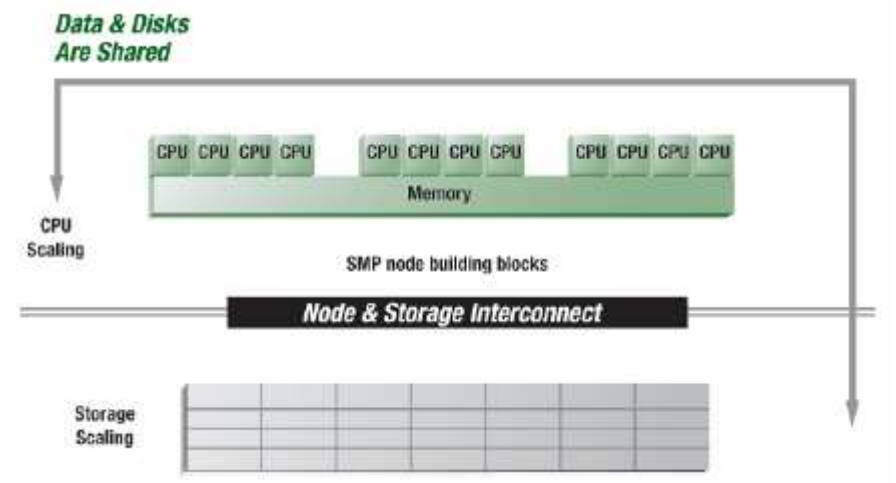


Figura 9: Architettura a condivisione fisica di dati.

Questo approccio evita ai DBA's di doversi preoccupare di strategie di partizionamento dell'unità di massa. Tuttavia, come nei casi precedenti, sorgono problemi quando molti dati devono essere passati da memoria di massa a processori e la natura condivisa dei dati può mettere un tetto alle performance ed alle possibilità di scalatura del sistema.

6. ***Blade servers:*** Una interessante soluzione è rappresentata dai cosiddetti blade servers che forniscono grande potenza di calcolo in unità molto compatte, dette schede. Ogni blade ha la propria schiera di processori, memoria e canali di I/O, da cui la dicitura "a server on a card".

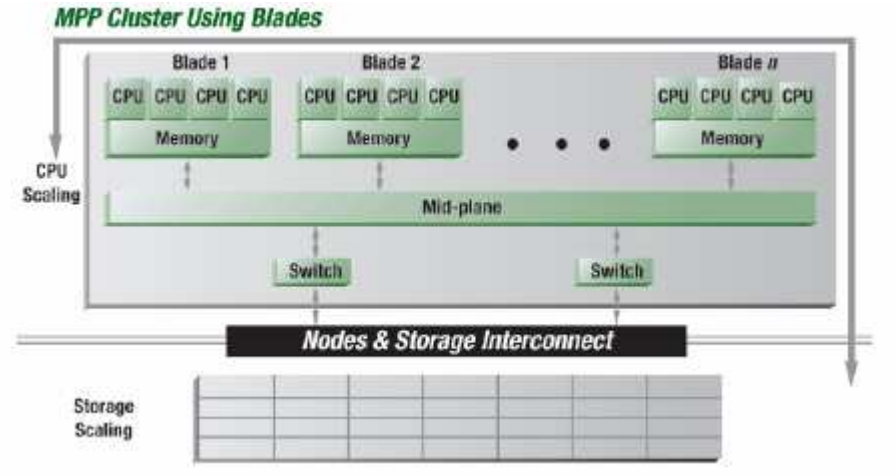


Figura 10: Architettura a server su scheda.

Svariate schede blade sono montate su un singolo telaio, detto anche armadio, e condividono area di memorizzazione, rete ed alimentazione elettrica. Le caratteristiche dell'armadio sono essenzialmente di tipo meccanico, come il numero di slot, le dimensioni di ingombro, le modalità di fissaggio. Il risultato è un'architettura integrata e compatta ad alta performance, con un framework di gestione comune che fornisce il controllo come su un unico sistema virtuale.

Molti produttori offrono soluzioni per DW basate su tecnologia blade. In un tipico scenario, ogni blade svolge il ruolo di un cluster SMP, con propri processori ed una memoria centrale condivisa, all'interno di una matrice di blade che operano in parallelo. Questo la rende molto simile all'architettura "MPP on clustered SMP" descritta in precedenza. Nello schema mostrato in fig. 10 ogni scheda comunica con i dispositivi di memoria di massa tramite un midplane di sistema che costituisce un percorso comune a tutte le unità inserite negli zoccoli del telaio. Per questo, tale approccio soffre del tipico problema delle architetture DW: grandi moli di dati devono essere trasferite su di un sentiero comune.

Una questione da sottolineare è il fatto che tale tipologia di sistema sfrutta a pieno le sue potenzialità se il software per il data warehousing è sviluppato ad hoc. Infatti, trasferendo semplicemente su di esso il software ottimizzato, ad esempio per un sistema legacy OLTP, ciò non conferirà nessun valore aggiunto al sistema, addirittura in alcuni casi i colli di bottiglia preesistenti potrebbero essere aggravati.

In sostanza, la prima generazione attuale di blade servers fornisce una piattaforma di elaborazione general – purpose con fattori di forma (dimensioni e forma dei drives) e profili di costo migliori rispetto alle tradizionali implementazioni SMP / MPP clusterizzate o non, ma per beneficiare pienamente dei suoi vantaggi, essa deve evolvere ed ottimizzare verso applicazioni specifiche.

Un esempio nel segmento industriale è la *"Google Search Appliance"*, una soluzione blade server "custom - designed" per permettere attività di ricerca di contenuti a velocità molto elevate.

1.4.3 Il modello multidimensionale

La rappresentazione multidimensionale dei dati è il concetto secondo il quale vengono organizzati logicamente i dati nel progetto di un sistema OLAP. E' il modello dei dati fondamentale in molti sistemi OLAP oggi in commercio. Esso consiste in tre entità principali: il *fatto*, la *dimensione* e la *misura*.

Il fatto è un concetto di rilevanza ai fini dei processi di business sul quale si intende svolgere attività di analisi; ad es. nel caso di un'impresa di vendita al dettaglio un fatto potrebbero essere le vendite mensili effettuate. Tali informazioni vengono collezionate ed organizzate in una tabella dei fatti (fact table).

La dimensione è il particolare punto di vista dal quale si vuole effettuare l'analisi per il supporto alle decisioni. Dimensioni tipiche sono il luogo e il tempo, ma considerando ancora il caso della vendita al dettaglio una dimensione può essere l'articolo venduto. I particolari valori di una dimensione sono detti *membri*. Si può pensare alle dimensioni come a delle caratteristiche che rispondono a domande del tipo: "chi ? cosa ? dove ? quando ? come ?".

La misura, invece, è una proprietà di un fatto che si vuole analizzare, tipicamente un attributo numerico (ad es. il numero di articoli venduti in un mese).

Le dimensioni vengono organizzate in livelli di aggregazione, un esempio è riportato in fig. 11.

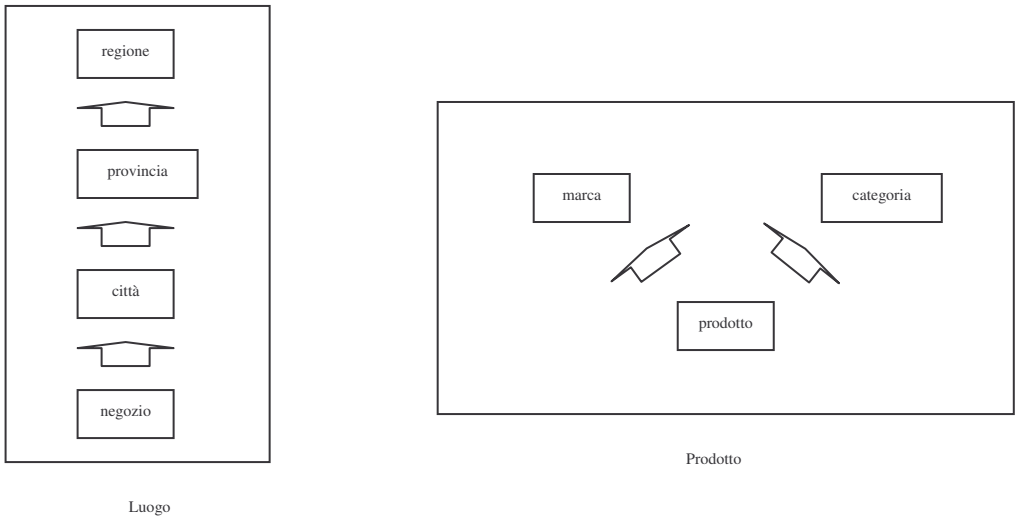


Figura 11: esempi di gerarchie di dimensioni.

Le frecce stanno ad indicare i versi di aggregazione lungo una stessa dimensione; si noti che possono effettuarsi aggregazioni differenti sullo stesso livello della struttura ma non aggregare membri sullo stesso livello. Come si evince dalla figura non è possibile aggregare le categorie per marca e viceversa.

Tutte le informazioni finora descritte vengono raggruppate in entità dette *cubi n – dimensionali*, delle strutture che hanno una intuitiva rappresentazione grafica.

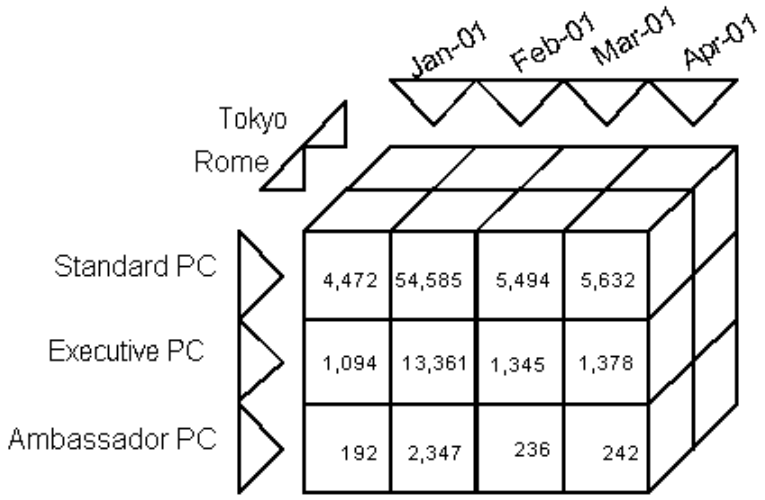


Figura 12: struttura di un cubo 3-D.

Essi sono costituiti da particelle atomiche dette celle, dove sono mantenuti i valori delle misure e lungo le dimensioni "geometriche" di questi cubi si distendono le dimensioni dei fatti.

Tali composizioni si prestano operazioni di analisi che si tramutano in operazioni sui cubi, le più note sono:

- *Slice and dice:* E' la selezione di un sottoinsieme di celle di un cubo. Figurativamente la si può immaginare come la creazione di un cubo di dimensioni minori.
- *Roll up:* Ne esistono due varianti. La prima è l'aggregazione dei dati del cubo *in seguito* all'applicazione di una funzione aggregativa (come la somma) lungo la struttura gerarchica di una dimensione.

  La seconda modalità è la completa eliminazione della dimensione *seguita* da una funzione aggregativa, come ad es. la trasformazione di un cubo tridimensionale in una semplice tabella bidimensionale.
- *Drill down:* E' l'operazione inversa del roll up. Si scende in profondità nella gerarchia della dimensione prescelta, aggiungendo dettagli alle misure contenute nelle celle. Ad es. nella dimensione tempo il passaggio da trimestre a mese.

1.4.4 Realizzazione di un Data Warehouse

Vi sono due approcci generali alternativi per la realizzazione di un DW. La prima soluzione consiste nella applicazione della ben nota tecnica relazionale, sovente utilizzata anche per i DBMS. Si applica cioè al modello dei dati il concetto di E – R (Entity - Relationship), i dati

vengono convogliati in tabelle e le operazioni tradotte in opportune istruzioni SQL. Tali sistemi sono detti ROLAP (Relational OLAP).

Il secondo approccio è più radicale ovvero i dati vengono già memorizzati in strutture multidimensionali che spesso sono proprietarie. Tali sistemi sono i MOLAP (Multidimensional OLAP). Esistono chiaramente anche soluzioni ibride, come ROLAP che permettono la memorizzazione multidimensionale di alcuni data mart (cfr. par. 1.4.6).

In una realizzazione ROLAP, un data mart portebbe essere realizzato secondo il seguente schema relazionale, detto "schema a stella":

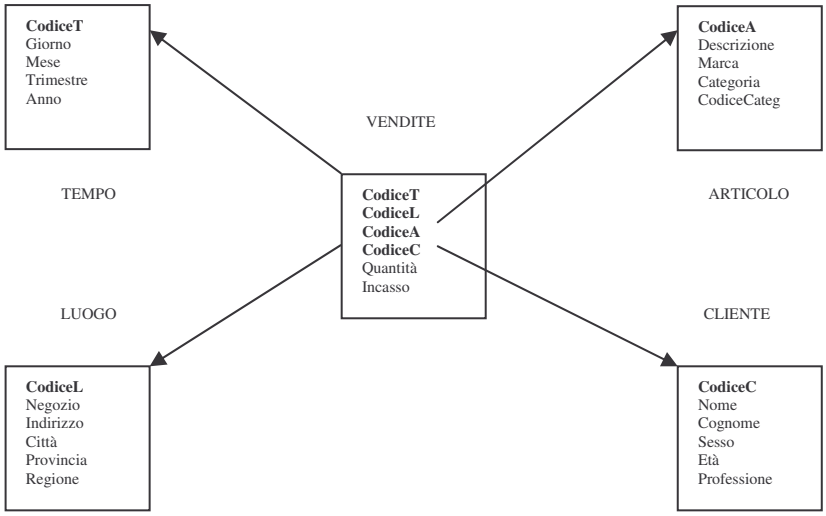


Figura 13: schema a stella.

Ha una struttura molto semplice:

- Una relazione principale, detta *tabella dei fatti*, che memorizza i fatti. Ha una chiave multipla composta da attributi che fanno riferimento alle tabelle dimensione mentre gli altri attributi sono le misure del fatto. Tipicamente è normalizzata.
- Varie *tabelle dimensione* che memorizzano le dimensioni dell'analisi. Esse hanno una chiave semplice e gli attributi rappresentano la struttura gerarchica della dimensione; tipicamente non sono normalizzate.
- *Vincoli di integrità referenziale* fra tabella dei fatti e tabella delle dimensioni.

Qualora si voglia normalizzare, anche in parte, uno schema stella si ottiene uno schema come quello in fig. 14, detto "schema a fiocco di neve" (snowflake):

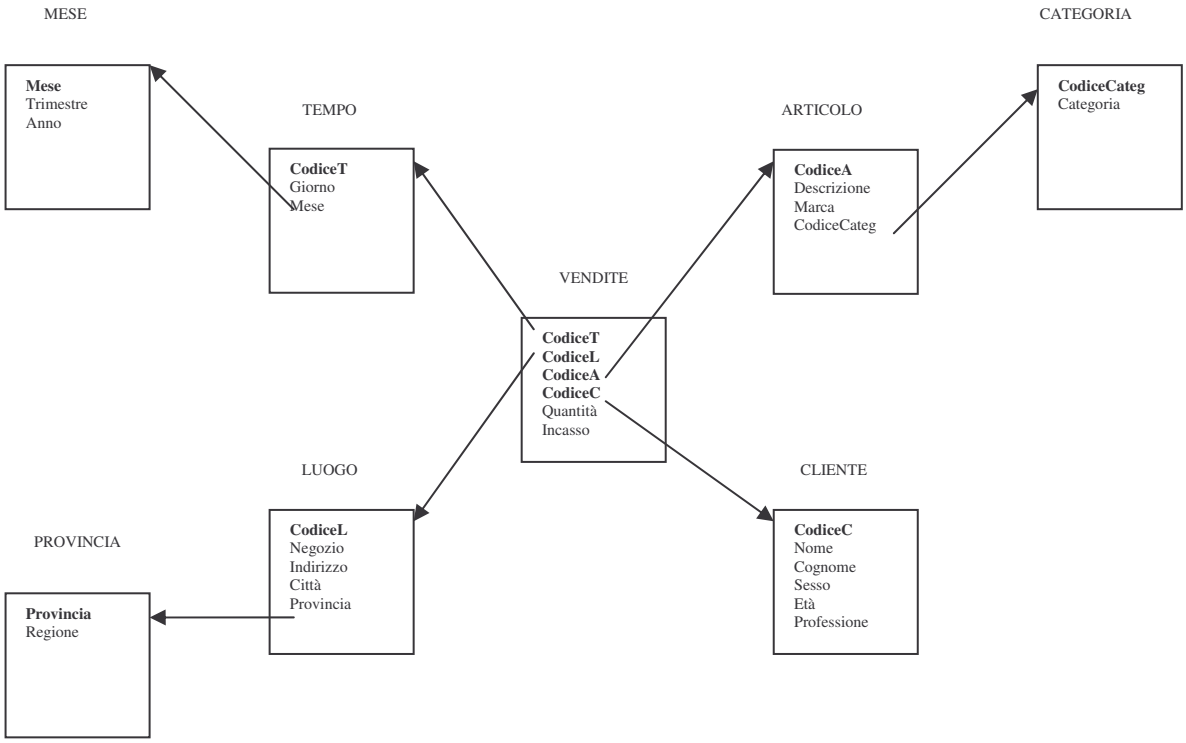


Figura 14: schema "snowflake".

In genere è sconsigliato effettuare normalizzazioni troppo spinte perché il beneficio che si ottiene nel risparmio di spazio non compensa il degrado delle prestazioni che si può ingenerare. C'è da dire, infatti, che la tabella più ingombrante, di solito, è la tabella dei fatti, riducendo quindi le tabelle delle dimensioni lo spazio totale guadagnato è limitato.

1.4.5 Progettazione di un Data Warehouse

Di enorme importanza nella progettazione e sviluppo di un sistema OLAP è l'acquisizione di adeguate metodologie atte ad ingegnerizzare l'intero processo. Tuttoggi si assiste ad un rapido sviluppo di sistemi di analisi in linea, sovente accompagnato da strumenti efficienti ma basati su osservazioni empiriche e dettate dalle contingenze e non da strategie di applicabilità generale.

Come risultato si può avere una qualità non soddisfacente del sistema realizzato e lo sfondamento del tetto di budget previsto dovuto ai costi di manutenzioni straordinarie.

Una metodologia può essere inquadrata come descritto in fig. 15, essa si compone di 4 fasi principali e di varie sottofasi, con i relativi dati di input ed output.

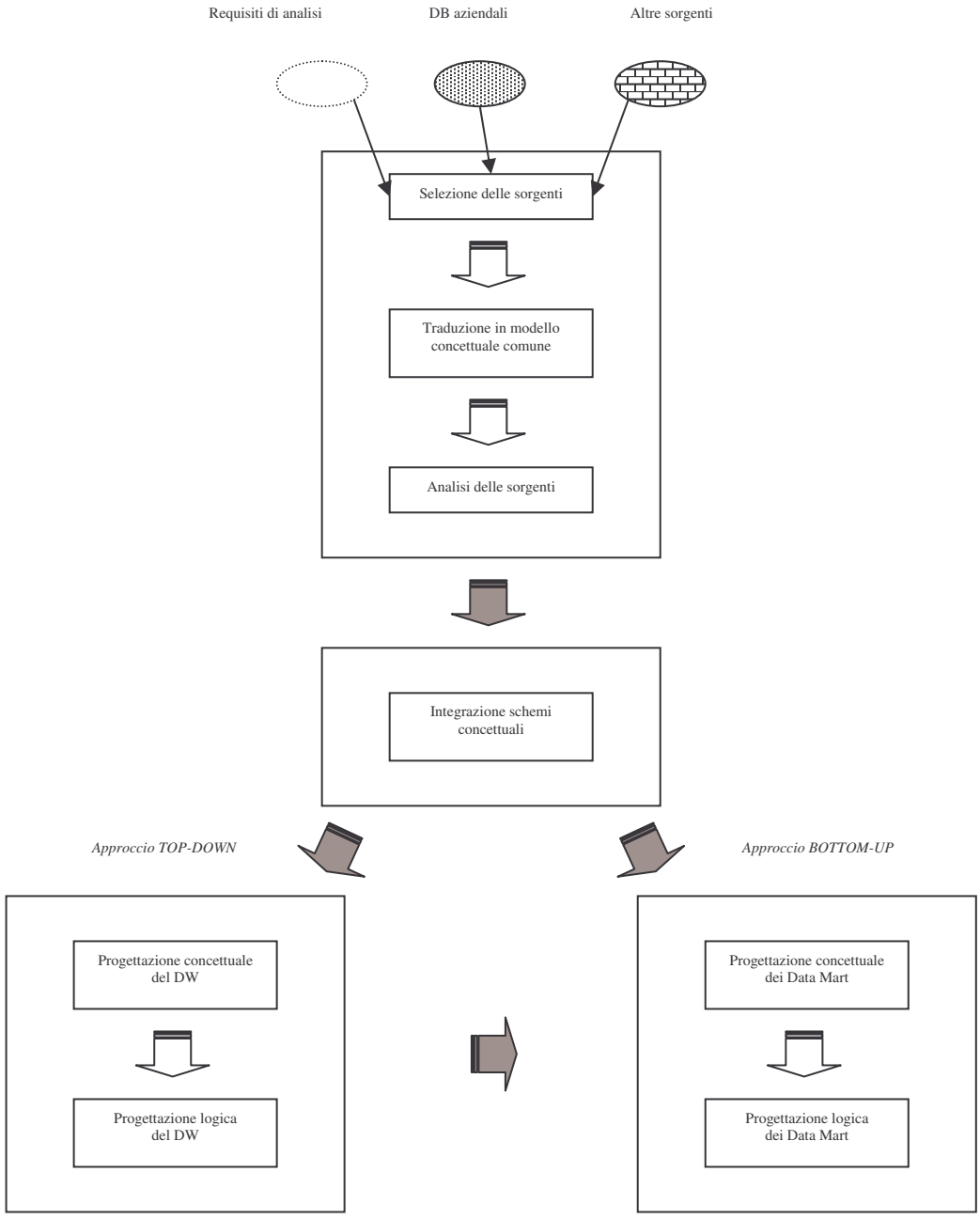


Figura 15: Sviluppo delle fasi di progetto di un DW.

Dati di ingresso. Si richiede la specifica delle tipologie di dati di ingresso. I requisiti sono la descrizione delle esigenze di analisi, spesso compiuta in linguaggio semistrutturato od anche naturale.

Gli schemi delle sorgenti informative descrivono la struttura e la composizione delle basi di dati operazionali dell'organizzazione, nonché di sistemi legacy e relativa documentazione di supporto.

Se il magazzino è alimentato da sorgenti esterne (come dati statistici o economici forniti da enti istituzionali) allora è necessaria anche la corrispondente descrizione.

Analisi dei dati di ingresso. La prima fase consiste nell'analisi dei dati di input alla luce dei requisiti, al fine di selezionare le sorgenti e determinarne le priorità. Infatti, alcune delle fonti possono risultare trascurabili ai fini dell'analisi che si intende svolgere; allo stesso tempo si possono stabilire preferenze fra di esse, scegliendo quelle in cui un concetto di interesse è trattato con maggiore accuratezza.
Dopo la selezione si procede alla rappresentazione dei dati secondo un unico modello per favorirne l'integrazione. Il modello concettuale preferenziale in tal caso è lo schema E-R. Nel caso di sistemi legacy la questione è più onerosa, in quanto possono essere necessarie complesse attività di reverse engineering, parzialmente semplificate dall'aiuto di potenti tools. Successivamente, la fase di analisi delle sorgenti informative, preliminare all'integrazione, procede ad identificare e rimuovere concetti e dettagli irrilevanti o ridondanti.
Il prodotto finale è uno schema concettuale di ogni sorgente utilizzata e la relativa documentazione.

Integrazione. Questa attività consiste nella fusione dei dati per avere una visione unificata del patrimonio informativo in un'unica base di dati. Lo scopo principale dell'integrazione è l'individuazione, nelle varie sorgenti, di porzioni rilevanti relative ad uno stesso aspetto della realtà di interesse all'analisi. Lo sforzo è orientato alla risoluzione di conflitti fra rappresentazioni diverse dei dati, che possono essere terminologico, strutturali o di codifica. Terminologici perché nomi diversi in schemi distinti possono rappresentare lo stesso concetto; strutturali perché un singolo concetto può essere rappresentato in forme (nomi) diversi; di codifica perché criteri diversi possono essere adoperati per codificare la stessa informazione (ad es. per il sesso, le lettere M / F oppure 0 / 1).
L'integrazione può essere utilmente guidata dai requisiti e dalle priorità individuate al passo precedente, privilegiando ad esempio le sorgenti ove la rappresentazione dati è più meticolosa.
Lo schema in uscita, quindi, è una vista complessiva dominio informativo che si vuole trattare, essa però detiene ancora caratteristiche tipiche delle basi di dati operazionali.

Progettazione. Qui si opera l'effettivo progetto e costruzione del DW. Sono possibili due approcci: *top – down* e *bottom – up*. Con il primo si parte dalla costruzione dell'EDW per procedere poi verso la suddivisione in data mart settoriali. Questo approccio suppone un notevole sforzo realizzativo, in quanto è necessaria una visione d'insieme sia

dell'applicazione che si vuole realizzare che del dominio informativo in cui essa si colloca. Se il progetto è molto ambizioso, tale compito risulta oneroso.
Con il secondo approccio si inizia dalla costruzione dei data mart e si procede poi verso l'EDW collezionandoli, senza necessaria integrazione. In scenari complessi si preferisce tale metodo per ridurre lo sforzo realizzativi con lo scotto della perdita di visione d'insieme.
Nella progettazione si parte da uno schema concettuale, senza considerare aspetti implementativi o tipologie di HW da utilizzare, e poi si passa a un livello logico dipendente dal sistema scelto. La progettazione concettuale qui, però, differisce da quella tradizionale (basi di dati), in quanto oltre ai requisiti ci si deve attenere ad un modello dei dati ottenuto dalla fase di integrazione. A tal fine lo strumento concettuale più adoperato è lo schema E – R.
La prima attività da espletare è l'individuazione dei soggetti dell'analisi multidimensionale, ovvero i fatti, le dimensioni, le misure, i membri. In un'applicazione commerciale un fatto tipico potrebbe essere la *vendita* di articoli e la dimensione il *volume* di vendita; altre volte il fatto potrebbe coincidere con un attributo, come il costo di un articolo.
Selezionato il fatto oggetto dell'analisi, si procede ad organizzarlo lungo le dimensioni, che permettono di raggruppare i fatti secondo alcuni criteri. Ad es. raggruppando gli articoli per punti vendita (dimensione luogo), per categoria (dimensione tipologia di prodotto), per data di vendita (dimensione tempo). Si vanno definendo così anche i cubi multidimensionali propri dell'analisi.
Una volta ottenuto uno schema abbastanza esaustivo si può procedere con attività di ristrutturazione navigando la struttura per aggiungere o rimuovere entità, relazioni o attributi per esplicitare dimensioni e fatti; ad esempio se un oggetto dell'analisi è un attributo o una relazione è opportuno trasformarlo in entità, che si presta meglio a rappresentare un fatto.
Avendo ora lo schema ristrutturato, si può procedere con la progettazione logica e ragionare in termini del sistema scelto, adottando ad es. uno strumento MOLAP o ROLAP.
Per gli strumenti MOLAP sono difficili considerazioni generali, in quanto essi adottano per lo più soluzioni proprietarie. Se la scelta ricade sulla tecnologia relazionale la scelta è implicata principalmente dal grado di normalizzazione che si vuole raggiungere. Se si desiderano normalizzazioni spinte, lo schema assumerà la tipica conformazione a fiocco di neve, in caso contrario si avrà il classico schema a stella.

1.4.6 I Data Mart

Un piccolo inciso a parte meritano i data mart. Essi sono una sorta di archivio semplificato di dati settoriali, di norma raccolgono dati ed informazioni relative a singoli dipartimenti di

un'organizzazione. Ad esempio il reparto amministrativo possiede il proprio data mart, così come il reparto marketing, il reparto vendite ecc. Ogni settore, quindi, ha la propria interpretazione su cosa il data mart deve contenere, sulla base degli specifici interessi di analisi.

"…un Data Warehouse non è nient'altro che l'unione di tutti i Data Marts…"

Ralph Kimball

Vi sono due tipi di data mart: *dipendenti* ed *indipendenti.*
Il data mart dipendente ha come sorgente dati il DW stesso, mentre quello indipendente trae i suoi dati dagli ambienti di applicazioni legacy. Quindi i data mart dipendenti sono legati ad una stessa fonte dati, il DW; tipicamente hanno una struttura architetturale abbastanza definita. I data mart indipendenti possono ricevere più flussi dati anche da applicazioni separate ed eterogenee; hanno quindi un modello meno definito e , nel lungo periodo, tendono ad essere instabili.

1.4.7 Interazioni fra OLAP e OLTP

Le applicazioni OLAP sono sensibilmente differenti dalle applicazioni On-Line Transactional Processing.
I sistemi OLTP effettuano un grande numero di transazioni relativamente semplici. Una tipica transazione recupera ed aggiorna pochi record contenuti in varie tabelle distinte. Le relazioni fra tali tabelle sono, in genere, molto semplici.
Una tipica transazione, relativa ad un ordine di un cliente presso un sistema transazionale, recupera tutti i dati inerenti il cliente ed inoltra il nuovo ordine. L'informazione è selezionata dai dati del cliente residenti nel sistema (se ad es. è previsto un login), i dati dell'ordine immessi dal cliente e da altre tabelle di dettaglio. Ogni riga di ogni tabella contiene un numero di identificazione del cliente che viene usato per relazionare i record in tabelle distinte. Le relazioni fra i record sono molto semplici e solo un numero limitato di essi viene modificato da una singola transazione.
Le differenze fra OLAP e OLTP possono essere così riassunte: i sistemi OLTP trattano la produzione di dati "mission-critical", acceduti tramite semplici interrogazioni; gli OLAP gestiscono dati "management-critical", acceduti tramite un'indagine iterativa a scopi di analisi.
Entrambi i sistemi hanno requisiti peculiari, di conseguenza esigono server ottimizzati per le diverse tipologie di processamento delle informazioni.

Nella seguente tabella si riassumono le differenze fondamentali:

	OLTP	OLAP
Scopo	Operazioni giornaliere	Analisi e recupero informazioni
Struttura	Relazionale	Relazionale
Modello dati	Normalizzato	Multidimensionale
Accesso	SQL	SQL + estensioni analisi dati
Tipo di dati	Dati che costituiscono il business	Dati che analizzano il business
Condizione dei dati	Mutevole, incompleta	Storica, descrittiva

Tabella 1 Analogie e differenze fra sistemi analitici e operazionali.

Per gestire efficientemente i sistemi OLTP viene incontro la tecnologia RDBMS, sviluppata con il preciso intento di supportare la memorizzazione di informazioni provenienti da tali sistemi. Il progetto dei database relazionali si concentra sull'affidabilità e sull'incremento delle prestazioni transazionali, piuttosto che sulle attività di supporto alle decisioni.

Il legame tra tali sistemi nasce in virtù del fatto che gli OLTP spesso fanno da collettore dati per i sistemi OLAP. Nel caso di una grande azienda di vendita on-line, i sistemi transazionali provvedono alle operazioni di vendita al cliente e alla fornitura di tutti i servizi in rete. Tutte le informazioni relative a queste attività, dopo l'opportuno pretrattamento, immagazzinate nel DW in strutture dati tipicamente bidimensionali (fig. 16). Tali strutture dati sono messe a disposizione del sistema OLAP che le organizzerà secondo il modello multidimensionale (cubi).

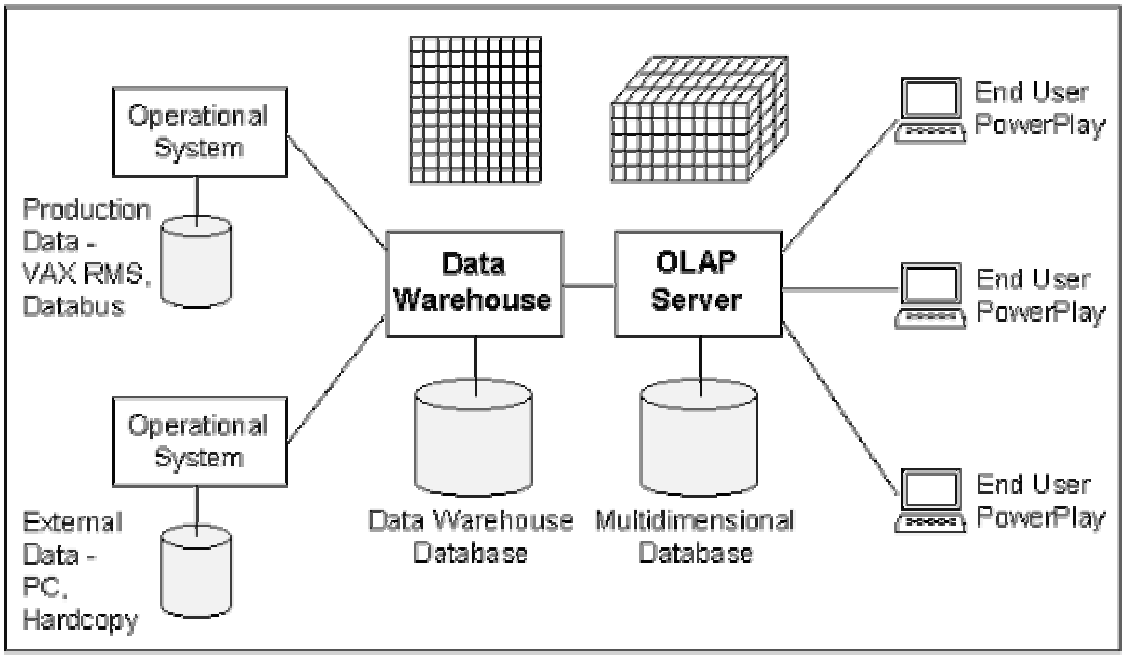


Figura 16: Ruoli e strutture dati in un possibile sistema informativo.

1.4.8 Supporto al KDD

Il campo del Data Warehousing, quindi, si riferisce all'orientamento delle attività di business di collezionare e pulire i dati provenienti da sistemi transazionali e renderli disponibili per analisi in linea e supporto alle decisioni. Esso supporta lo stadio di KDD in due modi ugualmente importanti: la pulizia dei dati e l'accesso ai dati.

Pulizia dei dati: Le organizzazioni vogliono pensare ad una vista logica unificata della grande varietà di dati e database che posseggono; perciò devono risolvere il problema del mapping dei dati in una singola convenzione dei nomi, gestire e rappresentare uniformemente i dati mancanti (missing value), e gestire rumore ed errori quando occorrono.

Accesso ai dati: Metodi ben definiti ed uniformi di accesso ai dati devono essere creati e fornire percorsi di accesso efficienti per indirizzare dati difficili da reperire, come quelli memorizzati fuori linea.

Una volta che l'organizzazione ed i propri membri hanno risolto tali questioni, nasce spontanea la domanda: "cosa farne di tutti questi dati ? ". Qui sorgono le grandi opportunità del processo di scoperta della conoscenza nelle basi di dati.

1.5 Il Processo KDD

Abbiamo visto che gli strumenti OLAP hanno come obiettivo si semplificare e supportare l'analisi interattiva dei dati, coadiuvata dall'analisi multidimensionale, per sua natura più potente dei costrutti SQL. L'obiettivo del processo KDD è più ambizioso, ovvero automatizzare il più possibile tale processo analitico. Quindi il KDD va oltre ciò che è attualmente supportato dalla maggior parte dei DBMS.

Riportiamo qui la definizione di Fayyad e Piatetsky-Shapiro :

"Il KDD è il processo non banale di identificazione di pattern nei dati che siano validi, nuovi, potenzialmente utili ed infine comprensibili".

Dunque, i *dati* sono un insieme di fatti (ad es. i casi di un database), e il *pattern* è un'espressione in qualche tipo di linguaggio che descrive un sottoinsieme dei dati o un modello applicabile ad essi. Da qui, estrarre un pattern significa anche adattare un modello ai dati; trovare strutture nei dati; o, in generale, dare una descrizione di alto livello dei dati.

Il termine *processo* implica la natura multifase del KDD, che comprende attività come la preparazione dei dati, la ricerca dei pattern, la valutazione della conoscenza, il raffinamento, tutte ripetute in iterazioni multiple. Con *"non banale"* si sottolinea il fatto che una qualche forma di inferenza è richiesta, non una semplice computazione di parametri predefiniti, come il calcolo del valor medio fra un insieme di numeri.

I pattern scoperti devono essere *validi* su dati nuovi con un certo grado di certezza. Si vuole inoltre che i pattern siano *nuovi*, se non per l'utente almeno per il sistema; ed anche *potenzialmente utili*, cioè in grado di apportare benefici agli utenti o ai compiti; infine, che siano *comprensibili*, se non immediatamente almeno dopo averli opportunamente processati.

Questa discussione implica che si possono definire misure quantitative di valutazione dei pattern estratti. In molti casi è possibile definire una misura della certezza, ad es. una stima dell'accuratezza di previsione su nuovi dati; o una misura dell'utilità, ad es. il guadagno, misurato in denaro risparmiato grazie a predizioni migliori o a diminuzioni del tempo di risposta del sistema.

Le nozioni di novità e comprensibilità sono più soggettive. In alcuni contesti, la comprensibilità può essere stimata tramite la semplicità, ad es. il numero di bit che descrivono un pattern.

Una nozione importante è, come viene definita dagli autori, la *interestingness* (Silberschatz e Tuzhilin, 1995) che potrebbe essere tradotta con attrattività o interesse. Essa è di solito vista come una misura complessiva del valore dei pattern, della validità combinata, della novità, utilità e semplicità. La funzione di interestingness può essere definita esplicitamente dall'utente o implicitamente basata su un ordinamento dei parametri stabilito dal sistema KDD sui modelli estratti.

Dati questi concetti, possiamo considerare un pattern essere conoscenza se esso supera una qualche soglia di interestingness, intendendo come conoscenza un insieme di nozioni orientate allo specifico dominio e determinate da quali funzioni l'utente sceglie e quali soglie definisce.

Il Data Mining è il passo, nel processo KDD, che consiste nell'applicare l'analisi dei dati e gli algoritmi di scoperta che producano, sotto un'accettabile efficienza computazionale, un'enumerazione dei pattern sui dati. Si noti che lo spazio dei pattern è spesso infinito, e l'enumerazione dei pattern comporta una qualche forma di ricerca in questo spazio. I vincoli computazionali reali piazzano severi limiti sui sottospazi che un algoritmo DM può esplorare.

Il processo KDD richiede l'uso dei database insieme con attività di selezione, pretrattamento, campionamento e trasformazione su di essi, applicando metodi di DM per elencare i pattern in

esso; e valutare i prodotti per identificare i sottoinsiemi dei pattern elencati giudicati conoscenza.

Il componente DM del processo KDD riguarda quindi la natura algoritmica con la quale i pattern sono estratti ed elencati dai dati.

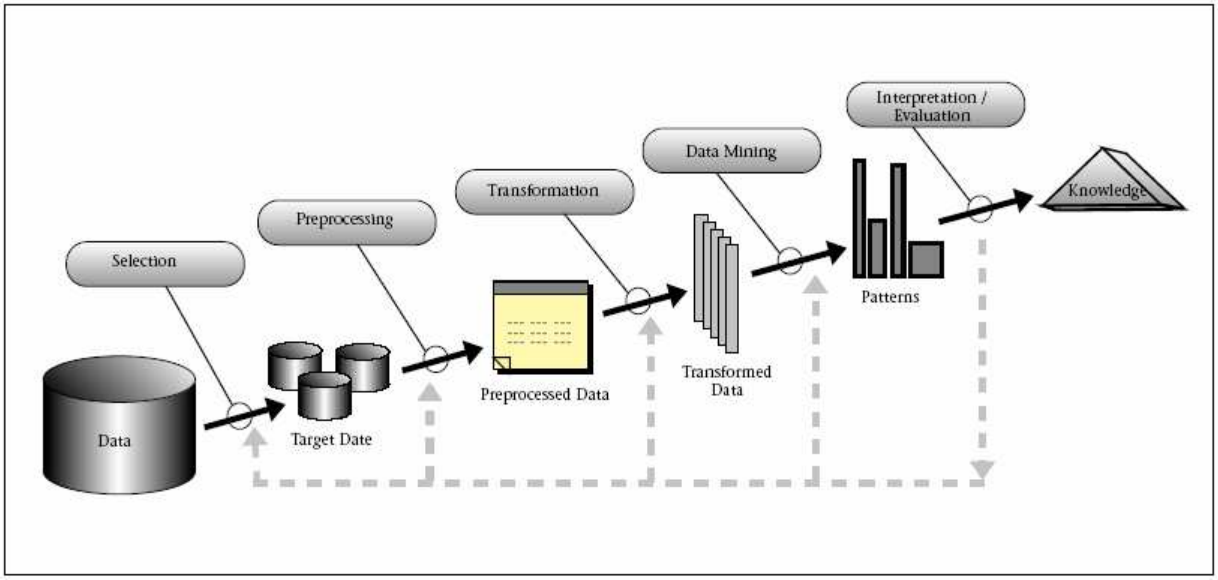


Figura 17: modello "waterfall" con retroazioni del processo KDD.

Il processo di KDD è interattivo ed iterativo, comprendente numerosi passi che necessitano molte scelte da parte dell'utente, che di solito è l'analista.

In fase preliminare è fondamentale lo sviluppo della comprensione del dominio applicativo e della conoscenza pregressa, rilevante ai fini dell'individuazione dell'obiettivo del processo dal punto di vista del cliente. In sostanza determinare gli obiettivi dell'applicazione.

Il primo passo è la creazione di un insieme di dati di riferimento (data selection), si veda fig. 17. Selezionare in un insieme di dati e focalizzare l'attenzione su un sottoinsieme di variabili o campioni, sui quali effettuare il processo di scoperta.

Il secondo passo è quello della pulizia dei dati e del pretrattamento (preprocessing). Tipiche operazioni di base sono la rimozione del "rumore" dai dati se necessario e la raccolta di informazioni per modellare o stimare il tipo di rumore; decidere strategie e politiche per la gestione dei campi dei dati mancanti; ed in generale appuntare informazioni su sequenze temporali, cambiamenti strutturali ecc.

Terzo passo è la trasformazione dei dati, anche nota come riduzione e proiezione, in quanto si effettua una riduzione dimensionale dei dati con vari metodi di trasformazione. In questo modo il numero effettivo di variabili in esame viene ridotto e vengono trovate rappresentazioni comuni nei dati. Le due fasi precedenti sono spesso molto onerose, in special modo se le fonti dei dati sono eterogenee.

La quarta fase è quella in cui si effettua il DM vero e proprio, cercando di inquadrare gli obiettivi dell'intero processo. Si inizia con una indagine esplorativa al fine di formulare le ipotesi analitiche; poi si procede alla selezione dei modelli e dei metodi tipici del DM, come la classificazione, regressione, clusterizzazione e cosi via, e degli algoritmi con i quali realizzarli.

Questa attività include la selezione di modelli e paramentri appropriati (ad es. un modello su dati categorici o su vettori di numeri reali) o di metodi di DM con particolari caratteristiche desiderabili per l'utente (ad es. l'utente potrebbe preferire un modello facilmente comprensibile ad uno con spiccate capacità predittive). Si entra di conseguenza nel cuore dell'attività di mining: la ricerca dei pattern di interesse sulle varie forme di rappresentazione dei dati o su un loro sottoinsieme; tali forme incudono regole o alberi di classificazione, regressioni e raggruppamenti.

Il quinto stadio del processo è l'interpretazione e la valutazione (interpretation / evaluation) dei pattern estratti eventualmente ripercorrendo i passi precedenti attraverso opportune retroazioni, come descritto in figura. Tipicamente questo passo include la visualizzazione e la presentazione dei modelli portati alla luce e l'eliminazione di pattern ridondanti.

Il passo ulteriore e finale del processo, che ne rappresenta anche lo scopo ultimo, è quello di agire secondo la conoscenza acquisita, integrandola magari con altri sistemi per operare ulteriori decisioni; o semplicemente documentarla e riportarla alle parti interessate. Tale fase può inoltre far sorgere conflitti con conoscenze precedentemente acquisite ed aiutare a risolverli.

Come prima accennato il processo di KDD può comportare un significativo numero di iterazioni contenenti loop verso fasi precedenti del flusso di base; quasi mai quindi esso viene effettivamente implementato come un modello a cascata.

1.6 Architettura di un sistema Data Mining

Basandosi su queste nozioni, si può comprendere l'architettura di un tipico sistema di data mining ed il ruolo dei suoi componenti principali:

1. **DBMS, Data Warehouse ed altri magazzini di informazioni.**
   Un singolo o un insieme di database, data warehouse, fogli elettronici, file, archivi o qualsiasi altro contenitore di informazioni da trattare.
2. **Data Base o Data Warehouse Server.**
   Tale unità e responsabile del mantenimento e recupero dei dati rilevanti alla luce delle richieste dell'utente.

3. **Conoscenza di base.**

   È la conoscenza di dominio che guida la ricerca ed utile a valutare i pattern risultanti. Questo tipo di conoscenza include gerarchie concettuali, usate per organizzare gli attributi od i valori degli attributi a diversi livelli di astrazione. Anche le opinioni dei clienti costituiscono conoscenza, che può essere adoperata per la validazione di pattern. Altri esempi di conoscenza di dominio possono essere vincoli ulteriori, soglie e metadati descriventi le sorgenti informative.

4. **Motore di Data Mining.**

   Componente essenziale del sistema DM, consiste idealmente in un insieme di moduli funzionali che insieme realizzano compiti come la caratterizzazione, l'analisi associativa, la classificazione e così via.

5. **Modulo di valutazione pattern.**

   Questo componente fondamentalmente impiega misure sull'interesse dei pattern ed interagisce con i moduli del motore per focalizzare la ricerca verso i pattern interessanti. Talvolta il modulo può essere integrato nel motore DM

6. **GUI (Graphical User Interface)**

   Questo modulo si interpone fra il sistema e l'utente, permettendo a quest'ultimo di interagire specificando i compiti e le interrogazioni, fornendo informazioni per aiutare la ricerca ed esplorando i risultati intermedi del processo. Inoltre il modulo permette di navigare la struttura del DBMS o del DW o anche dei modelli ottenuti, oltre che organizzare il reporting dei risultati in forme differenti.

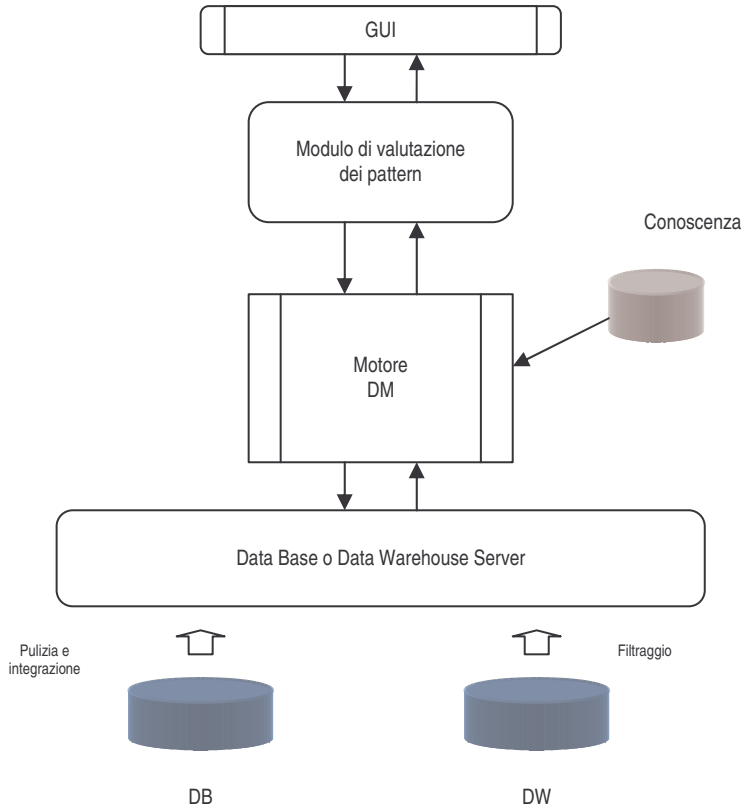


Figura 18: Architettura e principali componenti di un sistema Data Mining.

Cap. II

IL MODELLO DI PROCESSO

CRISP - DM

2.1 Panoramica del progetto

CRISP-DM (CRoss-Industry Standard Process – Data Mining) è un progetto finanziato dalla Commissione Europea, volto a definire un approccio standard ai progetti di data mining. Il CRISP-DM affronta le necessità di tutti gli utenti coinvolti nella diffusione di tecnologie di data mining per la soluzione di problemi aziendali. Scopo del progetto è definire e convalidare uno schema di approccio indipendente dalla tipologia di business. Questo tipo di approccio renderà l'implementazione di grandi progetti di data mining più veloce, più efficiente, più sicura e meno costosa. I problemi affrontati includono:

- passaggio da problemi di business a problemi di data mining
- acquisizione e comprensione dei dati
- identificazione e soluzione dei problemi inerenti i dati
- applicazione di tecniche di data mining
- interpretazione dei risultati del data mining nel contesto aziendale
- diffusione e manutenzione dei risultati raggiunti
- acquisizione e trasferimento del know-how acquisito per estendere il beneficio dell'esperienza ai progetti futuri

Oltre a fornire un modello di processo per affrontare i problemi di data mining, il progetto fornirà anche linee guida per la soluzione dei potenziali problemi che possono insorgere nella definizione di un progetto di data mining.

2.2 Gli attori

I partner del progetto sono:

- NCR, leader mondiale di soluzioni di data warehouse.
- Daimler-Benz, uno dei maggiori gruppi industriali europei, attivo nei settori automobilistico, aerospaziale, delle telecomunicazioni e dei servizi.

- Integral Solutions Limited (ISL), la società inglese che ha sviluppato il software di data mining Clementine. ISL è stata acquisita da SPSS nel dicembre 1998.
- OHRA, una delle maggiori compagnie indipendenti di assicurazioni, con sede in Olanda.

I membri del progetto forniscono la loro esperienza, maturata nella gestione di complessi progetti di data mining. Daimler Benz e OHRA forniscono anche le risorse per le fasi di test del modello di approccio CRISP, applicato a progetti di data mining di medie e grandi dimensioni.

I membri del progetto mirano a promuovere e a pubblicare il modello CRISP-DM con l'intento di stabilire uno standard. Un ruolo chiave è giocato dal CRISP-DM SIG (Special Interest Group). Il gruppo coinvolge numerosi utenti di data mining provenienti da diverse tipologie aziendali, assieme a fornitori di soluzioni di data mining. I membri del SIG condividono le loro esperienze, contribuiscono alla progettazione del modello e hanno accesso tempestivo ai risultati del progetto.

Il primo meeting CRISP-DM si è svolto ad Amsterdam alla fine di novembre del 1998, con oltre venti partecipanti fra fornitori e utenti di soluzioni di data mining. Scopo del convegno era discutere i requisiti e le aree di interesse per un modello di approccio al data mining. I risultati delle discussioni sono stati utilizzati per aggiornare il modello.

Oltre 100 fra fornitori di soluzioni di data mining, consulenti e utenti di grandi aziende hanno sottoscritto la loro adesione al SIG.

2.3 La metodologia

La metodologia CRISP per il data mining è descritta in termini di un processo gerarchico, consistente in un insieme di compiti (task) affrontati a 4 livelli di astrazione: **fase** (phase), **compito generico** (generic task), **compito specializzato** (specialized task) ed **istanza di processo** (process instances). Si veda la fig. 19

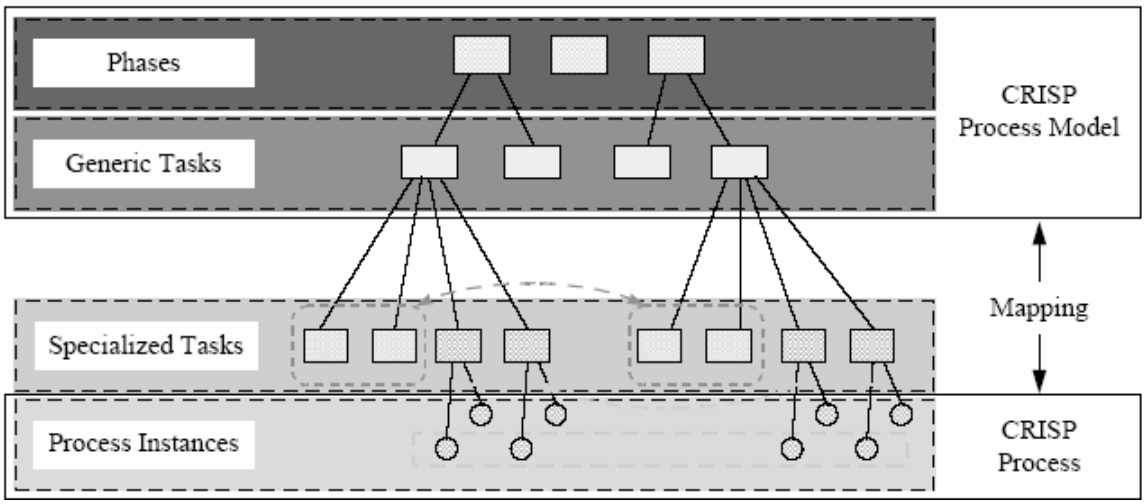


Figura 19: I livelli di astrazione fra il modello e l'istanza del processo.

Al livello più alto, il processo di data mining è organizzato in fasi; ogni fase consiste di diversi compiti generici di secondo livello. Questo secondo livello è chiamato generico in quanto intende essere sufficientemente generale da contemplare tutti i possibili casi di DM. Il compito generico vuole essere il più completo e stabile possibile. Completo perché mira a coprire l'intero processo di DM e tutte le sue eventuali applicazioni. Stabile perché il modello dovrebbe rimanere valido per sviluppi non previsti, come nuove tecniche di modellizzazione.
Il terzo livello, detto compito specializzato, descrive come le azioni del compito generico devono essere attuate in situazioni specifiche. Ad esempio, al secondo livello vi può essere un compito chiamato "pulisci dati"; al terzo livello si descrive come tale compito si diversifica nelle differenti situazioni, come pulire valori numerici piuttosto che attributi categorici, o se il tipo di problema è di clusterizzazione o modellazione predittiva.
La descrizione delle fasi e dei compiti come passi discreti e distinti da eseguire in un ordine predeterminato rappresenta una idealizzazione della sequenza degli eventi. Nella pratica i compiti sono sovente eseguiti in ordini diversi e spesso si rendono necessari ritorni a fasi precedenti per ripetere alcune azioni.
Il livello istanza di processo è la registrazione delle azioni, delle decisioni e dei risultati dell'attuale implementazione del processo di estrazione.

2.3.1 Il mapping fra modello generico e modello specifico

Come si evince dalla Fig.19 è necessario mappare i compiti generici del modello con quelli che ne rappresentano l'effettiva esecuzione. Questa operazione è guidata dal *contesto* del DM, di cui possiamo individuare 4 tipologie:

1. Il *dominio applicativo* è l'area specifica in cui il progetto DM si colloca.

2. La *tipologia di problema* descrive le classi e gli obiettivi che il progetto deve riguardare e raggiungere.
3. L'*aspetto tecnico* riguarda le questioni specifiche del processo di estrazione, come le sfide tecniche che si presentano durante la sua esecuzione.
4. Gli *strumenti* e le *tecniche* segnalano quali strumenti sono stati utilizzati durante lo sviluppo.

Uno specifico contesto di DM è costituito da una quadrupla di valori effettivi lungo queste dimensioni. Ad esempio, un progetto DM riguardante un problema di classificazione per la individuazione di frodi fiscali, con una pesante gestione degli outliers, per cui è stato utilizzato Clementine SPSS.

La strategia di base per mappare i modelli generici con quelli peculiari del problema può avvalersi dei seguenti suggerimenti:

- Analizzare il contesto specifico.
- Rimuovere i dettagli non applicabili al contesto.
- Aggiungere i dettagli applicabili al contesto.
- Specializzare ed istanziare i compiti generici secondo le caratteristiche concrete del contesto.
- Eventualmente rinominare compiti generici per fornire significati più espliciti inerenti al contesto.

2.4 Il modello di riferimento

Il ciclo di vita di un progetto di data mining consiste in 6 fasi, come descritto in fig. 20. La sequenza delle fasi non è stretta, muoversi avanti e dietro fra di esse è sempre richiesto; ciò dipende dall'output di ogni fase e/o da quale compito particolare deve essere eseguito nella fase successiva. Le frecce indicano le corrispondenze più frequenti fra le fasi.

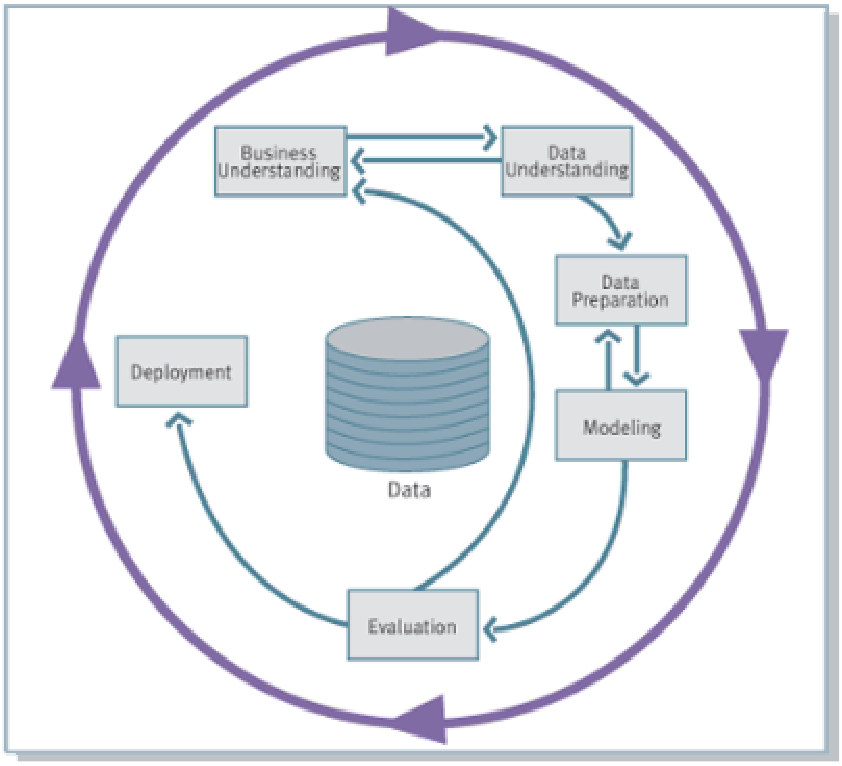


Figura 20: fasi del ciclo di vita di un progetto DM secondo il modello CRISP-DM.

Il cerchio esterno in figura sta a simboleggiare la natura ciclica del DM stesso; il processo non termina una volta che una soluzione è stata dispiegata, ma l'esperienza acquisita durante il processo e dalla soluzione può porre nuove questioni di business e problematiche più specifiche. I susseguenti processi DM beneficeranno dell'esperienza accumulata.

Diamo qui una breve descrizione delle fasi in cui il processo si snoda:

- **FASE 1: Comprensione del Business (Business understanding)**

  La fase iniziale si concentra sulla comprensione degli obiettivi del progetto e dei requisiti dal punto di vista dell'attività di Business da supportare, al fine di convertire questa conoscenza in una definizione del problema da affrontare e concepire una pianificazione preliminare per raggiungere gli obiettivi.

- **FASE 2: Comprensione dei dati (Data understanding)**

  Questa fase parte con un collezione iniziale dei dati per procedere poi con una serie di attività mirate a "familiarizzare" con essi: identificare la qualità dei dati, intuire prime conformazioni sulla loro natura, individuare sottoinsiemi interessanti per formulare ipotesi su informazioni nascoste in essi.

- **FASE 3: Preparazione dei dati (Data Preparation)**

  La fase di preparazione ricopre tutte quelle attività atte a costruire il set di dati finale, che alimenterà gli strumenti utilizzati, dai dati grezzi iniziali. Questa fase risulta molto

pesante e viene espletata in diverse occasioni e non in un ordine prestabilito. I compiti inerenti possono includere selezioni di tabelle, attributi, record e operazioni di pulizia.

- **FASE 4: Modellizzazione (Modeling)**

  In questa fase le varie tecniche di modellizzazione sono scelte ed applicate e vengono calibrati i relativi parametri a seconda del contesto. Tipicamente esistono svariate tecniche per lo stessa tipologia di problema, esse differiscono nelle esigenze sul formato dei dati, quindi sono spesso richiesti ritorni alla fase di preparazione.

- **FASE 5: Valutazione (Evaluation)**

  A questo stadio un modello (o più modelli) è stato costruito, e può sembrar avere una qualità sufficiente dal punto di vista dell'analisi. E' buona norma, però, procedere ad una attenta revisione del modello prima di procedere ed accertarsi della sua efficacia, cioè verificare se effettivamente realizza gli obiettivi di partenza. Alla fine di questa fase devono essere stilate le decisioni sull'uso dei risultati del DM.

- **FASE 6: Deployment**

  Se il fine del modello è incrementare la conoscenza sui dati, essa deve essere organizzata e presentata in modo che l'utente possa utilizzarla comodamente. A seconda dei requisiti, la fase di deployment può consistere nella semplice generazione di report o più complesso, come l'implementazione di un processo DM ripetibile. Tipicamente l'attore di questo passo non è l'analista ma il cliente, è fondamentale quindi la comprensione da parte sua delle azioni da compiere per fare un uso proficuo del modello creato.

Di seguito è riportata una vista complessiva dell'intero processo, dove sono riportati i compiti generici (in grassetto) ed i relativi output (in corsivo).

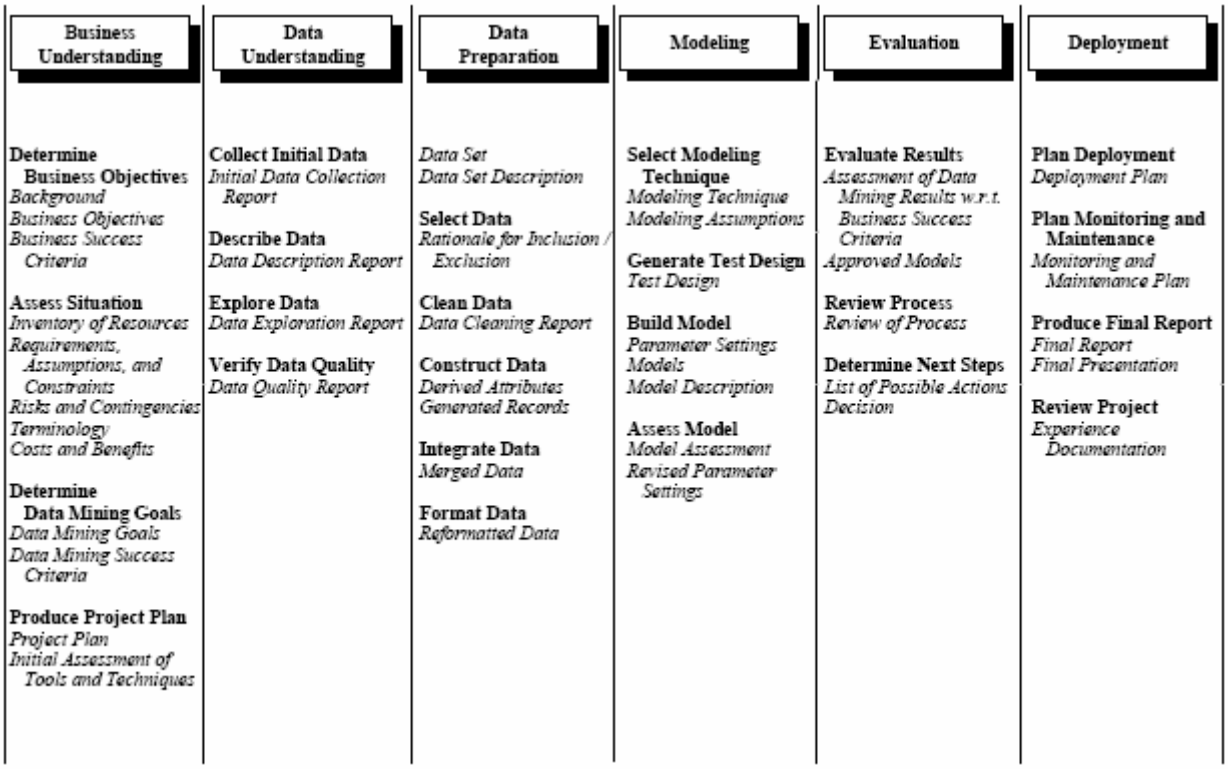


Figura 21: Fasi e compiti generici del modello riportati nel documento ufficiale [7].

Procediamo ad una analisi più approfondita dei task e dei relativi prodotti di uscita.

2.4.1 Comprensione del Business

Determinazione degli obiettivi del Business

Compito: **Determinare gli obiettivi del Business**

Il primo obiettivo dell'analista è quello di capire, dal punto di vista del business, cosa il cliente vuole portare a termine. Spesso il cliente può fornire obiettivi contrastanti e vincoli che devono essere opportunamente bilanciati. L'obiettivo principale dell'analista e di scoprire all'inizio i fattori più importanti che possono influenzare radicalmente lo sviluppo. Una possibile conseguenza della negligenza di questo passo può essere quella di produrre le risposte giuste alle domande sbagliate.

Output: **Background**

Dettagli sulle informazioni note sul business dell'azienda all'atto di intraprendere il progetto.

Attività: ORGANIZZAZIONE

- Sviluppare l'organigramma delle divisioni, dipartimenti e gruppi di progetto. Il quadro informativo deve identificare i nomi dei manager e relative responsabilità.

- Identificare le figure chiave del business ed il loro ruolo.
- Reperire uno sponsor interno (sponsor finanziario, esperti di dominio).
- Segnalare il comitato direttivo ed i suoi membri.
- Identificare i settori del business che verranno influenzati dal progetto DM (Marketing, Vendite, Finanza, Assicurazioni).

AREA DEL PROBLEMA

- Delineare l'area del problema (Marketing, Servizio Clienti, Sviluppo di Business).
- Descrivere il problema in termini generali.
- Verificare lo stato corrente del progetto (se i settori di business sono a conoscenza del progetto che si sta iniziando o è necessario annunciarlo).
- Chiarire i prerequisiti del progetto (es. qual è la motivazione del progetto? Si sta già usando il DM?)
- Se necessario, preparare una presentazione del DM ai settori di business coinvolti.
- Identificare i gruppi fruitori del progetto (se sono necessari rapporti per i vertici amministartivi o il sistema sarà usato da semplici utenti).
- Identificare le esigenze e le aspettative dell'utente.

SOLUZIONE CORRENTE

- Descrivere la soluzione attualmente utilizzata per il problema.
- Illustrare i vantaggi e gli svantaggi della soluzione ed il grado con cui è accettata dagli utenti.

Output: **Obiettivi del Business**

Lista di tutte le domande poste dal cliente.

A fianco dell'obiettivo principale del progetto il cliente pone domande collaterali alle quali desidera una risposta. Ad esempio, per una banca l'obiettivo primario può essere la previsione su quali clienti sono inclini a passare ad un concorrente sul mercato, ma una domanda specifica può essere la seguente: "Quali sono i canali (reti ATM, internet, sportelli) che influenzano maggiormente la scelta fra rimanere o cambiare banca?".

Attività:

- Descrizione informale dei problemi che si intendono risolvere con il DM
- Specificare tutti gli interrogativi del business precisamente.

- Specificare tutti i requisiti del business (ad es. non si vuole perdere nessun cliente).
- Esprimere i benefici attesi in termini di business.

Output: **Criteri di successo del Business**

Descrizione di quali sono i criteri o motivi di successo dell'applicazione dal punto di vista del business. Criteri quantitativi come la riduzione della fuga dei clienti ad un certo livello, o più soggettivi come dare una interpretazione utile alle relazioni trovate. Nell'ultimo caso deve essere indicato anche chi dà l'interpretazione.

Ogni criterio di successo deve essere relazionato ad almeno uno degli obiettivi di business specificati.

Attività:

- Esplicitare i criteri di successo del business (es. incremento del tasso di risposta ad una campagna pubblicitaria a mezzo posta del 10%, percentuale di iscrizioni aumentata del 15% ecc.).
- Identificare chi accerterà i criteri di successo.

Accertamento della situazione

Compito: **Accertare la situazione**

Questo compito riguarda l'indagine sui fatti circa tutte le risorse, i vincoli, le assunzioni e gli altri fattori che devono essere considerati nel determinare gli obiettivi dell'analisi dei dati e del piano di progetto. Nel compito precedente il fine è di andare rapidamente al punto cruciale del problema, qui è di articolare tutti i dettagli.

Output: **Inventario delle risorse**

Lista di tutte le risorse disponibili per il progetto e di qualunque natura: personale (esperti di business, esperti di dati, supporto tecnico, personale per il data mining), dati (estratti stabili, accesso ai warehouse attivi o a dati operazionali), risorse computazionali (piattaforme HW), software (tool di data mining, altri software coinvolti).

Attività: RISORSE HARDWARE

- Identificare l'hardware di base.
- Stabilire la disponibilità dell'HW di base per il progetto DM.
- Verificare se il piano di manutenzione intralcia la disponibilità dell'HW per il progetto.

SORGENTI DI DATI E CONOSCENZA

- Identificare le sorgenti dati.
- Identificare il tipo di sorgenti dati (sorgenti on-line, esperti, documentazione scritta ecc.).
- Identificare le sorgenti di conoscenza.
- Identificare il tipo di sorgenti di conoscenza (sorgenti on-line , esperti, documentazione scritta, ecc.).
- Verificare le tecniche e gli strumenti disponibili.
- Descrivere il know-how di background.

RISORSE UMANE

- Identificare gli sponsor di progetto (se diverso dallo sponsor interno).
- Identificare l'amministratore di sistema, l'amministratore di database, la squadra di supporto tecnico, ecc.
- Identificare gli analisti di mercato, esperti data mining, statistici e verificarne la disponibilità.
- Verificare la disponibilità degli esperti di dominio per le fasi a seguire.

Output: **Requisiti, Assunzioni e Vincoli**

Lista di tutti i requisiti di progetto inclusi prospetti di completamento, comprensibilità e qualità dei risultati e sicurezza, emendamenti e note legali.

Accertamento della possibilità di accesso a tutti i dati.

Lista delle assunzioni fatte sul progetto. Queste possono essere assunzioni sui dati verificabili durante il processo di DM, ma possono anche essere assunzioni non verificabili caratteristiche del business aziendale sul quale il progetto poggia. Nel secondo caso è importante una lista dettagliata se esse dettano condizioni sulla validità dei risultati.

Lista dei vincoli di progetto. Possono essere vincoli sulla disponibilità delle risorse, ma anche vincoli tecnologici come sulla dimensione fisica dei dati da sottoporre a modellizzazione.

Attività: REQUISITI

- Specificare il profilo del gruppo obiettivo.
- Catturare i requisiti sulla programmazione del progetto.
- Catturare i requisiti sulla comprensibilità, accuratezza, manutenibilità e ripetibilità del progetto.
- Catturare i requisiti sulla sicurezza, privacy e restrizioni legali.

ASSUNZIONI

- Chiarificare le assunzioni implicite e renderle esplicite (es. per mirare alla soluzione di una domanda di business è necessario avere un numero minimo di clienti con età media di 50).
- Elencare le assunzioni sulla qualità dei dati (accuratezza, disponibilità,…).
- Elencare le assunzioni sui fattori esterni (questioni economiche, prodotti competitivi sul mercato, vantaggi tecnici).
- Chiarificare le assunzioni che portano ad una stima quantitativa (es. il prezzo di uno strumento è presumibilmente minore di € 1000).
- Elencare le assunzioni su come deve essere descritto ed esposto il modello, a scopo cognitivo, al management o agli sponsor.

VINCOLI

- Verificare i vincoli generici (economici, legali, temporali, di risorse).
- Verificare i diritti d'accesso alle sorgenti dati (restrizioni di accesso, password,…).
- Verificare l'accesso tecnico ai dati (sistema operativo, sistema di gestione, formato dei database o dei file).
- Verificare l'accessibilità della Conoscenza.
- Verificare i vincoli di budget (costi fissi, costi variabili, costi di implementazione,…).

Output: **Rischi e Contingenze**

Lista dei rischi. Ovvero gli eventi che possono occorrere e ritardare il progetto o causarne il fallimento. Lista dei piani di contingenza ovvero di quali azioni intraprendere se i rischi si presentano.

Attività: RISCHI

- Identificare i rischi di business (es. se i concorrenti escono prima con risultati migliori).
- Identificare i rischi organizzativi (es. se il dipartimento che commissiona il progetto non ha fondi sufficienti).
- Identificare i rischi finanziari (es. se i fondi supplementari dipendono dai primi risultati di mining).
- Identificare i rischi tecnici (es. l'indisponibilità di alcune piattaforme può causare l'arenaggio del progetto).
- Identificare i rischi dipendenti dai dati (es. scarsa qualità).

CONTINGENZE

- Determinare le condizioni secondo cui i rischi possono presentarsi.
- Sviluppare un piano di contingenza.

Output: **Terminologia**

Glossario dei termini rilevanti per il progetto, esso si compone di due parti: il glossario dei termini di business, che forma una parte della comprensione del business; e il glossario dei termini di DM.

Attività:

- Verificare la presenza di glossari precedenti, altrimenti iniziare ad abbozzarne uno.
- Comunicare con gli esperti di dominio per comprendere la terminologia.
- Acquisire dimestichezza con la terminologia di business.

Output: **Costi e Benefici**

Analisi costi-benefici del progetto. Comparazione dei costi del progetto con i benefici derivanti al business in caso di successo. Il confronto deve essere il più preciso possibile, anche usando misure monetarie in casi commerciali.

Attività:

- Stimare i costi per la raccolta dati.
- Stimare i costi per lo sviluppo e l'implementazione della soluzione.
- Identificare i benefici una volta attuata la soluzione (es. aumento della soddisfazione dei clienti, ritorno sugli investimenti, incremento delle entrate,…).
- Stimare i costi operativi
- Individuare e stimare costi nascosti come quelli relativi a ripetizioni di fasi di estrazione e trasformazione dati, cambiamenti nel flusso di lavoro, periodi di formazione degli utenti.

Determinare gli obiettivi del Data Mining

Compito: **determinare gli obiettivi del DM**

Un obiettivo di business esprime i suoi scopi nella propria terminologia. Un obiettivo di DM esprime i suoi scopi in termini tecnici. Per esempio, un obiettivo di business può essere "incrementare la vendita dei cataloghi ai clienti"; mentre un obiettivo di DM può essere "stimare quanti articoli un cliente comprerà, dati i suoi acquisti negli ultimi 3 anni, informazioni demografiche e prezzo degli articoli".

Output: **Obiettivi del DM**

Descrizione degli output desiderati dal progetto che permetteranno il raggiungimento degli obiettivi di business.

Attività:

- Tradurre le questioni di business in obiettivi di data mining (es. una campagna di marketing richiede la segmentazione dei clienti per decidere a chi rivolgere la campagna; può includere la specifica del livello e della dimensione di ciascun segmento).
- Specificare la natura del problema di DM (es. classificazone, segmentazione, descrizione, clustering, previsione, …) .

Output: **Criteri di successo del DM**

Definizione dei criteri per la riuscita del progetto in termini tecnici, come per esempio il grado di accuratezza di previsione. Può essere necessario descrivere tali criteri in termini soggettivi, in tal caso il soggetto deve essere identificato.

Attività:

- Specificare i criteri per l'accertamento del modello (accuratezza del modello, prestazioni, complessità,…).
- Definire dei casi di prova per i criteri di successo.
- Esplicitare i criteri che sono riconducibili a criteri di successo del business. Si ricordi che i criteri di successo del DM sono differenti dai criteri di successo del business definiti in precedenza.

Produzione di un piano di progetto

Compito: **Produrre un piano di progetto**

Descrivere il piano voluto per il raggiungimento degli obiettivi di DM e , dai qui, gli obiettivi di business. Il piano deve specificare in anticipo l'insiene di passi da effettuare nel resto del progetto, inclusa la selezione iniziale di strumenti e tecniche.

Output: **Piano di progetto**

Lista delle tappe da eseguire nel processo, inclusa la durata, le risorse richieste, input, output e dipendenze. Dove possibile fare esplicito riferimento ad eventuali iterazioni delle fasi di data mining, come le fasi di modellizzazione e valutazione.

Come parte del piano di progetto, analizzare le dipendenze fra il prospetto temporale e i rischi. Rimarcare i risultati di questa analisi associando ad essa raccomandazioni e azioni da intraprendere al palesarsi dei rischi.

Il piano di progetto contiene piani dettagliati per ogni fase, ad esempio a che punto della fase di valutazione la strategia scelta viene messa in pratica.

Il piano di progetto è quindi un documento dinamico nel senso che alla fine di ogni fase è prevista una revisione dei progressi e di conseguenza un aggiornamento del documento. Tali punti di revisione sono parte integrante del piano di progetto.

Attività:

- Definire il piano di processo iniziale e discuterne la fattibilità con il personale coinvolto.
- Organizzare gli obiettivi identificati e le tecniche selezionate all'interno di una procedura coerente per risolvere le questioni di business e soddisfare i relativi criteri di successo.
- Stimare lo sforzo e le risorse necessarie alla realizzazione di una soluzione (è utile qui sfruttare l'esperienza di altre persone). Ad es. è stato postulato che, in un progetto DM, il 50% - 70% del tempo e dello sforzo è impiegato nella fase di *Preparazione dei Dati*, il 20% - 30% nella fase di *Comprensione dei Dati*, mentre solo il 10% - 20% nelle fasi di *Comprensione del Business*, Modellizzazione, *Valutazione* e il 5% - 10% nella fase di *Deployment.*
- Identificare i passi critici.
- Rimarcare i punti decisionali.
- Rimarcare i punti di revisione.
- Identificare le iterazioni più importanti.

Output: **Valutazione iniziale di strumenti e tecniche**

Selezione degli strumenti di data mining, che supportano vari metodi a seconda della fase in cui si ci trova. E' importante valutare tecniche e strumenti all'inizio del processo in quanto la loro selezione tende ad influenzare fortemente il prosieguo delle attività.

Attività:

- Creare una lista di selezione di strumenti e tecniche (o usarne una se già disponibile).
- Scegliere i potenziali strumenti e tecniche.
- Valutare l'appropriatezza delle tecniche.
- Rivedere e dare una priorità alle tecniche alla luce della valutazione di soluzioni alternative.

2.4.2 Comprensione dei dati

Collezione dei dati

Compito: Collezionare i dati

Acquisire all'interno del progetto i dati elencati nelle risorse di progetto. Questa collezione iniziale include il caricamento dei dati se necessario alla loro comprensione. Se ad es. si utilizza uno strumento per la comprensione dei dati, ha perfettamente senso caricare i dati per renderli disponibili a tale strumento. Questo sforzo è un primo passo che inizia la fase di preparazione dei dati. Se si acquisiscono dati da sorgenti multiple, l'integrazione è un compito ulteriore da effettuare sia in questa sede o successivamente in fase di preparazione.

Output: Report sulla collezione iniziale dei dati

Lista dei set di dati acquisiti, insieme alla loro locazione nel progetto, i metodi adottati per acquisirli e i problemi incontrati durante lo svolgimento. I problemi sorti e le soluzioni trovate vengono elencate al fine di aiutare replicazioni future di questo passaggio del processo su progetti analoghi.

Attività: PIANIFICAZIONE DEI REQUISITI DEI DATI

- Individuare quali informazioni sono necessarie (es. se solo gli attributi dati o altre informazioni addizionali).
- Accertarsi che tutte le informazioni necessarie a raggiungere gli scopi del DM siano disponibili.

CRITERI DI SELEZIONE

- Specificare i criteri di selezione (es. quali attributi sono necessari per gli specifici obiettivi, quali possono essere considerati irrilevanti, quanti attributi possiamo gestire con le tecniche selezionate).
- Selezionare tabelle e/o file di interesse,
- Selezionare dati all'interno di file o tabelle.
- Decidere quanti dati storici usare, se disponibili (es. se sono disponibili dati relativi agli ultimi 18 mesi, possono essere sufficienti quelli relativi agli ultimi 12).
- Prestare attenzione ai dati acquisiti da sorgenti diverse. Ad es. quando sono integrati nella base principale, essi possono far sorgere problemi di inconsistenza di formato, invalidità dei dati, ecc.

INSERIMENTO DATI

- Codificare eventuali entry di testo libero nei dati o raggrupparle in specifiche registrazioni.
- Tentare di acquisire gli attributi mancanti.
- Descrivere come estrarre i dati.

Descrizione dei dati

Compito: **Descrivere i dati**

Esaminare sommariamente le proprietà "di superficie" dei dati acquisiti e riportarne le osservazioni fatte.

Output: **Report sulla descrizione dei dati**

Lista descrittiva dei dati acquisiti, inclusi: il formato dei dati, la quantità, il numero di record e di campi in ogni tabella, l'identità dei campi ed ogni altra caratteristica individuata.

Attività: ANALISI VOLUMETRICA DEI DATI

- Identificare i dati e il metodo di acquisizione.
- Accedere alle sorgenti dati.
- Utilizzare analisi statistiche se appropriate.
- Rapportare le tabelle e le loro relazioni.
- Verificare il volume dei dati, il numero di tuple (nota sul termine), la loro complessità.
- Esaminare le entry di testo libero.

TIPI DI ATTRIBUTI E VALORI

- Verificare l'accessibilità e la disponibilità degli attributi.
- Verificare il tipo degli attributi (numerico, simbolico, categorico,…).
- Verificare il campo dei valori degli attributi.
- Analizzare le correlazioni fra attributi.
- Comprendere il significato di ogni attributo e dei relativi valori in termini di business.
- Calcolare statistiche di base per ogni attributo (es. distribuzioni, medie, massimi, minimi, deviazioni standard, varianze).
- Analizzare le statistiche di base e ricondurle al loro significato in termini di business.
- Verificare l'effettiva rilevanza degli attributi rispetto agli obiettivi.
- Verificare la coerenza d'uso del significato dell'attributo.
- Intervistare esperti di dominio sulla rilevanza degli attributi.

- Effettuare un bilanciamento dei dati se necessario.

CHIAVI

- Analizzare le relazioni fra le chiavi.
- Verificare le sovrapposizioni di valori di attributi chiave fra le relazioni.

REVISIONE DELLE ASSUNZIONI E DEGLI OBIETTIVI

- Aggiornare la lista delle assunzioni se necessario.

Esplorazione dei dati

Compito: **Esplorare i dati**

Questo compito affronta direttamente le domande di DM che possono trovare risposta con interrogazioni, visualizzazioni e reporting sui dati. Queste possono riguardare: la distribuzione degli attributi chiavi nelle tabelle; l'individuazione di un attributo oggetto di una previsione; relazioni tra coppie o gruppi di attributi similari in tabelle diverse; risultati di semplici aggregazioni effettuate in precedenza da terzi; proprietà di sottopopolazioni significative; semplici analisi statistiche. Queste analisi possono rivolgersi direttamente agli obiettivi di DM; ed anche contribuire al raffinamento della descrizione dei dati e dei report sulla qualità.

Output: **Report sull'esplorazione dei dati**

Descrizione dei risultati di questo task, incluse le prime considerazioni ed ipotesi e il loro impatto sul resto del progetto. Il report deve contenere grafici e diagrammi che descrivono le caratteristiche dei dati e suggeriscono considerazioni significative per ulteriori ispezioni.

Attività: ESPLORAZIONE DEI DATI

- Analizzare dettagliatamente le proprietà di attributi di interesse (statistiche di sottopopolazioni interessanti).
- Identificare le caratteristiche delle sottopopolazioni.

FORMULAZIONE IPOTESI PER ANALISI FUTURE

- Considerare e valutare le informazioni nei rapporti descrittivi dei dati.
- Formulare ipotesi ed identificare azioni.
- Trasformare le ipotesi in obiettivi di DM, se possibile.
- Chiarificare gli obiettivi di DM e renderli più precisi. La ricerca alla cieca non è necessariamente inutile, ma una ricerca mirata agli obiettivi è ovviamente preferibile.
- Effettuare analisi di base per verificare le ipotesi.

Verifica della qualità dei dati

Compito: **Verificare la qualità dei dati**

Esaminare la qualità dei dati significa rispondere a domande del tipo: "i dati sono completi ? sono corretti o contengono errori ? se si, quanto sono frequenti tali errori ? ci sono valori omessi nei dati ? in tal caso quanto sono comuni? quali attributi riguardano ?".

Output: **Report sulla qualità dei dati**

Lista dei risultati della verifica di qualità. Se esistono problemi di qualità, elenco delle possibili soluzioni. Le soluzioni alle questioni di qualità dei dati generalmente dipendono pesantemente sia dai dati che dalla conoscenza del business.

Attività:

- Identificare valori speciali e catalogare il loro significato.

REVISIONE CHIAVI E ATTRIBUTI

- Verificare la copertura (es. se sono rappresentati tutti i possibili valori).
- Controllare le chiavi.
- Verificare la congruenza fra significato e valore degli attributi,
- Identificare attributi mancanti e campi vuoti.
- Verificare attributi con valori differenti ma significati simili (es. occupazione/professione).
- Verificare l'ortografia dei valori (alcuni valori iniziano con una lettera maiuscola, altri con una lettera minuscola).
- Esaminare gli scostamenti (es. se una deviazione può essere considerata rumore o può rappresentare un fenomeno notevole).
- Verificare la plausibilità dei valori (es. se tutti i campi hanno gli stessi valori o quasi).
- Rivedere gli attributi che possono avere significati conflittuali con il senso comune (es. adolescenti con alti introiti).
- Usare diagrammi di visualizzazione (istogrammi, torte, ecc.) per mostrare le inconsistenze nei dati.

VERIFICA DELLA QUALITÀ DATI NEI FILE

- Verificare quale delimitatore è usato e se è usato consistentemente in tutti gli attributi.
- Verificare la coincidenza del numero di campi in ogni record del file.

RUMORE E INCONSISTENZE TRA LE SORGENTI

- Verificare inconsistenze e ridondanze tra sorgenti diverse.

- Programmare come trattare il rumore.
- Determinare il tipo di rumore (es. rumore bianco, rumore colorato) e quali attributi ne sono affetti.

2.4.3 Preparazione dei dati

Output finale: **Set dei dati**

Questo è l'insieme (o insiemi) dei dati prodotti nella fase di preparazione che verrà usato in fase di modellazione e sul quale verrà eseguita la maggior parte del lavoro di analisi.

Descrizione del set dei dati

Descrizione dell'insieme dei dati che verrà usato per il lavoro di analisi.

Selezione dei dati

Compito: **Selezionare i dati**

Decidere su quali dati devono essere usati per l'analisi. I criteri si basano sulla rilevanza ai fini degli obiettivi DM, sulla qualità, su vincoli tecnologici come il volume o il formato dei dati. Si noti che la selezione dei dati consiste nella selezione di record e attributi nelle tabelle.

Output: **Ragioni fondamentali per l'inclusione / esclusione**

Lista dei dati inclusi ed esclusi e dei motivi delle scelte.

Attività:

- Raccogliere i dati addizionali appropriati (sia da fonti interne che esterne).
- Effettuare test di significato e correlazione per decidere se i campi devono essere inclusi.
- Riconsiderare i criteri di selezione dei dati, (cfr. par. 2.4.2) nel compito *"collezionare i dati"*, alla luce delle esperienze sulla qualità e sulla esplorazione dei dati (può essere necessario includere/escutere altri set di dati).
- Riconsiderare i criteri di selezione dei dati, (cfr. par. 2.4.2) nel compito *"collezionare i dati"*, alla luce delle esperienze sulla modellizzazione dei dati (il modello adottato può esigere altri dati).
- Selezionare sottoinsiemi di dati differenti, nel senso di non ridondanti (es. attributi distinti, attributi che verificano certe condizioni).
- Considerare *tecniche di campionamento*, cioè un metodo per avere campioni dell'insieme dei dati di test (può capitare che lo strumento non sia capace di gestire l'intero insieme di dati di test). Può rivelarsi utile avere campioni

pesati del data set, per dare importanze differenti ad attributi differenti o differenti valori dello stesso attributo.
- Verificare la disponibilità di tecniche per il campionamento dati.
- Documentare le ragioni fondamentali delle scelte di inclusione/esclusione.

Pulizia dei dati

Compito: Pulire i dati

Innalzare la qualità dei dati al livello richiesto dalle tecniche di analisi scelte. Questo può comprendere la selezione di sottoinsiemi di dati puliti, l'inserimento di opportuni valori di default o metodi più ambiziosi, come la stima dei valori mancanti in fase di modellizzazione.

Output: Report sulla pulizia dei dati

Insieme di report che descrivono quali decisioni ed azioni sono state prese per far fronte a problemi di qualità dei dati messi in luce nel compito *"verificare la qualità dei dati"* nella fase di *"comprensione dei dati"*.

Attività:

- Riconsiderare come trattare il tipo di rumore osservato.
- Correggere, rimuovere o ignorare il rumore.
- Decidere come occuparsi dei valori speciali e del loro significato. Il settore dei valori speciali può condurre a strani risultati, perciò deve essere attentamente esaminato. Esempi di valori speciali possono saltar fuori anche da un rapido sguardo a domande che non sono poste o che non trovano risposta. Ad es. il valore '99' per un dato sconosciuto (relativi a stato civile o affiliazione politica). Valori speciali possono anche sorgere a seguito di troncamenti di valori numerici, ad es. il valore '00' per persone con 100 anni di età, oppure '100' per automobili con 100.000 km sul contachilometri.
- Riconsiderare i criteri di selezione dei dati, (cfr. par. 2.4.2) nel compito *"collezionare i dati"*, alla luce delle esperienze sulla pulizia dei dati (può essere necessario includere/escutere altri set di dati).
- Documentare anche il rumore di campi giudicati irrilevanti ai fini degli obiettivi DM, in quanto le circostanze possono cambiare in seguito.

Costruzione dei dati

Compito: Costruire i dati

Questo compito include operazioni costruttive per la preparazione dei dati, come la produzione di attributi derivati (nota) o di nuovi record; trasformazione di valori di attributi già esistenti.

Attività:

- Verificare i vari meccanismi di costruzione di cui dispongono gli strumenti utilizzati.
- Decidere se è più opportuno effettuare la costruzione all'interno del tool o esternamente (ad es. per questioni di ripetibilità ed efficienza).
- Riconsiderare i criteri di selezione dei dati, (cfr. par. 2.4.2) nel compito *"collezionare i dati"*, alla luce delle esperienze sulla costruzione dei dati (può essere necessario includere/escutere altri set di dati).

Output: **Attributi derivati**

Gli attributi derivati sono nuovi attributi dedotti da uno o più attributi già esistenti, ad es. $area = \frac{base * altezza}{2}$.

Attività: ATTRIBUTI DERIVATI

- Decidere se qualche attributo deve essere normalizzato.
- Considerare l'aggiunta di nuove informazioni su attributi di grande importanza, generando nuovi attributi (es. attributi peso, normalizzazioni pesate,…).
- Decidere su come devono essere costruiti gli attributi mancanti (aggregazioni, medie, induzioni).
- Introdurre attributi derivati per facilitare il compito all'algoritmo di modellazione e non al solo scopo di ridurre il numero di parametri in input (ad es. 'reddito pro-capite', può essere più utile di 'reddito familiare').

TRASFORMAZIONI DI ATTRIBUTI SINGOLI

- Considerare la trasformazione di un singolo attributo (es. cambiamento di scala, conversione,…) per andare incontro alle esigenze dello strumento.
- Specificare i passi di trasformazione in base alle tecniche disponibili (es. modifiche alla discretizzazione di un attributo numerico).
- Effettuare i passi di trasformazione. La trasformazione può servire a cambiare intervalli di valori in campi simbolici (es. età in fasce di età) o viceversa.

Output: **Record generati**

Descrivere la creazione di record generati ad hoc. Esempio: creare record di clienti che non hanno effettuato acquisti nell'ultimo anno. Ancora, una volta segmentati i dati, può essere utile generare un record relativo al membro prototipo di ciascun segmento. Non c'è ragione di trovare record del genere nei dati grezzi di partenza, ma per esigenze di modellazione essi possono avere importanza.

Attività:

- Verificare le tecniche disponibili (es. meccanismi per la costruzione dei prototipi dei segmenti).

Integrazione dei dati

Compito: Integrare i dati

Questi sono i metodi con cui l'informazione viene combinata da tabelle o record multipli per creare nuovi record e valori

Output: Dati unificati

La fusione di tabelle si riferisce all'unione (*joining*) di tabelle contenenti informazioni diverse sul medesimo oggetto. Esempio: una catena di vendita al dettaglio ha una tabella con le informazioni generiche di ogni negozio (tipologia della costruzione, quadratura, disposizione sui piani ecc.), una tabella con le informazioni dettagliate di vendita (ricavi, profitti, incrementi/decrementi percentuali rispetto agli anni precedenti ecc.) ed un'altra tabella riguardante le informazioni demografiche dell'area circostante. Ognuna di queste tabelle contiene un record per ogni negozio. Queste tabelle possono essere fuse in una unica contenente un record per ogni negozio, ottenuto dalla combinazione dei campi delle tabelle di partenza.

I dati unificati riguardano anche le aggregazioni. L'*aggregazione* è l'operazione dove nuovi valori sono calcolati ricapitolandone altri da record o tabelle diversi. Ad es. convertire una tabella con un record per ogni acquisto di un cliente in una tabella avente un record per ogni cliente e dei campi con numero di acquisti totali, importo medio d'acquisto, percentuale degli ordini accontati su carta di credito ecc.

Attività:

- Verificare le funzionalità di integrazione, se esse possono integrare le diverse sergenti come richiesto.
- Integrare sorgenti dati e magazzini dati.

- Riconsiderare i criteri di selezione dei dati, (cfr. par. 2.4.2) nel compito *"collezionare i dati"*, alla luce delle esperienze sulla integrazione dei dati (può essere necessario includere/escludere altri set di dati).

Formattazione dei dati

Compito: **Formattare i dati**

Le trasformazioni di formattazione si riferiscono in primo luogo a modifiche sintattiche fatte sui dati che non ne alterano il significato, ma che possono essere richieste dagli strumenti di modellizzazione.

Output: **Dati riformattati**

Insieme dei dati riorganizzato secondo le esigenze dei tools.

Alcuni strumenti presentano requisiti sull'ordine degli attributi, del tipo: il primo campo deve essere chiave primaria; l'ultimo campo è quello dell'attributo sul quale si effettuerà la predizione.

Può essere richiesto anche l'ordinamento dei record: alcuni strumenti richiedono l'ordinamento dei dati sul valore dell'attributo risultato della predizione. Una situazione comune è anche quella in cui inizialmente i record sono ordinati in qualche maniera, ma lo strumento richiede i dati in ordine casuale.

Per di più, ci sono modifiche puramente sintattiche per soddisfare i requisiti dello strumento: rimozioni di virgole in campi testuali all'interno di file organizzati secondo strutture dati delimitati da virgole; cimatura di tutti i valori ad un massimo di 32 caratteri e così via.

Attività:

- Effettuare il riordinamento degli attributi, se richiesto dallo strumento.
- Effettuare il riordinamento dei record, se richiesto dallo strumento.
- Effettuare le modifiche sintattiche, se richieste dallo strumento.
- Riconsiderare i criteri di selezione dei dati, (cfr. par. 2.4.2) nel compito *"collezionare i dati"*, alla luce delle esperienze sulla formattazione dei dati (può essere necessario includere/escutere altri set di dati).

2.4.4 Modellazione

Selezione della tecnica di modellazione

Compito: **Selezionare la tecnica di modellazione**

Come primo passo della fase di modellazione, selezionare l'effettiva tecnica che verrà utilizzata da ora in poi. Sebbene si sia già scelto uno strumento nella fasi di "comprensione del business", questo compito si riferisce ad una specifica tecnica. Ad es. alberi decisionali costruiti con l'algoritmo C4.5 oppure reti neurali con "back propagation". Se sono applicate più tecniche, ripetere questo passo per ognuna separatamente.

Non bisogna dimenticare che non tutte le tecniche e gli strumenti sono disponibili per qualsiasi compito. Per alcune tipologie di problemi, solo alcune tecniche sono appropriate (cfr. par. 3.2). Inoltre, fra queste tecniche vi sono "requisiti politici" ed altri vincoli che possono ulteriormente limitare le possibili scelte. Al limite può accadere che solo uno strumento o tecnica è effettivamente disponibile per il problema in questione, e che tale strumento non sia tecnicamente il migliore.

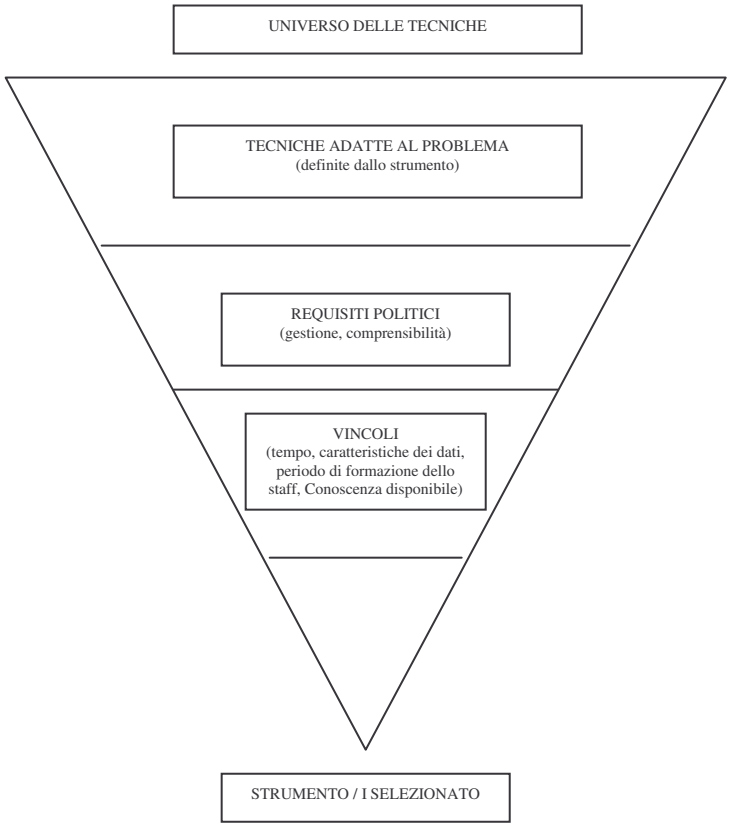


Figura 22: restrizioni e vincoli nella scelta di uno strumento.

Output: **Tecnica di modellazione**

Tecnica o tecniche di modellazione effettivamente usate.

Attività:

- Decidere sulla tecnica da esercitare, tenendo presente lo strumento scelto.

Output: **Assunzioni di modellazione**

Specifiche assunzioni sui dati richieste dalle tecniche di modeling. Ad es. attributi aventi distribuzione uniforme, valori mancanti non permessi, attributi di classe simbolici ecc.

Attività:

- Definire ogni assunzione intrinseca alla tecnica sui dati (qualità, formattazione, distribuzione,…).
- Paragonare tali assunzioni con quelle contenute nel *report sulla descrizione dei dati* (cfr. par. 2.4.2, compito *"Descrivere i dati"*).
- Assicurarsi se queste assunzioni non portino ad un ritorno alla fase di *"Preparazione dei Dati"* (cfr. par. 2.4.3).

Generazione di un piano di test

Compito: Generare un piano di test

Prima di costruire effettivamente un modello, si ha bisogno di una procedura o meccanismo per testare la qualità e la validità del modello. Per esempio, a scopi di data mining supervisionato (cfr. par. 3.1) come la classificazione, è frequente usare i tassi di errore come misura della qualità dei modelli. Perciò, una volta separati i dati in set di apprendimento (training set) e set di test (test set), si costruisce il modello sul training set e ne si stima la qualità sul test set.

Output: Piano di test

Questo documento descrive il piano stilato per l'istruzione, il test, e la valutazione del modello. Un componente primario sono le decisioni su come dividere l'insieme di dati a disposizione in set di apprendimento, set di test e set di valutazione.

Attività:

- Verificare i casi di test per ogni obiettivo DM separatamente.
- Decidere i passi necessari (es. numero di iterazioni ecc.).
- Preparare i dati per il test.

Costruzione del modello

Compito: Costruire il modello

Eseguire lo strumento di modellazione sui dati preparati per creare uno o più modelli.

Output: Settaggio dei parametri

Questo report elenca i parametri ed i relativi valori scelti, insieme con le motivazioni fondamentali delle scelte operate. In molti strumenti di modellazione vi sono un gran numero di parametri che devono essere regolati.

Attività:

- Settare i parametri iniziali.
- Motivare la scelta dei valori.

Output: **Modelli**

I modelli realmente prodotti dagli strumenti (non è un report).

Attività:

- Eseguire la tecnica scelta sui dati di ingresso per costruire il modello.
- Processare i risultati di mining (es. editando le regole, raffigurando gli alberi).

Output: **Descrizione dei modelli**

Report descrittivo del modello risultante, della sua interpretazione e delle difficoltà incontrate per la sua comprensione.

Attività:

- Descrivere le caratteristiche del modello che potrebbero essere utili in futuro.

Accertamento del modello

Compito: **Accertare il modello**

L'ingegnere di data mining interpreta i modelli sulla base della sua conoscenza del dominio applicativo, dei criteri di successo e del piano di test desiderato. Questo compito interferisce con la successiva fase di valutazione (evaluation phase). Premesso che l'ingegnere DM valuta il successo di un'applicazione di modellazione in termini tecnici, egli successivamente contatta gli analisti di business e gli esperti di dominio, in modo da discutere sui risultati del processo nel contesto del business. Si noti che questo compito considera i soli modelli prodotti, laddove la fase di valutazione prende in esame tutti gli altri risultati ottenuti nel corso dell'intero progetto.

L'ingegnere DM, inoltre, classifica i risultati. Egli stima il modello secondi i criteri di valutazione, tenendo conto, finchè possibile, gli obiettivi ed i criteri di successo del business.

Nella maggior parte dei progetti di DM, l'ingegnere applica una singola tecnica più di una volta, oppure genera risultati distinti applicando tecniche diverse. In tal caso si procede alla comparazione dei risultati secondo i criteri di valutazione.

Iterare questo ed il precedente compito fino a che non si sono raggiunte la validità e la qualità desiderata del modello.

Output: **Stima del modello**

Sommario dei risultati del compito, lista delle qualità dei modelli generati (ad es. in termini di accuratezza) e messa in evidenza delle relazioni reciproche.

Attività:

- Valutare i risultati secondo i criteri di valutazione.
- Testare i risultati secondo la strategia di testing.
- Interpretare i risultati.
- Stilare una classifica dei risultati secondo i criteri di valutazione.
- Selezionare i modelli migliori.
- Interpretare i risultati in termini di business (se possibile a questo stadio).
- Verificare la plausibilità del modello.
- Verificare l'impatto sugli obiettivi DM.
- Verificare il modello rispetto alla conoscenza di partenza, per vedere se le informazioni scoperte sono nuove ed utili.
- Verificare l'attendibilità dei risultati.
- Analizzare le capacità per il deployment di ogni risultato.
- Se vi è una descrizione verbale del modello generato (es. tramite regole), verificare la logicità e la verosimiglianza delle regole e se vanno contro il senso comune.
- Accertare i risultati.

Output: **Correzione del settaggio dei parametri**

Revisione e regolazione dei parametri di setting secondo la stima effettuata per successive esecuzioni del compito *"costruzione del modello"* .

Attività:

- Resettare i parametri per ottenere un modello migliore.

2.4.5 Valutazione

Un buon modo per definire l'output totale di un progetto di data mining è usare l'equazione:

$RISULTATI = MODELLI + SCOPERTE$.

In questa equazione si mette in risalto che i prodotti in uscita al progetto non sono solo i modelli generati, sebbene importanti, ma anche i ritrovamenti (findings) di conoscenza giudicati importanti al fine di raggiungere gli obiettivi di business o di porre nuove questioni precedentemente ignorate o, ancora, di svelare effetti collaterali (es. problemi di qualità dei dati scoperti durante le pratiche di mining).

Valutazione dei risultati

Compito: Valutare i risultati

Il precedente passo di valutazione ha a che fare con l'accuratezza e la generalità del singolo modello prodotto. In questa sede si osserva il grado con il quale il modello va incontro agli obiettivi di business, ovvero si misura la sua efficacia. Contestualmente si determinano le eventuali ragioni di possibili deficienze e inattitudini del modello.

Una interessante opzione di valutazione è quella di testare l'applicazione dei risultati dedotti dal modello in situazioni reali, ovviamente se ciò è possibile e il tempo e il budget lo permettono.

Durante l'osservazione dei risultati di DM, tutte le conclusioni non necessariamente relazionate agli obiettivi di business originari possono inaugurare nuove discussioni, svelare nuove informazioni e suggerire direzioni future.

Output: Accertamento dei risultati di DM secondo i criteri di successo di business

Sommario dei risultati in termini di criteri di successo del business, incluso un resoconto finale su come il progetto va incontro agli obiettivi finali di business.

Attività:

- Comprendere i risultati di DM.
- Interpretare i risultati in termini dell'applicazione.
- Valutare l'impatto dei risultati sugli obiettivi DM.
- Valutare i risultati DM rispetto alla conoscenza di base, per vedere se la conoscenza scoperta è nuova ed utile.
- Valutare i risultati secondo i criteri di successo di business.
- Creare una classifica dei risultati secondo i criteri di successo di business.
- Verificare l'impatto dei risultati sull'obiettivo iniziale dell'applicazione.

Output: Modelli approvati

Sottoinsieme dei modelli generati che soddisfano i criteri di selezione.

Revisione di processo

Compito: Rivedere il processo

A questo punto il modello può sembrare soddisfacente in relazione agli obiettivi di business. È raccomandata comunque una rivisitazione più approfondita del processo DM istanziato per notare fattori rilevanti o compiti che sono sfuggiti. Questa revisione contempla anche questioni di assicurazione della qualità: "Si è costruito il modello correttamente ? si sono utilizzati gli attributi permessi e disponibili anche per analisi future ?".

Output: **Revisione di processo**

Rendiconto della revisione di processo, con messa in risalto di attività omesse o che devono essere ripetute.

Attività:

- Analizzare il processo DM.
- Identificare gli insuccessi.
- Identificare i passi fuorvianti.
- Identificare possibili azioni alternative, insieme con percorsi inattesi del processo.

Determinazione delle prossime azioni

Compito: **Determinare le prossime azioni**

Secondo l'accertamento dei risultati e la revisione di processo, si decide su come il progetto deve procedere. Si decide se il progetto volge al termine o se è appropriata la fase di "deployment", se iniziare iterazioni supplementari o settare nuovi progetti. A supporto di tali decisioni si analizzano le risorse residue e la copertura del budget.

Output: **Lista delle possibili azioni**

Lista delle azioni da scegliere insieme ai pro e i contro di ogni opzione.

Attività:

- Analizzare il potenziale esplicativo di ogni risultato.
- Stimare il potenziale per migliorie del processo corrente.
- Verificare la presenza di risorse per sviluppare iterazioni extra del processo.
- Segnalare prosecuzioni alternative per il processo.
- Rifinire il piano di processo.

Output: **Decisioni**

Decisioni definitive sulle azioni da intraprendere, con le giustificazioni delle scelte.

Attività:

- Classificare le possibili azioni.
- Selezionare una delle possibili azioni.
- Documentare la scelta.

2.4.6 Deployment

Pianificazione del deployment

Compito: Pianificare il deployment

Per esporre i risultati raggiunti dal DM, questo compito comprende un'ulteriore valutazione dei risultati per concludere una strategia per la spiegazione (deployment). Se sono state identificate procedure generali per la creazione dei modelli, esse vengono illustrate qui.

Output: Piano di deployment

Sommario della strategia di deployment, dei passi necessari e di come eseguirli.

Attività:

- Sviluppare piani alternativi per il deployment.
- Identificare tutti i possibili problemi durante la fase di deployment dei risultati.

Pianificazione di monitoraggio e manutenzione

Compito: pianificare monitoraggio e manutenzione

Monitorare e mantenere sono problematiche importanti se i risultati di DM diventano parte delle attività quotidiane di Business Intelligence (BI).

Una preparazione meticolosa di una strategia di manutenzione aiuta ad evitare lunghi periodi di uso non corretto dei risultati.

Per visionare l'attività di deployment dei risultati, il progetto necessita di un piano accurato di monitoraggio particolarizzato sulla specifica strategia di deployment

Output: Piano di monitoraggio e manutenzione

Documento descrittivo delle strategie di monitoraggio e manutenzione.

Attività:

- Verificare gli aspetti dinamici (es. cosa può cambiare nell'ambiente del business, dell'applicazione, delle risorse che si stanno struttando).
- Notificare quando i risultati di DM non devono essere più usati e quali sono le conseguenze derivanti dal disuso dei modelli.
- Documentare gli eventuali cambiamenti nel tempo degli obiettivi di business di uso del modello rispetto a quelli originari.
- Sviluppare il piano di monitoraggio e manutenzione.

Produzione del rapporto finale

Compito: Produrre il rapporto finale

Alla fine del progetto, il progettista capo e la sua squadra scrivono la relazione finale. Essa dipende sostanzialmente dal piano di deployment: se il rapporto deve essere un semplice

compendio del progetto e delle sue esperienze oppure una presentazione dettagliata e professionale dei risultati.

Output: **Rapporto finale**

Resoconto finale del progetto di data mining, comprendente tutti la documentazione prodotta durante il processo, in aggiunta all'organizzazione e presentazione dei risultati.

Attività:

- Determinare quali rapporti sono desiderati (es. presentazione su slides, sommario gestionale, dettaglio delle scoperte, spiegazione dei modelli,…).
- Analizzare come e quanto gli obiettivi di DM sono stati raggiunti.
- Identificare i gruppi destinatari dei report.
- Tracciare la struttura ed il contenuto dei report.
- Selezionare le scoperte (findings) che devono essere incluse nei report.
- Scrivere il/i report.

Presentazione finale

Incontro finale al termine del progetto, dove i risultati sono presentati verbalmente al cliente.

Attività:

- Identificare il gruppo destinatario della presentazione finale.
- Selezionare quali elementi del rapporto finale devono essere inclusi nella presentazione finale.

Revisione di progetto

Compito: **Rivedere il progetto**

Giudicare cosa è andato bene e cosa è andato male. Segnalare le fasi che devono essere migliorate e valorizzate.

Output: **Documentazione di esperienza**

Ricapitolazione delle esperienze più importanti fatte durante il progetto. Ad esempio tranelli, approcci fuorvianti, suggerimenti alla selezione delle tecniche di DM più adatte in contesti similari.

In un progetto ideale, la documentazione di esperienza concerne tutti i rapporti scritti dai singoli membri del gruppo di lavoro durante le fasi ed i compiti.

Attività:

- Intervistare tutte le persone significative coinvolte nel progetto e porre domande riguardo la loro esperienza accumulata durante il progetto.

- Intervistare gli utenti finali che lavorano con i risultati di DM; misurare il loro grado di soddisfazione; chiedere consigli per ampliamenti e miglioramenti; fornire supporto se richiesto.
- Descrivere le retroazioni del processo.
- Analizzare il processo (cose ha funzionato e cosa no, errori commessi, lezioni imparate,…).
- Astrarsi dai dettagli per rendere l'esperienza riusabile per progetti futuri.

Cap. III

DATA MINING:

METODOLOGIE E TECNICHE

Lo stadio cruciale di Data Mining nel processo KDD spesso richiede applicazioni ripetute di particolari metodi. In questo capitolo si tenta di dare una panoramica degli obiettivi primari del DM; una descrizione dei metodi e dei modelli che si adottano per raggiungere tali obiettivi ed una breve trattazione degli algoritmi che realizzano tali metodi.

3.1 Obiettivi del Data Mining

Gli obiettivi sono sostanzialmente definiti dall'uso che si intende fare del sistema.
Possiamo subito distinguere in due tipologie di obiettivo: Verifica (*Verification Model*) e Indagine (*Discovery Model*). Con la verifica il sistema si limita a verificare le ipotesi formulate dall'utente testandone la validità sui dati. L'enfasi è sull'utente-analista che è responsabile della formulazione dell'ipotesi e dell'emissione dell'interrogazione sui dati per affermare o negare l'ipotesi.
Per la divisione marketing di un'azienda, ad esempio, che ha a disposizione un budget limitato per il lancio di un nuovo prodotto, è importante identificare la sezione di popolazione più propensa all'acquisto dell'articolo. L'utente formula un'ipotesi per identificare i potenziali clienti e le caratteristiche in comune. I dati storici sugli acquisti dei clienti e le informazioni demografiche verranno recuperate per rivelare acquisti simili e le caratteristiche condivise dai clienti, che saranno usati come obiettivo della campagna promozionale, ad es. una campagna postale.
La pecca di questo approccio è che nessuna nuova informazione è prodotta, ma piuttosto le interrogazioni restituiranno sempre record verificanti o meno l'ipotesi. Qui il processo di ricerca è iterativo nel senso che l'output è revisionato e riesaminato; si pongono nuove domande e vengono riformulate le ipotesi allo scopo di raffinare la ricerca. L'utente scopre nuovi fatti sui dati utilizzando una varietà di tecniche come query articolate, analisi multidimensionale e visualizzazione dati per guidare l'esplorazione. Ciò tipicamente viene fatto su sistemi OLAP con gli strumenti standard incorporati.

Con il modello di Indagine il sistema trova autonomamente nuovi pattern. I dati sono setacciati alla ricerca di comportamenti frequenti, trend e generalizzazioni senza l'intervento dell'utente. Gli strumenti di DM permettono l'individuazione di molti "fatti" in un tempo relativamente breve.

La reale utilità del processo di DM è quello di scoprire andamenti nei dati che sarebbero altrimenti invisibili "ad occhio nudo", quindi il modello di scoperta ne rappresenta la vera essenza.

Possiamo ulteriormente suddividere gli obiettivi di scoperta in predittivi, dove il sistema scopre i pattern per predire comportamenti futuri di alcune entità, e descrittivi, dove il sistema presenta i patterns trovati all'utente in forma umanamente intelligibile.

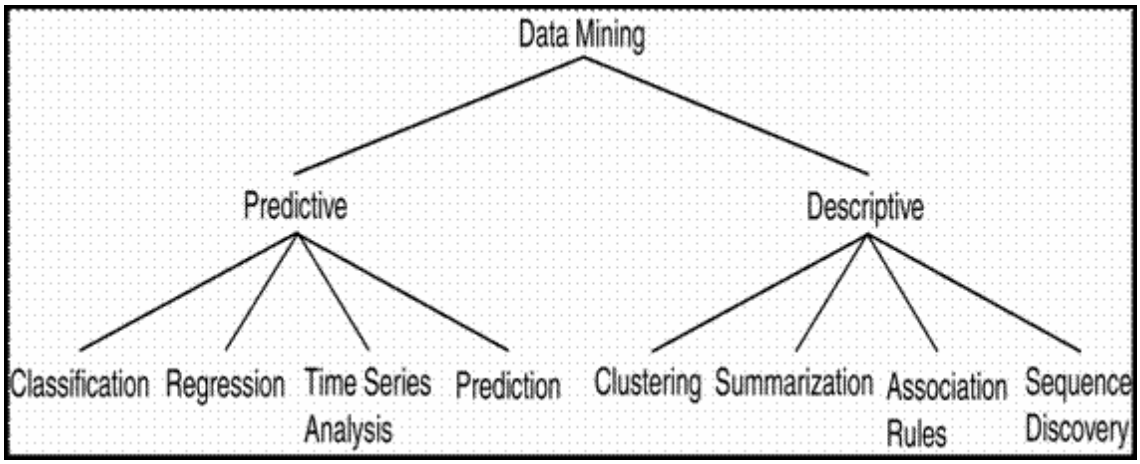


Figura 23: Tipologie di Data Mining.

Il DM concerne l'adattamento di modelli ai dati osservati ed i modelli adattati giocano il ruolo della conoscenza dedotta. Se i modelli riflettono elementi utili e significativi essi costituiscono una parte del processo di conoscenza, che comunque non può fare a meno del giudizio umano.

Di conseguenza anche gli strumenti di mining in commercio si dividono secondo queste due grandi categorie: gli strumenti di indagine aiutano a identificare tendenze nei dati, mentre gli strumenti di verifica sono quelli classici per l'analisi statistica. Se gli strumenti di indagine possono portare a interessanti scoperte sulla propria attività, non possono spiegare perché e nemmeno se, queste scoperte sono valide e utili. La maggior parte degli strumenti di indagine esegue test per analizzare le differenze tra gruppi. Spesso questi test portano a conclusioni sbagliate, per via della loro natura probabilistica. Gli strumenti di verifica servono anche a convalidare le scoperte fatte in sede di indagine, per garantire decisioni corrette.

Due sono i formalismi matematici usati per l'adattamento: l'approccio statistico, che permette l'individuazione di effetti non deterministici, e l'approccio logico, per modelli puramente

deterministici. In questa sede si focalizzerà sull'approccio statistico al DM, che è di gran lunga il più adottato nelle applicazioni pratiche, data la naturale presenza di incertezza nei processi di generazione dati nel mondo reale.

La maggior parte dei metodi di DM sono tecniche consolidate che provengono dall'intelligenza artificiale, dal riconoscimento di strutture e dalla statistica: classificazione, clusterizzazione, regressione e cosi via. La grande varietà degli algoritmi che "sta sotto" a queste diciture è talmente ampia da confondere anche un analista esperto; e la maggior parte delle tecniche citate in letteratura rappresentano solo quelle fondamentali.

L'apprendimento induttivo (inductive learning), che fa da base concettuale alla presente e ad altre discipline, è l'inferenza di informazioni dai dati per il processo costruttivo del modello dove l'ambiente, ovvero il database, è analizzato in modo da ritrovare i pattern. Oggetti simili sono raggruppati in classi e sono formulate regole con le quali è possibile stimare la classe di appartenenza di oggetti non classificati. Questo processo identifica classi in modo che ognuna di esse possiede un pattern di valori che forma la descrizione della classe. Se la natura dell'ambiente è dinamica il modello deve essere adattativi.

Il processo di inferenza della conoscenza ha due principali strategie:

Apprendimento supervisionato: è l'apprendimento basato su esempi che vengono forniti al sistema per la costruzione del modello che definisce le classi. Vengono insomma fornite classi esemplari. Una volta formulata la descrizione che forma la regola, essa può essere utilizzata per predire la classe di nuovi oggetti.

Apprendimento non supervisionato: è l'apprendimento dall'osservazione e dalla scoperta. Il sistema è munito degli oggetti ma nessuna classe è definita, cosicché esso deve osservare gli esempi o riconoscere i pattern autonomamente. Il risultato consiste in un insieme di descrizioni di classe, una per ogni classe osservata nell'ambiente.

3.2 Metodi per tipologie di problemi

I due obiettivi principali di alto livello del DM, come si è visto, sono la predizione e la descrizione. Come prima accennato la predizione concerne l'uso di alcune variabili o campi dei dati per prevedere valori futuri (e sconosciuti) di altre variabili d'interesse, mentre la descrizione si concentra sul ritrovamento di forme descrittive dei dati di facile comprensione per l'utente. Tuttavia la distinzione fra DM predittivo e descrittivo non è nitida (alcuni modelli predittivi possono essere considerati descrittivi per il loro grado di comprensibilità e viceversa), ma è importante per la comprensione dell'obiettivo ultimo dell'applicazione da

sviluppare. L'importanza relativa fra predizione e descrizione può variare considerevolmente a seconda della particolare applicazione.
Gli obiettivi della predizione e della descrizione possono essere raggiunti tramite l'utilizzo di svariate tecniche e metodi che caratterizzano la tipologia del problema.

3.2.1 Sommarizzazione

La descrizione dei dati o sommarizzazione (summarization) mira alla descrizione concisa della natura dei dati, in forma sia elementare che aggregata. Questo dà all'utente una veduta generale della struttura dei dati. Talvolta la sommarizzazione può essere la meta finale di un progetto DM, ma il più delle volte essa rappresenta un sotto-obiettivo del processo, tipicamente nelle fasi iniziali. Essa quindi si trova nella parte inferiore della scala dei problemi di Data Mining.
Quando un processo viene istanziato, l'utente spesso ignora l'obiettivo preciso dell'analisi e non conosce la struttura dei dati. Un'analisi esplorativa preliminare dei dati può aiutare a comprenderne la natura ed a generare prime ipotesi sulle informazioni nascoste. Semplici statistiche descrittive e tecniche di visualizzazione provvedono a fornire una prima vista dei dati, come la tabulazione di medie e deviazioni standard di ogni campo di interesse. Ad esempio, ad un venditore possono essere utili informazioni sulla distribuzione dell'età dei clienti e della residenza per individuare i gruppi potenzialmente reattivi ad una campagna pubblicitaria.
La sommarizzazione tipicamente viene effettuata in combinazione con altri problemi di DM. Essa può anche portare alla definizione di segmenti significativi nei dati, che vengono poi descritti e ricapitolati.
Si intuisce quindi la natura preliminare della sommarizzazione dei dati, che è consigliabile effettuare prima di entrare nel merito dello specifico problema DM. Infatti la descrizione dei dati costituisce un compito preciso nella fase di comprensione dei dati nel modello CRISP-DM (cfr. par. 2.4.2).
La sommarizzazione gioca un ruolo importante anche nella presentazione dei risultati finali. Le conclusioni di altri problemi di mining, sia predittivi che descrittivi, possono essere considerate sommarizzazioni ad un livello concettuale più alto.
La maggior parte dei sistemi OLAP, sistemi di reporting, package statistici incorporano funzionalità di questo tipo, ma non forniscono strumenti per modellazioni avanzate. Anche la semplice esplorazione di dati multidimensionali tramite le operazioni fondamentali (cfr. par. 1.4.3), se il problema non è complesso, può essere sufficiente a tale scopo. Se l'obiettivo del

progetto comprende la sola sommarizzazione si può optare per una scelta del genere senza munirsi di strumenti di mining specifici.
Va detto comunque che tale metodo può servirsi di tecniche più sofisticate come la deviazione delle regole di sommario (Agrawal et al. 1996), tecniche di visualizzazione multivariabile (Zembowicz e Zytkow 1996).

3.2.2 Segmentazione/Clusterizzazione

La segmentazione, chiamata anche clusterizzazione (clusterization), mira alla separazione dei dati in sottogruppi o classi significativi. Tutti i membri di un sottogruppo condividono caratteristiche comuni. Ad esempio, nell'analisi dei carrelli della spesa (market basket analysis) si possono definire segmenti (o clusters) di carrelli dipendenti dagli articoli in essi contenuti.
In fig. 24 abbiamo un semplice data set artificiale su 2 dimensioni. Ogni punto nel piano rappresenta un cliente di una banca, sull'asse delle ascisse è riportato il reddito e sulle ordinate i debiti personali (ipoteche, rate per il pagamento dell'automobile ecc.). La banca può voler determinare sottogruppi di clienti aventi un profilo comune per decidere se concedere un prestito oppure no. In figura , a titolo illustrativo, sono riportati 3 cluster differenti che possono essere caratterizzati da una descrizione dettagliata. Si noti che i cluster si sovrappongono, permettendo ad un punto di appartenere a più sottogruppi.

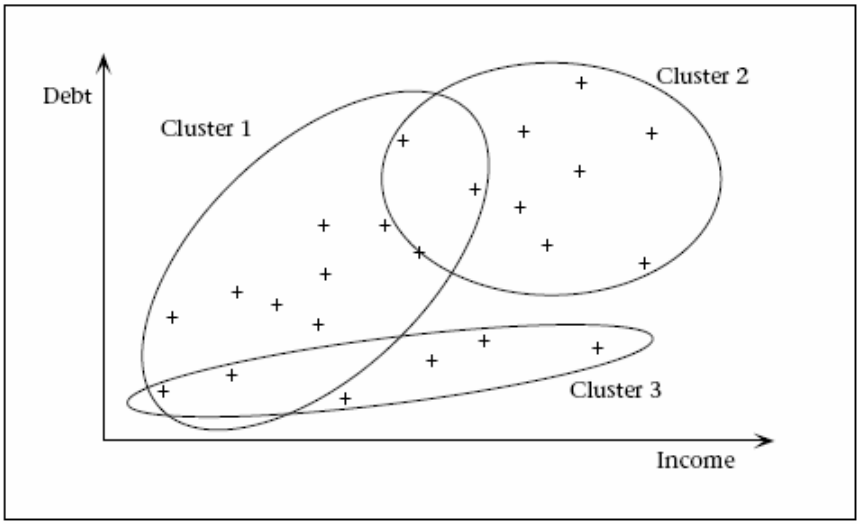


Figura 24: un semplice esempio di clustering per clienti bancari.

La segmentazione può essere effettuata manualmente o automaticamente. L'analista può ipotizzare le caratteristiche di alcune classi alla luce della rilevanza verso le questioni di business in base alla sua conoscenza pregressa o grazie ai risultati di una sommarizzazione.

Avremo quindi un apprendimento supervisionato da parte del sistema: le caratteristiche delle classi sono fornite dall'utente.

Comunque sono disponibili tecniche di clusterizzazione automatica che possono individuare strutture nascoste nei dati, inaspettate fino a quel momento, che permettono ulteriori segmentazioni.

Si ha in tal caso un apprendimento non supervisionato, dove è il sistema a scovare le caratteristiche dei sottogruppi ed i criteri di appartenenza degli oggetti nei dati. Tali criteri sono valutati in base ad alcune *metriche*, che stabiliscono la similarità e la prossimità degli oggetti. Il punto chiave sta nel tradurre delle misure intuitive di similarità in misure quantitative. Gli oggetti sono spesso suddivisi in un insieme di cluster esaustivi e/o mutuamente esclusivi.

Vi sono vari approcci per formare i cluster, uno di essi è comporre delle regole che dettano l'appartenenza al gruppo in base al livello di similarità fra gli oggetti. Per una applicazione bancaria, una semplice regola può essere la seguente:

```
if STATO = coniugato and REDDITO > 10000
      and CASA_PROPRIETA = si
then TIPO_INVESTIMENTO = buono
```

Un altro approccio è quello di costruire delle set functions che misurano alcune proprietà delle partizioni in base a dei parametri, come nel caso delle reti neurali.

La segmentazione può essere essa stessa un problema di DM. L'individuazione dei segmenti può essere l'obiettivo principale dell'attività di mining. Ad esempio, tutte le persone aventi lo stesso codice di avviamento postale (CAP), con età superiore all'età media e percepenti uno stipendio possono essere l'obiettivo di una campagna pubblicitaria a mezzo posta di un'assicurazione sull'assistenza sanitaria a domicilio.

C'è da dire che, spesso, la segmentazione è solo un passo verso la risoluzione di problemi di altro tipo. L'obiettivo può anche essere il contenimento delle dimensioni dei dati per gestirli più agilmente o il ritrovamento di insiemi omogenei di dati più facilmente analizzabili. Tipicamente, in grandi volumi di dati le varie influenze si sovrappongono le une alle altre ed oscurano i pattern di interesse, quindi un appropriato raggruppamento rende il compito più facile. Si pensi a voler trovare le dipendenze fra gli articoli contenuti in milioni di cestelli della spesa, una buona parte potrebbero essere non significative o addirittura casuali. E' più facile, e verosimilmente più utile, identificare dipendenze in segmenti d'interesse di tali cestelli, come cestelli con importo alto, o relativi ad uno specifico periodo temporale.

Tecniche appropriate:

- Tecniche di clustering
- Reti neurali
- Tecniche di visualizzazione

3.2.3 Descrizione dei concetti

La descrizione dei concetti mira ad una descrizione comprensibile delle classi. L'obiettivo non è sviluppare modelli completi con una alta accuratezza di predizione, ma di riuscire a vedere a fondo.

La descrizione dei concetti ha una stretta relazione sia con la segmentazione che con la classificazione. La segmentazione può portare ad una enumerazione degli oggetti appartenenti ad una classe senza una descrizione esplicativa. Tipicamente la segmentazione avviene prima della descrizione dei concetti. Alcune tecniche, come il clustering concettuale, effettuano allo stesso tempo sia la segmentazione che la descrizione dei concetti.

La descrizione concettuale può essere utilizzata anche a scopi di classificazione (cfr. par. 3.2.4). D'altro canto la classificazione può produrre modelli comprensibili che possono essere considerati descrizioni di concetti. La distinzione fondamentale è che la classificazione vuole essere completa, nel senso che è applicata a tutti i casi della sottopopolazione selezionata, mentre la descrizione dei concetti può non esserlo. E' sufficiente che essa descriva solo le caratteristiche salienti del relativo insieme di dati.

Tecniche appropriate:

- Metodi di induzione delle regole
- Clustering concettuale

Nel caso di una azienda che voglia appurare la fedeltà dei clienti in base a dei parametri, delle possibili regole di induzione sono:

```
if SESSO = M and ETA > 51 then CLIENTE = fedele
if SESSO = F and ETA > 21 then CLIENTE = fedele
if PROFESSIONE = manager and ETA < 51 then CLIENTE = non_fedele
if SESSO = M and STATO_CIVILE = celibe and ETA < 40 then CLIENTE =
non_fedele
```

3.2.4 Classificazione

Classificare un insieme di oggetti (record), caratterizzati da alcuni attributi, significa determinare la loro appartenenza ad alcune classi ed etichettarli opportunamente. L'etichetta della classe è un valore discreto (simbolico) ed è noto per ogni oggetto. L'obiettivo è costruire un modello di classificazione, chiamato classificatore, che assegni una etichetta ad oggetti non

ancora osservati o etichettati. I modelli di classificazione sono i più usati per la modellazione predittiva.

Le etichette di classe possono essere date in anticipo (definite dall'utente) o determinate da una segmentazione. Inoltre la segmentazione può servire a dimensionare l'insieme di dati da classificare.

La classificazione è una delle tipologie di problemi che più si presenta nelle applicazioni. Molti problemi di DM possono essere ricondotti a problemi di classificazione. Ad esempio, l'assegnazione di un punteggio ad un nuovo cliente da parte di una banca serve a stimare il rischio di credito su un finanziamento. Questo può essere visto come un problema di classificazione creando 2 classi: clienti buoni e clienti cattivi. Il modello di classificazione è costruito a partire dai clienti esistenti, in base agli attributi che ne descrivono il comportamento. Tale modello è usato poi per assegnare al nuovo cliente una delle etichette.

Anche problemi di predizione possono essere trasformati in problemi di classificazione discretizzando i valori continui delle etichette di classe. Nel caso si effettui una predizione su un attributo numerico, gli intervalli discreti sono usati come etichette in luogo dell'esatto valore numerico. Alcune tecniche applicate a taluni casi producono da sole classi comprensibili che non necessitano della descrizione del concetto. Vi è anche una connessione con l'analisi delle dipendenze (cfr. par.3.2.6), in quanto tipicamente i classificatori delucidano e mettono in evidenza le dipendenze tra gli attributi.

Prima di costruire il modello è importante esaminare le deviazioni e gli outliers (valori isolati) nei dati. Essi ,infatti, possono oscurare i pattern e compromettere l'aderenza del modello ai dati. Viceversa i modelli possono essere utili a identificare le deviazioni ed interpretarne il significato.

Esempi di metodi di classificazione usati nelle applicazioni includono la classificazione di andamenti e tendenze nei mercati finanziari e l'identificazione automatica di oggetti in grandi database di immagini.

Prendendo come esempio ancora il caso di una banca, nella figura è rappresentato un insieme di punti: le 'x' indicano clienti che in passato sono venuti meno ai prestiti concessi, mentre le 'o' indicano clienti corretti che hanno rispettato i termini. Tale insieme di dati è partizionato secondo un limite decisionale lineare in due classi; la banca userà queste classi per decidere se concedere prestiti in futuro ai candidati. Si noti che con un limite lineare tipicamente non è possibile dividere esattamente le classi (specialmente al crescere dei dati), perciò spesso si utilizzano limiti di ordine superiore.

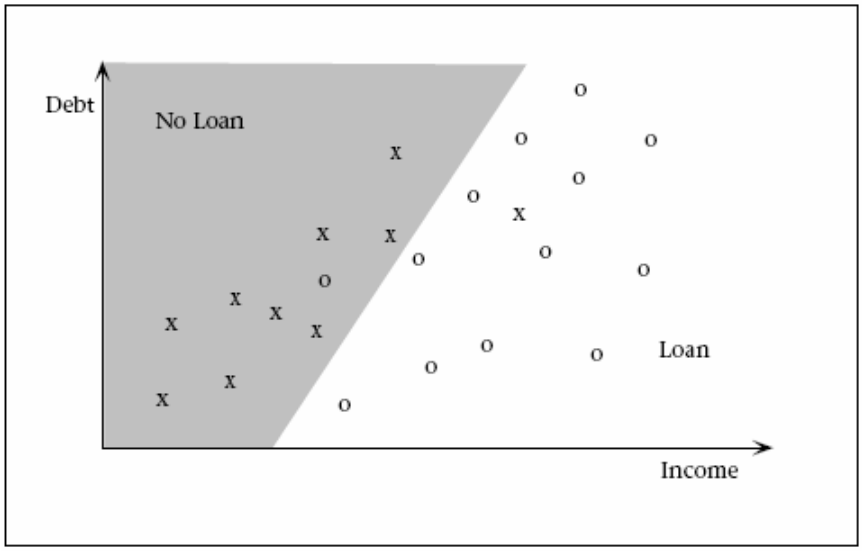


Figura 25: esempio di classificazione con limite lineare.

La classificazione è un processo che si snoda in due fasi, la prima delle quali è la costruzione di un modello da un insieme dati predeterminato. Il modello è sviluppato esaminando le tuple descritte dai loro attributi, dove l'appartenenza di una tupla ad una classe è nota ed è determinata da uno specifico attributo detto attributo etichetta di classe. L'insieme di tuple necessario per istruire il modello è detto insieme di apprendimento (training data set). Siccome è nota la classe di cui ogni tupla fa parte, questo è un tipo di apprendimento supervisionato; in contrasto con l'apprendimento non supervisionato, dove le etichette di classe sono sconosciute e spesso accade che non si conosce nemmeno il numero di classi distinte.

Tipicamente il modello sviluppato viene descritto in termini di regole di classificazione, alberi decisionali o formule matematiche. Ad esempio, avendo a disposizione informazioni sul credito bancario dei clienti, si possono classificare in base al loro affidamento.

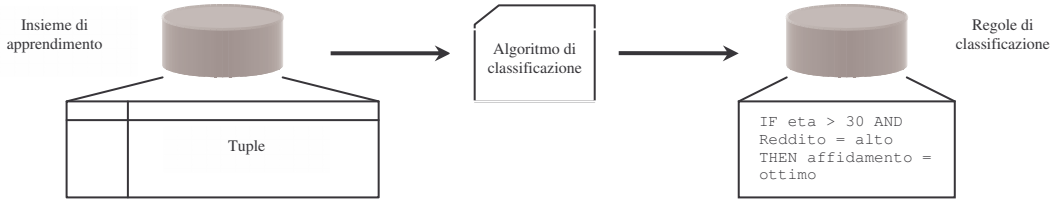


Figura 26: Passi di apprendimento per un modello di classificazione.

In fig. 26 è schematizzata l'attività di apprendimento del modello.

Nella seconda fase il modello è usato per la classificazione. In primo luogo viene misurata l'accuratezza del modello, a tale scopo viene selezionato casualmente dai dati un insieme di test, costituito da tuple ancora etichettate. Il modello classifica tale insieme e l'accuratezza viene ponderata osservando le etichette predette e quelle esistenti. Si potrebbe osservare che

tale misura potrebbe essere fatta sul training set, ma ciò non è consigliabile in quanto il modello tende a sovradattarsi ad esso; in altre parole il modello potrebbe incorporare anomalie peculiari nei dati usati per l'apprendimento che non sono presenti in altri campioni, in tal caso la misura dell'accuratezza potrebbe rivelarsi troppo ottimistica.
Una volta che l'accuratezza del modello può considerarsi accettabile, si può procedere alla classificazione di nuovi dati futuri o precedentemente inosservati, come rappresentato in fig. 27.

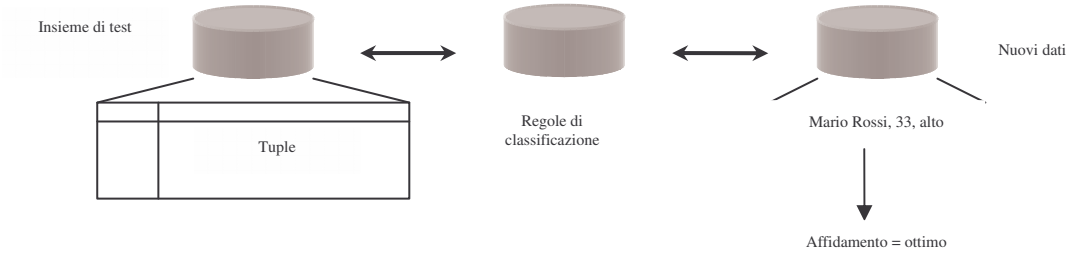


Figura 27: Fase di classificazione su dati attuali

Tecniche appropriate:

- Analisi discriminante
- Classificatore di Bayes semplice
- Reti bayesiane
- Metodi di induzione delle regole
- Alberi decisionali
- Reti neurali
- K-nearest neighbour
- Algoritmi genetici

3.2.5 Predizione

Un altro tipico problema che si presenta in un ampio raggio di applicazioni è la predizione. Essa è molto simile alla classificazione. La differenza fondamentale e che nella predizione l'attributo target (la classe) non è un valore discreto ma continuo, ciò significa che l'obiettivo della predizione è trovare il valore numerico dell'attributo oggetto della stima per record non ancora esaminati. In letteratura questo tipo di problema è chiamato anche regressione. Quando la predizione ha a che fare con dati di serie temporali è detta previsione.
Ad esempio le entrate annuali di una compagnia internazionale sono correlate ad altri attributi come il tasso di inflazione, il tasso di cambio, le campagne pubblicitarie ecc. Avendo a

disposizione questi valori (o una loro stima) la compagnia può stimare il suo reddito atteso per il prossimo anno.

Considerando ancora il caso bancario, in fig. 28 è rappresentata una semplice regressione lineare dove il debito è visto come funzione lineare del reddito. Come si vede l'adattamento è scarso in quanto la correlazione fra le due variabili è debole.

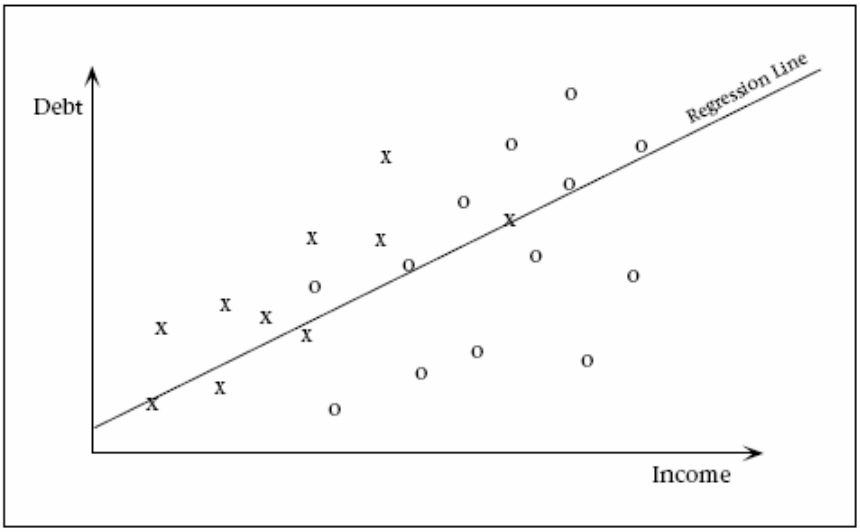


Figura 28: esempio di stima tramite regressione lineare.

Parimenti, un'applicazione di predizione potrebbe essere la stima dell'importo di finanziamenti richiesti dai clienti (attributo numerico continuo) della banca sulla base delle informazioni contenute nel DB.

I metodi di classificazione e predizione possono essere comparati e valutati in base ai seguenti criteri:

- Accuratezza: la capacità del modello di prevedere correttamente i valori delle etichette o degli attributi numerici.
- Velocità: i costi di calcolo nella generazione ed uso del modello.
- Robustezza: l'abilità del modello nel fare predizioni corrette in presenza di rumore nei dati o valori mancanti.
- Scalabilità: le prestazioni del modello su grandi moli di dati.
- Interpretabilità: il livello di comprensione ed intuibilità del modello.

3.2.6 Analisi delle dipendenze

L'analisi delle dipendenze consiste nel trovare un modello che descriva le dipendenze, ovvero le associazioni fra oggetti dato o eventi. Le dipendenze possono essere usate per predire il valore di un dato date le informazioni su altri dati. Sebbene possano essere usate per la modellazione predittiva, le dipendenze si prestano meglio alla comprensione dei dati.

Le associazioni sono un caso particolare di dipendenze che recentemente sono divenute molto popolari. Esse possono descrivere e dare una misura delle affinità fra gli oggetti, ad esempio eventi che si presentano frequentemente insieme. Un tipico scenario applicativo è la market basket analysis (MBA) e la regola più celebre è quella che relaziona la vendita di birre a quella di noccioline, ed es.: "nel 30% dei casi la birra è venduta insieme alle noccioline".
Gli algoritmi che producono le associazioni sono abbastanza veloci e ne producono in gran quantità, la sfida è scegliere le più significative a livello empirico ed escludere quelle di natura puramente probabilistica. A tale scopo viene in soccorso la segmentazione. In grandi insiemi di dati le dipendenze sono raramente significative, in quanto numerose influenze si sovrappongono generando relazioni casuali. In tali casi è consigliabile effettuare l'analisi delle dipendenze su segmenti di dati omogenei.
Come già accennato vi è una collaborazione con la predizione (e la classificazione), infatti le dipendenze vengono spesso usate per formulare ipotesi predittive o di classificazione e valutare se e come esse trovano riscontro nei dati reali.
Un altro caso particolare di dipendenze sono i pattern sequenziali, dove invece degli oggetti (item-based)sono considerati gli eventi (event-based). Nella MBA, le associazioni descrivono le relazioni fra oggetti in un istante temporale, mentre i pattern sequenziali esaminano le tendenze di acquisto dei consumatori, o gruppi di consumatori nel tempo.

Tecniche appropriate:

- Analisi di correlazione
- Analisi di regressione
- Regole di associazione
- Reti Bayesiane
- Programmazione logica induttiva
- Tecniche di visualizzazione

Effettuando, ad esempio, una analisi di regressione, l'analista più trovare una dipendenza fra le vendite totali di un articolo, il prezzo e le spese pubblicitarie promozionali. Avendo a disposizione una relazione matematica che correla le tre variabili, l'analista può raggiungere un livello di vendita variandole opportunamente.
Oppure, applicando uno specifico algoritmo per le regole di associazione, una casa automobilistica può notare che se è ordinata una radio, nel 95% dei casi è desiderato il cambio

automatico. In base a ciò la casa può proporre entrambi gli accessori in un unico pacchetto optional, il che porta ad una riduzione dei costi.

3.3 Struttura comune delle tecniche

Si è gia detto che le tecniche di modellazione provengono da vari campi di ricerca, come la statistica, l'intelligenza artificiale ecc. Questo fatto, insieme al grande numero di algoritmi disponibili per i problemi di mining, potrebbe generare confusione anche ad un analista esperto. Per diminuire, almeno in parte, questa confusione è utile sottolineare il fatto che le varie tecniche hanno una struttura generale comune. Essa può essere descritta da 3 componenti principali:

- **RAPPRESENTAZIONE DEL MODELLO:**

  E' la forma funzionale dei modelli prodotti dall'algoritmo. Formalmente può essere espressa come una funzione del tipo $y = f(x, P)$. x rappresenta l'input, ovvero coppie attributo-valore definite da $x = (a_i, v)$, con $i = 1...N$ dove N è il numero totale degli attributi usati nel modello. P è un set di parametri che descrivono il modello e sono specifici per ogni modello.

  Ad esempio, nel caso degli algoritmi per alberi decisionali *y* rappresenta un insieme di nodi ed archi, mentre per le regole di associazione *y* è un insieme di regole espresse a limite anche in linguaggio naturale. Questioni importanti che riguardano la rappresentazione sono il tipo di dati in ingresso: continui, discreti, interi, categorici o tutti insieme; il potere esplicativo del modello; le caratteristiche della funzione di approssimazione (lineare, non lineare…); la forma dell'output del modello.

- **CRITERIO DI STIMA:**

  Quando è data una particolare rappresentazione *f*, il criterio valuta quanto bene un particolare set di parametri *P* si adatta ai dati. Questi criterio di stima è interno alla particolare tecnica e non va confuso con i criteri di accertamento del modello per un modello già costruito (cfr. par. 2.4.4). Il criterio di stima valuta la costruzione di istanze differenti di f, durante la ricerca nello spazio delle istanze usa il metodo di ricerca, che è la terza componente basilare della struttura delle tecniche. Caratteristiche tipiche del criterio di stima sono: sensitività e robustezza di un criterio di stima per un modello particolare come funzione della dimensione del campione dati e della dimensionalità del problema; le assunzioni alla base del criterio di tipo logico o probabilistico. Il criterio differisce sensibilmente da tecnica a tecnica ed è una conseguenza del particolare modello e dal metodo di ricerca adottato.

- **METODO DI RICERCA:**

  Data una forma di rappresentazione ed un criterio di stima, il metodo governa la ricerca nello spazio di tutte le possibili istanze di rappresentazione quella che ottimizza (minimizza o massimizza) il criterio. Precisamente questo significa che, data una rappresentazione ed un criterio, le tecniche di modellazione lavorano come algoritmi di ottimizzazione ed hanno le loro tipiche caratteristiche: metodologia di ricerca (algoritmi avidi, euristici, esaustivi), complessità di ricerca, controllo della ricerca (vincoli di tempo e/o memoria, criteri di stop).

3.4 Analisi dei cluster

Si è gia accennato al fatto che il clustering si riferisce ad un tipo di apprendimento non supervisionato, quindi esso apprende dall'osservazione piuttosto che dagli esempi (training example). Il clustering rappresenta un ampio campo di ricerca, i cui sforzi sono mirati a soddisfare e migliorare alcuni dei requisiti di base per la sua applicabilità al DM. I maggiori requisiti sono:

1. *Scalabilità:* molti algoritmi di clustering lavorano bene su piccoli insiemi di dati contenenti poche centinaia di oggetti; d'altro canto i grandi DB aziendali contengono record nell'ordine dei milioni. Il clustering effettuato su campioni di dati può condurre a risultati polarizzati.
2. *Gestione di tipi di attributi differenti:* molti algoritmi di clustering sono progettati per operare su tipi numerici; molti problemi, però, sono basati su tipi di dati categorici (nominali), binari, ordinali o misti.
3. *Cluster di forma arbitraria:* molti algoritmi costruiscono raggruppamenti sulla base della distanza Euclidea o di Manhattan, quindi essi risultano spesso di forma simil-circolare o ellittica, aventi dimensione e densità similari. È importante sviluppare tecniche che diano cluster di qualsiasi forma ove siano opportuni.
4. *Sensibilità ai parametri:* i cluster risultanti sono spesso molto sensibili ai parametri di input definiti dall'utente (es. numero di cluster). Molti parametri sono difficili da impostare in special modo quando si ha a che fare con oggetti ad alta dimensionalità e non si ha una sufficiente conoscenza del dominio applicativo.
5. *Trattamento del rumore:* i dati provenienti dal mondo reale sono affetti da errori, dati mancanti, dati sconosciuti, deviazioni (outliers). Alcuni algoritmi sono troppo sensibili a questi fattori e possono produrre cluster di scarsa qualità.

6. *Indipendenza dall'ordinamento dei record:* alcuni algoritmi producono cluster molto differenti sullo stesso insieme di dati ordinato in maniera differente. È necessario sviluppare algoritmi non afflitti da questo problema.
7. *Alta dimensionalità:* gli oggetti spesso contengo un gran numero di dimensioni o attributi. La maggior parte degli algoritmi si comporta bene su 2-3 dimensioni, nel qual caso può intervenire il giudizio umano per giudicare i risultati. Il clustering diventa arduo quando ci si trova in uno spazio *n*-dimensionale (con *n* nell'ordine delle decine o centinaia), che tipicamente è sparso e con molti valori deviati.
8. *Vincoli reali:* nelle applicazioni reali può risultare difficile determinare i vincoli ai quali deve sottostare la tecnica scelta e formare insiemi di dati accettabilmente omogenei sui quali operare.
9. *Interpretabilità e usabilità:* l'utente si aspetta che i risultati ottenuti siano facilmente interpretabili. Il compito è trovare il legame tra interpretazioni semantiche e applicazioni tecniche, e valutare come gli obiettivi dell'applicazione possono influenzare la scelta della tecnica di clustering.

3.4.1 Tipi di dati

In questa sezione vedremo quali sono i tipi di dati più frequenti nell'analisi dei cluster. Supponiamo di avere *n* oggetti che possono rappresentare qualsiasi tipo di entità: persone, case, documenti ecc.

La maggior parte degli algoritmi operano sulle seguenti strutture:

1. **Matrice dei dati:**

$$\begin{bmatrix} x_{1,1} & \dots & x_{1,f} & \dots & x_{1,p} \\ \dots & \dots & \dots & \dots & \dots \\ x_{i,1} & \dots & x_{i,f} & \dots & x_{i,p} \\ \dots & \dots & \dots & \dots & \dots \\ x_{n,1} & \dots & x_{n,f} & \dots & x_{n,p} \end{bmatrix}$$

   che è una matrice *(n x p)* con *n* oggetti (record) e *p* variabili (attributi).

2. **Matrice delle dissimilarità:**

$$\begin{bmatrix} 0 & & & & \\ d(1,2) & 0 & & & \\ d(1,3) & \dots & 0 & & \\ \dots & \dots & \dots & 0 & \\ d(n,1) & \dots & \dots & \dots & 0 \end{bmatrix}$$

matrice *(n x n)*, dove ogni elemento rappresenta la *dissimilarità* o differenza fra l'elemento i e l'elemento j. Essa è simmetrica con diagonale nulla, $d(i,j) = d(j,i)$, $d(i,i) = 0$.

I cluster sono determinati sulla base della dissimilarità o similarità fra gli oggetti. In generale la dissimilarità $d(i,j)$ è una quantità non negativa che misura la "vicinanza" dell'oggetto i all'oggetto j, essa tende a zero quanto più essi sono vicini.

Alcune misure di dissimilarità possono essere desunte da semplici votazioni effettuate da esperti di dominio. Alternativamente essa può essere calcola tramite i coefficienti di dissimilarità. Dati *n* oggetti, la misura dell'affinità tra due possibili variabili è data dalla formula della correlazione di Pearson:

$$R(f,g) = \frac{\sum_{i=1}^{n}(x_{i,f} - m_f)(x_{i,g} - m_g)}{\sqrt{\sum_{i=1}^{n}(x_{i,f} - m_f)^2}\sqrt{\sum_{i=1}^{n}(x_{i,g} - m_g)^2}}$$

I coefficienti possono essere calcolati in due modi:

$d(f,g) = (1 - R(f,g))/2$; $d(f,g) = 1 - |R(f,g)|$

analogamente è possibile utilizzare un coefficiente di *similarità*:

$s(f,g) = 1 - d(f,g)$

tali coefficienti danno una misura dell'importanza relativa di una coppia di variabili, dato un insieme di oggetti.

VARIABILI NUMERICHE

La dissimilarità fra gli una coppia di oggetti è basata sul concetto di ***distanza***. Date delle variabili numeriche, vi sono alcuni approcci per definire una distanza. La più nota è la distanza *Euclidea* definita come $d(i,j) = \sqrt{|x_{i,1} - x_{j,1}|^2 + |x_{i,2} - x_{j,2}|^2 + ... + |x_{i,p} - x_{j,p}|^2}$, dove *i* e *j* sono due oggetti *p*-dimensionali. Un'altra metrica conosciuta è la distanza di *Manhattan*, spesso usata per clustering su territori urbani, infatti è detta anche *city block distance*, ed è definita come: $d(i,j) = |x_{i,1} - x_{j,1}| + ... + |x_{i,p} - x_{j,p}|$

Entrambe le distanze soddisfano le seguenti proprietà:

1. $d(i,j) \geq 0$
2. $d(i,i) = 0$
3. $d(i,j) = d(j,i)$

4. $d(i, j) \le d(i, h) + d(h, j)$ (disuguaglianza triangolare)

La distanza di *Minkowsky* è una generalizzazione delle precedenti, infatti è definita come:

$$d(i, j) = \left(\left| x_{i,1} - x_{j,1} \right|^q + ... + \left| x_{i,p} - x_{j,p} \right|^q \right)^{1/q}$$

Se le variabili in gioco hanno una importanza relativa differente, dettata dall'applicazione, si possono aggiungere degli opportuni coefficienti di peso, in tal caso la distanza Euclidea sarà:

$$d(i, j) = \sqrt{w_1 \left| x_{i,1} - x_{j,1} \right|^2 + ... + w_p \left| x_{i,p} - x_{j,p} \right|^2}$$

VARIABILI BINARIE

Una variabile binaria ha solo due stati: 0 e 1. Non è possibile trattare le variabili binarie come numeriche in quanto ciò porterebbe a risultati di clustering errati. La dissimilarità va misurata in altro modo.

Un approccio consiste nel calcolare la *matrice di dissimilarità* tramite le tabelle di contingenza. Consideriamo due oggetti, avremo una tabella 2 x 2 del tipo:

i \ j	1	0	Sum
1	a	b	a+b
0	c	d	c+d
Sum	a+c	b+d	P

dove a è il numero di variabili binarie con valore 1 per entrambi gli oggetti; b il numero di variabili di valore 1 per l'oggetto i e 0 per l'oggetto j, e cosi via.

Una variabile binaria è detta simmetrica se entrambi gli stati hanno lo stesso peso, cioè non vi è alcuna preferenza se il valore debba essere codificato come 1 o come 0. La similarità applicata a variabili binarie simmetriche è detta similarità invariante, in questo caso il coefficiente più adoperato è il *simple matching coefficient*, definito come:

$$d(i, j) = \frac{a + b}{a + b + c + d}.$$

Una variabile binaria è detta simmetrica se il presentarsi di uno dei sue stati non sono ugualmente importanti; ad esempio in un test di sieropositività, l'occorrenza di due '1' è più significativa dell'occorrenza di due '0'. In questo caso si utilizza il *coefficiente di Jaccard*:

$$d(i, j) = \frac{a + b}{a + b + c}$$

dove il numero di concordanze negative d è ritenuto trascurabile ed escluso dalla computazione.

VARIABILI NOMINALI

Le variabili nominali sono una estensione delle variabili binarie, nel senso che esse possono avere un numero arbitrario *M* di stati. Ad esempio una variabile `colore_occhi` può avere 4 stati: {nero, castano, verde, azzurro}. Gli stati possono essere codificati in varie meniere: lettere, simboli, interi (in tal caso gli interi non hanno nessun ordinamento). La dissimilarità può essere calcolata con l'approccio simple match: $d(i,j) = \frac{p-m}{p}$, dove *m* è il numero di corrispondenze (matches) fra variabili omologhe di oggetti diversi aventi lo stesso stato; e *p* è il numero di variabili totale.

Si possono codificare le variabili nominali anche con un set di variabili binarie pari a M, ogni variabile x_i, con $i = 1...M$, assume il valore 1 se la variabile nominale si trova nello stato $i-esimo$, 0 in tutti i rimanenti. Fatto questo si procede a calcolare i coefficienti di dissimilarità come per le variabili binarie.

VARIABILI ORDINALI

Una variabile ordinale *discreta* è simile alle variabili nominali eccetto per il fatto che gli M stati hanno un ordinamento significativo. Esse sono usate per rappresentare qualità soggettive che sono misurabili oggettivamente e posizionate in una classificazione dipendente dalle loro importanza assoluta o relativa. Ad esempio, una variabile `posizione_professionale` può assumere gli stati: {impiegato, quadro, subalterno, dirigente}. Un set di variabili ordinali *continue*, misurate in una qualsiasi scala, rappresentano un insieme di dati continui dove quello che è significativo è la distanza reciproca fra esse e non il valore assoluto. Si pensi ad una variabile `tempo_sul_giro`, misurata in secondi, dove è fondamentale rappresentare il gap fra l'arrivo al traguardo delle autovetture. Da ciò si intende che esse possono essere ottenute dalle variabili numeriche discretizzandole e raggruppandole in varie classi predefinite; e da questa analogia esse possono essere trattate come le numeriche per il calcolo della dissimilarità. Sia *f* una variabile ordinale descrivente n oggetti e sia M_f il numero di stati possibili aventi una ben precisa sequenza. Se $x_{i,f}$ è il valore della variabile in questione dell'oggetto *i*, la si sostituisce con il valore ordinale dello stato in cui si trova ovvero $r_{i,f} \in \{1...M_f\}$, dopodiché la si normalizza nell'intervallo $[0,1]$ calcolando lo z-score nel modo seguente $z_{i,f} = \frac{r_{i,f}-1}{M_{i,f}-1}$. A questo punto è possibile calcolare la distanza secondo gli approcci descritti per le variabili numeriche.

VARIABILI NON LINEARI

Hanno valori positivi misurati su scale non lineari, come ad esempio scale esponenziali descritte approssimativamente dalle formule: $Ae^{Bt}; Ae^{-Bt}$, con A e B costanti positive. Esse possono essere trattate come variabili numeriche lineari o ordinali continue, ma non è la soluzione migliore in quanto la scala risulterà distorta. L'approccio migliore è applicare preventivamente una trasformazione logaritmica del tipo $y_{i,f} = \log(x_{i,f})$ per renderle lineari. Le trasformazioni opportune sono ovviamente dettate dal tipo di non-linearità.

VARIABILI MISTE

Nella maggior parte dei magazzini dati si ha a che fare con variabili miste appartenenti alle categorie sopra citate. Un approccio banale può essere quello di raggruppare le variabili dello stesso tipo ed effettuare il clustering su ogni gruppo distintamente. Tale metodo, però, porta a risultati che tipicamente non sono accettabili o che non hanno rilevanza pratica. Una via più sistematica è quella proposta da (Ducker et al. 1965) che racchiude le dissimilarità delle variabili in un'unica matrice ed effettua la scalatura su di un intervallo comune $[0,1]$.

Supponiano si avere *p* variabili miste, la dissimilarità è così computata: $d(i,j) = \dfrac{\sum_{f=1}^{p} \delta_{i,j}^{(f)} d_{i,j}^{(f)}}{\sum_{f=1}^{p} d_{i,j}^{(f)}}$,

dove $\delta_{i,j}^{(f)} = 0$ se mancano le misure $x_{i,f}$ o $x_{j,f}$ oppure se sono variabili binarie asimmetriche e risulta $x_{i,f} = x_{j,f} = 0$, altrimenti $\delta_{i,j}^{(f)} = 1$. Il contributo alla dissimilarità fra i e j è calcolato a seconda del tipo della variabile $f - esima$:

1. Binaria o nominale: $d_{i,j}^{(f)} = 0$ se $x_{i,f} = x_{j,f}$, 1 altrimenti.
2. Numerica: $d_{i,j}^{(f)} = \dfrac{\left| x_{i,f} - x_{j,f} \right|}{\max_h x_{h,f} - \min_h x_{h,f}}$, con *h* indice delle sole variabili aventi valore.
3. Ordinali o non lineari: calcolare $r_{i,j}$ e poi $z_{i,f} = \dfrac{r_{i,f} - 1}{M_{i,f} - 1}$ e trattare lo z-score come variabile numerica.

3.4.2 Metodi di partizionamento

Dato un insieme di *n* oggetti o tuple, un metodo di partizionamento costruisce *k* partizioni ognuna delle quali rappresenta un cluster, essendo $k \le n$. I raggruppamenti devono soddisfare le due proprietà seguenti: (1) ogni cluster deve contenere almeno un oggetto, (2) ogni oggetto deve appartenere ad un solo cluster (alcune tecniche, dette fuzzy, non rispettano la seconda

proprietà). La logica alla base è di racchiudere in ogni cluster oggetti simili tra loro e far si che oggetti in gruppi diversi siano il più possibile "lontani" tra loro. I cluster sono costruiti per ottimizzare una funzione di similarità, come la distanza.

Algoritmo delle k-medie

Questo algoritmo prende in input il numero k di cluster desiderati e misura la similarità degli oggetti rispetto al valor medio di ogni gruppo.

Esso procede in questo modo: inizialmente sceglie casualmente k oggetti che formano i cluster iniziali; ognuno dei rimanenti oggetti è assegnato al cluster avente la media più vicina (simile); infine la media del cluster viene aggiornata. Il processo iterativo continua finchè la funzione di similarità converge. Tipicamente viene usato l'errore quadratico a minimizzare, definito da: $E = \sum_{i=1}^{k} \sum_{x \in C_i} |x - m_i|^2$ dove m_i è la media del cluster C_i (si noti che sia x che m_i sono multidimensionali). Riportiamo la procedura in pseudo-codice:

```
//ALGORITMO K_MEDIE
INPUT (k,n)
OUTPUT (k clusters)
      scegli k oggetti iniziali;
      REPEAT
            assegna oggetto i al cluster più simile;
            aggiorna la media del cluster;
      UNTIL nessun cambio;
```

in fig. 29 sono rappresentati a titolo esemplificativo tre tipi di raggruppamento all'evolvere dell'algoritmo: nel primo si nota la scelta di un oggetto come centroide del gruppo, negli altri si notano gli spostamenti delle medie (denotate con +) e i passaggi di alcuni oggetti da un gruppo ad un altro che determinano i cambiamenti di forma.

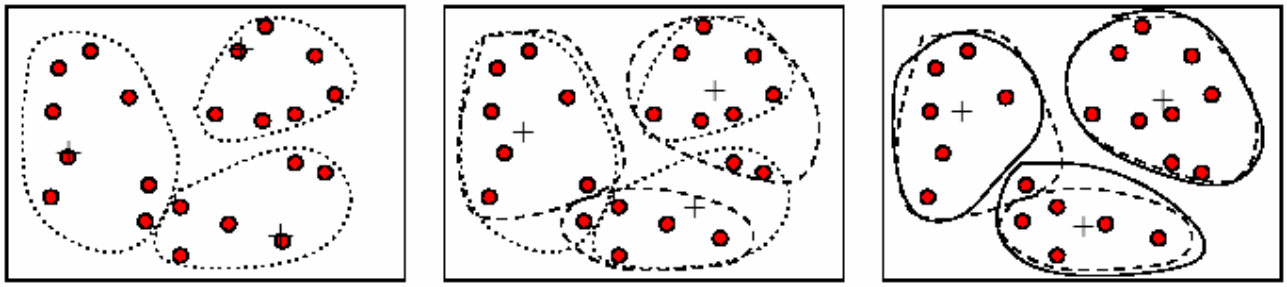


Figura 29: Composizione dei cluster all'evolvere dell'algoritmo delle k-medie.

la complessità dell'algoritmo è $O(nkt)$, con t = numero di iterazioni. Nella maggior parte dei casi si ha una convergenza verso un ottimo locale. Tale algoritmo ovviamente è applicabile solo quando la media è definibile, il che non è possibile ad esempio se si trattano attributi categorici. Inoltre in numero k di cluster può essere difficile da determinare da parte dell'utente (allo scopo è utile applicare preliminarmente sul set di dati un algoritmo

agglomerativo gerarchico per avere un valore *k* plausibile). Infine l'algoritmo è sensibile agli outliers, che possono deviare gravemente il valor medio dei cluster.
Una variante è l'algoritmo EM[2] (Expectation-Maximization) (Lauritzen 1995) che estende il paradigma delle k-medie: invece di assegnare una oggetto ad un unico cluster esso lo assegna a vari cluster secondo dei pesi che ne rappresentano la probabilità di appartenenza.

Algoritmo dei k-medioidi

Questo algoritmo, invece di prendere come valore di riferimento il valor medio del cluster, prende uno specifico oggetto posizionato centralmente. In questo modo gli effetti degli outliers sono attenuati. PAM (Partitioning Around Medoid) è un algoritmo di questo tipo. Dopo la scelta iniziale di k oggetti rappresentativi (medioidi), esso raggruppa secondo le dissimilarità e poi valuta ogni possibile coppia di oggetti per trovare medioidi più rappresentativi.

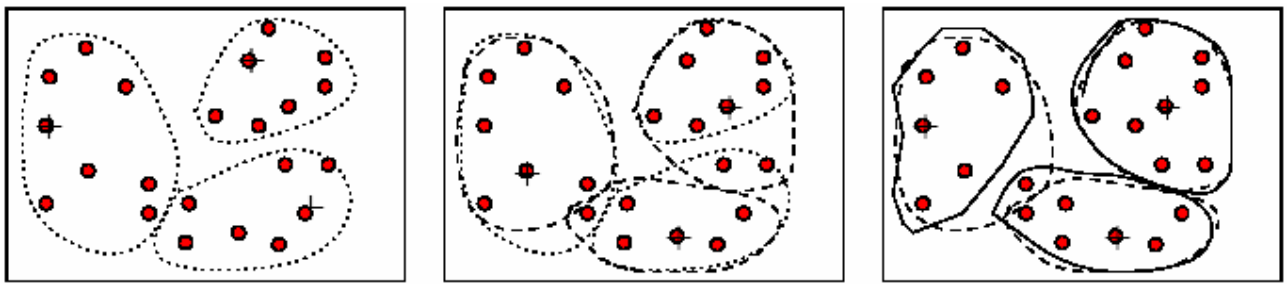


Figura 30: Cmposizione dei cluster e scelta dei medioidi nell'algoritmo PAM.

Dalla fig. 30 si vede che il valore di riferimento (+) è rappresentato da un oggetto specifico.
La procedura in psudo-codice è riportata di seguito:

```
//ALGORITMO PAM
INPUT (k,n)
OUTPUT (k cluster)
      Scegli k medioidi iniziali;
      REPEAT
            Assegna oggetti ai cluster;
            Calcola funzione obiettivo (somma delle dissimilarità di tutti
            gli oggetti rispetto al medioide);
            Scambia medioide x con oggetto y se ciò produce una riduzione
            della funzione obiettivo;
      UNTIL nessun cambio
```

Il costo di una singola iterazione è $O(k(n-k)^2)$. PAM è più robusto rispetto agli outliers rispetto alle k-medie, ma il costo di processamento è maggiore; inoltre anche esso ha bisogno del parametro k in input.

Algoritmo CLARA

[2] Una implementazione di questa versione è presente negli strumenti di Analysis Services in Microsoft SQL Server 2000 – 2005.

PAM lavora bene con insiemi di dati medio-piccoli, ma non scala bene su grandi DB. Per questo ultimo caso è stato sviluppato un metodo basato su campionamento (*sampling-based*) che sfrutta PAM: CLARA (Kaufman-Rousseeuw, 1990).
L'idea alla base di CLARA è la seguente: invece dell'intero insieme dati vengo scelti dei sottoinsiemi campione sui quali applicare PAM; se i campioni sono scelti casualmente essi rappresenteranno abbastanza bene l'intero insieme. CLARA dà in output il clutering migliore su tutti i campioni. La complessità di una iterazione diventerà $O(kS^2 + k(n-k))$, dove S è la dimensione del campione.
Si noti che PAM cerca i migliori k medioidi in un singolo insieme, mentre CLARA cerca i migliori k medioidi negli insiemi selezionati; se i migliori k medioidi dell'insieme di partenza non sono selezionati per qualche motivo, CLARA non troverà mai il clustering migliore. Insomma un buon metodo di clustering basato su campionamento non produrrà un clustering altrettanto buono se il campionamento è polarizzato.

Algoritmo CLARANS

Un miglioramento di CLARA è CLARANS stato proposto da (Ng – Han, 1994), la cui metodologia è basata sugli algoritmi di ottimizzazione. CLARA lavora sempre sullo stesso campione dati, CLARANS estrae casualmente un campione ad ogni passo della ricerca del clustering migliore. Questo passo di ricerca può essere descritto con un grafo di ricerca, dove ogni nodo rappresenta una soluzione, ovvero un insieme di k medioidi; un nodo unito ad un altro da un arco è chiamato "vicino" (neighbour) e differisce da esso per un solo medioide. Il numero di vicini da esaminare può essere limitato da un parametro. Costruendo questo grafo si perviene ad un ottimo locale, quando esso è trovato CLARANS sceglie un nuovo campione dati dall'insieme ed il processo è nuovamente avviato.
Nelle sperimentazioni CLARANS si è dimostrato più efficace sia di PAM che di CLARA; la sua complessità cresce linearmente con il numero di oggetti. Da rimarcare è il fatto che il presente algoritmo non necessità del settaggio del paramentro k, in quanto esso lo determina tramite un *coefficiente di silhouette*, che misura quanto un oggetto è legato al cluster di appartenenza. Inoltre CLARANS permette, come i suoi predecessori, l'individuazione di outliers.

3.4.3 Metodi gerarchici

I metodi gerarchici creano una decomposizione gerarchica dell'insieme di dati. Essi possono essere classificati in due categorie:

1. **Agglomerativi:** si fondano sulla metodologia bottom-up. Inizialmente ogni oggetto dell'insieme forma un cluster differente, poi i cluster vengono fusi (merged) sulla valutazione di alcune distanze fino alla soddisfazione di una condizione di arresto, o, banalmente, alla formazione di un unico grande cluster.
2. **Divisivi:** procedono al contrario, ovvero secondo l'appproccio top-down. Vi è un unico cluster iniziale che viene decomposto (splitted) lungo un albero di cluster fino al raggiungimento di una condizione di arresto, come il numero di cluster desiderati o il raggiungimento di un limite inferiore per la distanza dei due cluster più vicini.

Questi tipo di algoritmi soffrono della irreversibilità delle scelte effettuate (fusione o decomposizione), quindi se una operazione non è effettuata oculatamente non vi sarà possibilità di correggere l'errore ed il risultato non sarà ottimale.

Le misure più usate per la distanza fra i cluster sono le seguenti:

- $d_{\min}(C_i, C_j) = \min_{p \in C_i, p \in C_j} |p - p'|$
- $d_{mean}(C_i, C_j) = |m_i - m_j|$
- $d_{avg}(C_i, C_j) = \frac{1}{n_i n_j} \sum_{p \in C_i} \sum_{p' \in C_j} |p - p'|$
- $d_{\max}(C_i, C_j) = \max_{p \in C_i, p \in C_j} |p - p'|$

con le solite notazioni ed indicando con $|p - p'|$ la distanza fra due punti arbitrari.

AGNES (AGglomerative NESting) è un algoritmo agglomerativo che usa la distanza Euclidea minima fra due cluster ($d_{\min}$). DIANA (*DIvisia ANAlysis*) è di tipo divisivo e per effettuare le decomposizioni si basa sulla distanza Euclidea massima ($d_{\max}$) applicata agli oggetti più vicini nel cluster.

Per superare i problemi sopra menzionati, una direzione promettente è quella di utilizzare algoritmi gerarchici in combinazione con altre tecniche come algoritmi rilocativi (con scambi di oggetti da un cluster ad un altro), ottenendo cosi delle procedure multi-fase.

Algoritmo BIRCH

Un algoritmo di tipo gerarchico intergrato è il BIRCH (*Balancing Iterative Reducing and Clusterign using Hierarchies*), proposto da (Zhang, Ramakrishnan e Livny, 1996). Esso introduce due concetti, il primo è la *Clustering Feature* (CF), che è una tripla che riassume i dati statistici di sottoinsiemi di punti, così definita: $CF = \left(N, \vec{LS}, SS \right)$ dove N è il numero di

punti, $\vec{LS} = \sum_{i=1}^{N} \vec{X}_i$ (somma lineare o momento del 1° ordine) e $SS = \sum_{i=1}^{N} \vec{X}_i^{\,2}$ (somma quadratica o momento del 2° ordine). L'altro elemento è il *CF-tree*, ovvero un albero bilanciato che memorizza le CF. Esso ha due parametri B (Branching factor) e T (Threshold). B specifica il massimo numero di figli che un nodo può avere e T indica il diametro massimo dei sottocluster memorizzati dai nodi foglia. Ogni nodo non foglia mantiene le CF relative ad ogni suo figlio.

L'algoritmo si articola un due fasi:

- Fase 1: scandisce l'intero DB e costruisce un CF-tree iniziale in memoria centrale.
- Fase 2: applica uno specifico algoritmo di clustering ai nodi foglia.

La costruzione dell'albero è dinamica durante l'inserimento dei punti, in tal senso il metodo è incrementale. Ogni punto viene inserito nel nodo foglia più vicino; se dopo l'inserimento viene superata la soglia T, il cluster in questione, o anche altri, possono essere scissi per formarne altri. L'informazione detenuta dal punto inserito, modifica la CF del nodo di appartenenza e raggiunge la radice dell'albero. Se l'albero costruito è troppo complesso per essere gestito in memoria centrale si può scegliere un valore di T più alto e ricostruire l'albero. Il processo di ricostruzione è fatto a partire dai nodi foglia del vecchio albero per evitare una nuova scansione dei dati. L'albero è quindi prodotto con una singola lettura dei dati; alle volte sono integrati alcuni metodi di gestione degli outlier che necessitano di scansioni addizionali.

Effettuata la costruzione dell'albero, nella seconda fase può essere adoperato un qualsiasi algoritmo di clustering, tipicamente vengono usati metodi di partizionamento.

BIRCH tenta di trovare il clustering migliore con le risorse a disposizione; usa un quantità limitata di memoria centrale; riduce al minimo le operazioni di I/O e la sua complessità è $O(N)$. D'altronde siccome i sottoinsiemi hanno diametro limitato, talvolta la struttura ottenuta può non corrispondere al clustering "naturale"; inoltre se i cluster non hanno forma sferica, la procedura non si adatta bene perchè fa uso delle nozioni di raggio e diametro per controllare i confini dei cluster.

Algoritmo CURE

Gli algoritmi visti finora favoriscono la formazione di cluster sferici e sono mediamente sensibili agli outliers. Un interessante metodo che cerca di ammortizzare queste predisposizioni è l'algoritmo CURE (*Clustering Using REpresentatives*) sviluppato da (Guha, Rastogi e Shim, 1998). Esso combina tecniche gerarchiche e algoritmi di partizionamento.

Invece di usare un singolo centroide, sono presi vari punti rappresentativi ben sparsi nel cluster, poi questi sono traslati verso il centro di gravità del cluster di una frazione α (fattore di shrinking) così da catturarne la forma reale. Il ritirarsi del cluster attorno al suo centro di gravità aiuta anche ad individuare eventuali valori deviati.

I passi più importanti della procedura sono i seguenti: (1) si sceglie casualmente un campione *s*; (2) si divide il campione in *p* partizioni, ognuna di dimensione *s / p*; (3) si formano *s / pq* sottocluster, con $q > 1$ opportuno, basati sulla distanza media minima; (4) si eliminano gli outliers: se i sottocluster crescono troppo lentamente, si elimina l'outlier; (5) si sposta il punto rappresentativo di ogni sottocluster (il valor medio) verso il centro di gravità secondo il parametro *α*, definito dall'utente; (6) i sottocluster con i punti rappresentativi più vicini sono fusi insieme fino a raggiungere *p* cluster.

Nelle figg. 31-32 possiamo vedere l'evoluzione dell'algoritmo, con i seguenti parametri: $s = 50$, $p = 2$, $q = 1.75$, così da formare 14 sottocluster.

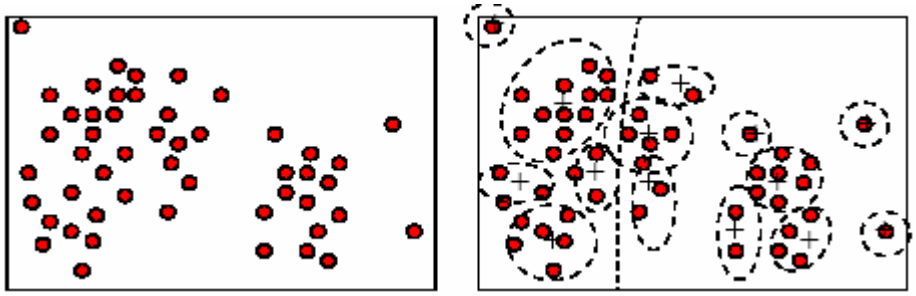


Figura 31: 1° fase dell'algoritmo CURE: partizionamento e formazione dei sottocluster.

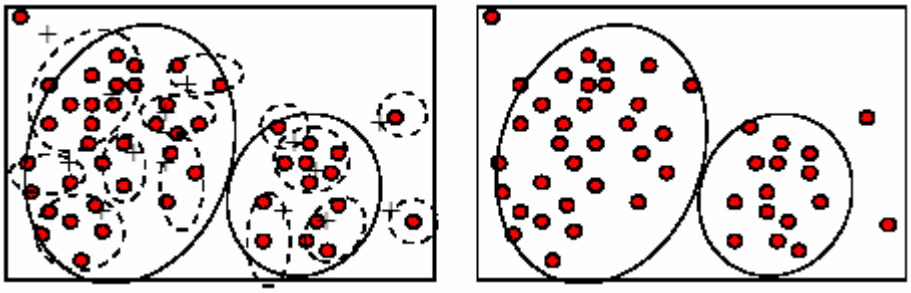


Figura 32: 2° fase dell'algoritmo CURE: eliminazione delle deviazioni e raggruppamento dei cluster.

CURE produce cluster di alta qualità, di forme arbitrarie ed in presenza di forti deviazioni. Esso necessita di alcuni parametri in input: *s*, *p* ed *α* devono essere definiti dall'utente. La struttura risultante dipende sensibilmente da questa scelta, quindi è consigliabile un'analisi della qualità alle variazioni dei parametri.

3.4.4 Metodi basati sulla densità

I metodi di partizionamento costruiscono strutture aventi cluster approssimativamente sferici in quanto fanno uso della nozione di distanza fra 2 oggetti. Per ovviare a questo limite sono stati sviluppati dei metodi basati sul concetto di densità. L'idea generale è: una volta trovato un cluster, si continua ad accrescerlo aggiungendovi punti se la densità nelle "vicinanze" (neighbourhood) supera una certa soglia. È intuitivo il fatto che così facendo si possono scovare cluster di qualsivoglia contorno, filtrando allo stesso tempo gli outliers.

Algoritmo DBSCAN

DBSCAN (*Density Based Spatial Clustering of Application with Noise*) è un algoritmo basato sulla densità, sviluppato da (Ester, Kriegel, Sander e Xu, 1996). Esso definisce un cluster come un insieme di punti *connessi in densità* (*density - connected*). Vediamo le definizioni dei concetti di cui fa uso la tecnica.

Un oggetto è detto nucleo (core–object) se all'interno della sua ε - vicinanza (ε - neighbourhood), definita da un raggio (ε), si trovano un numero sufficiente di punti; tale quantità è definita dal parametro (*MinPts*).

Un oggetto p è ***direttamente raggiungibile in densità*** dall'oggetto q rispetto al raggio ε ed a un numero minimo di punti *MinPts* in un insieme D se p si trova all'interno della ε – vicinanza di q che contiene almeno *MinPts* punti.

Un oggetto p è ***raggiungibile in densità*** dall'oggetto q rispetto al raggio ε ed a un numero minimo di punti *MinPts* in un insieme D se esiste una successione di punti $p_1, p_2, ..., p_n$ con $1 \le i \le n$, $p_1 = q, p_n = p, p_i \in D$ tali che p_{i+1} è raggiungibile in densità da p_i rispetto a ε e MinPts.

Un oggetto p è ***connesso in densità*** ad un oggetto q rispetto a ε e MinPts in D se $\exists$ un oggetto $o \in D$ tale che p e q sono raggiungibili in densità da o rispetto a ε e MinPts.

Si noti che la raggiungibilità è la chiusura transitiva della raggiungibilità diretta ed entrambe sono relazioni asimmetriche, solo i nuclei sono mutuamente raggiungibili in densità. La connessione, invece, è una relazione simmetrica. Per evidenziare tali proprietà si faccia riferimento alla fig. 33.

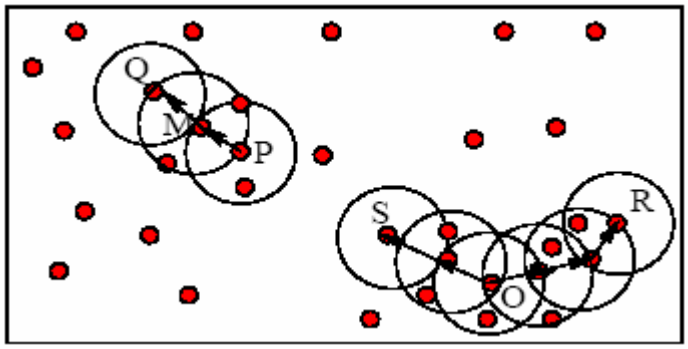


Figura 33: Concetti di raggiungibilità e connessione in densità.

Assumiamo MinPts = 3 e notiamo che M è direttamente raggiungibile in densità da P; Q è raggiungibile in densità da P; peraltro P non è raggiungibile da Q; S e R sono raggiungibili da O, se ne deduce che S e R sono connessi in densità.

Un ***cluster basato su densità*** è un insieme di oggetti connessi in densità che è massimo rispetto alla raggiungibilità; ogni elemento estraneo al cluster è rumore.

L'algoritmo verifica la ε – vicinanza di ogni punto dell'insieme dati, se viene raggiunto un numero MinPts di punti appartenenti alla ε – vicinanza di un punto p, è creato un nuovo cluster con nucleo p. Poi procede all'aggiunzione dei punti direttamente raggiungibili dal cluster, ciò può portare anche alla fusione di regioni raggiungibili. La procedura termina quando non è più possibile aggiungere punti ai cluster trovati.

Algoritmo OPTIC

L'algoritmo DBSCAN necessita di alcuni parametri definiti dall'utente, ε e MinPts. In alcuni casi risulta molto difficile una buona calibrazione dei parametri ed inoltre le strutture rislultanti possono differire sensibilmente anche per lievi scostamenti. Per superare queste difficoltà si può utilizzare l'algoritmo OPTIC (*Ordering Points To Identify the Clustering structure*) che costruisce un ordinamento delle strutture di clustering che corrispondono ad un ampio campo di settaggio dei parametri. OPTOC fu proposto da (Ankerst, Breunig, Kriegel e Sander, 1999).

Osservando il funzionamento di DBSCAN possiamo notare che, fissato MinPts, i cluster a più alta densità (con ε minore) sono completamente contenuti nei cluster a densità inferiore. A questo punto, per produrre cluster basati su densità appartenenti ad un range di parametri, si ha bisogno di conoscere l'ordine con il quale prendere in esame i vari oggetti in modo che oggetti raggiungibili in densità con il valore minore di ε sono processati prima. Avendo questa idea alla base, la tecnica introduce due nuovi concetti: la ***distanza dal nucleo*** (*core - distance*) di un oggetto p è la più piccola distanza ε' tra p ed un oggetto nella sua ε – vicinanza tale che

p sia nucleo rispetto a ε'; altrimenti tale distanza è non definita. La ***distanza di raggiungibilità*** (*reachability - distance*) di un oggetto p rispetto a dun altro oggetto o è la più piccola distanza tale che p è direttamente raggiungibile in densità da o se o è un nucleo. Se o non è nucleo, nemmeno rispetto alla distanza generatrice ε, tale quantità rimane indefinita.
L'algoritmo crea un ordinamento memorizzando le citate quantità per ogni oggetto dell'insieme dati. Queste informazioni sono sufficienti per generare strutture di clustering basato su densità rispetto a qualsiasi distanza ε' più piccola di quella iniziale ε.

Algoritmo DENCLUE

DENCLUE (*DENsity – based CLUstEring*) è un algoritmo di clustering che si fonda sulla teoria delle *funzioni distribuzione di densità*. Esso è stato sviluppato da (Hinneburg e Keim, 1998). La logica di base sfrutta le seguenti osservazioni: (1) il peso relativo di un punto (oggetto) può essere modellato attraverso un ente matematico, chiamato funzione di influenza, che descrive l'impatto o influenza del punto nella sua vicinanza; (2) la densità complessiva dello spazio dati può essere modellato come somma delle singole funzioni dei punti appartenenti ad esso; (3) i cluster possono essere determinati individuando gli attrattori di densità, ovvero i massimi locali della funzione di influenza complessiva.
La funzione di influenza di un dato punto $y \in F^d$, dove F^d è uno spazio d-dimensionale, è un'applicazione $f_B^y : F^d \to R_0^+$ definita in termini di una funzione di influenza elementare $f_B^y = f_b(x, y)$ con $x \in F^d$. La funzione di influenza elementare può essere una funzione arbitrariamente definita a patto che sia riflessiva e simmetrica, in quanto basata sulla nozione di distanza fra due punti; ad esempio un'onda quadra : $f_{Onda}(x, y) = \begin{cases} 0 & d(x, y) \geq \sigma \\ 1 & altrimenti \end{cases}$ o una Gaussiana: $f_{Gauss}(x, y) = e^{\frac{-d(x,y)}{2\sigma^2}}$. La funzione di influenza complessiva, detta ***funzione di densità***, è la somma delle funzioni di influenza di tutti i punti: $f_B^D = \sum_{i=1}^{N} f_B^{x_i}(x)$ dove D è un insieme di N punti d-dimemsionali rappresentativo degli oggetti. Se la funzione di densità è continua e differenziabile è possibile definire il gradiente, che servirà all'algoritmo per individuare i massimi della funzione.
Un cluster di forma circolare può essere visto come un sottoinsieme C di punti la cui funzione di densità è non inferiore ad una soglia s (se la funzione è minore di s, si ha un outlier); mentre una cluster di forma arbitraria è l'unione di più C, dove esiste un percorso P, da una regione ad un'altra, lungo il quale la funzione non scende sotto la soglia s.

DENCLUE ha molte caratteristiche vantaggiose: esso è fondato su solide teorie matematiche; generalizza gli altri metodi come quelli di partizionamento e gerarchici; può analizzare grandi DB con molto rumore; dà una descrizione matematica compatta dei cluster. Al solito è necessaria una attenta calibrazione dei parametri di soglia (s) e densità (σ).

3.4.5 Metodi basati su griglia

Questi metodi quantizzano i dati in singole celle che vanno a formare una struttura a griglia e le operazioni di raggruppamento sono effettuate su questa struttura. Il vantaggio principale che si ottiene è la velocità di processamento, che è tipicamente indipendente dal numero *n* di oggetti.

Algoritmo STING

STING (*STatistical INformation Grid*) è un metodo basato su una struttura a griglia multi-risoluzione. Secondo questo approccio lo spazio dati è diviso in celle rettangolari, organizzate in una gerarchia: ogni cella al livello i è divisa in varie sottocelle di livello i+1. in ogni cella sono memorizzate informazioni statistiche di base dei punti costituenti: numero di punti, media, massimo, minimo, deviazione standard, tipo di distribuzione dei punti (uniforme, normale, esponenziale, nessuna). In Figura 34 è riportata una struttura gerarchica delle celle.

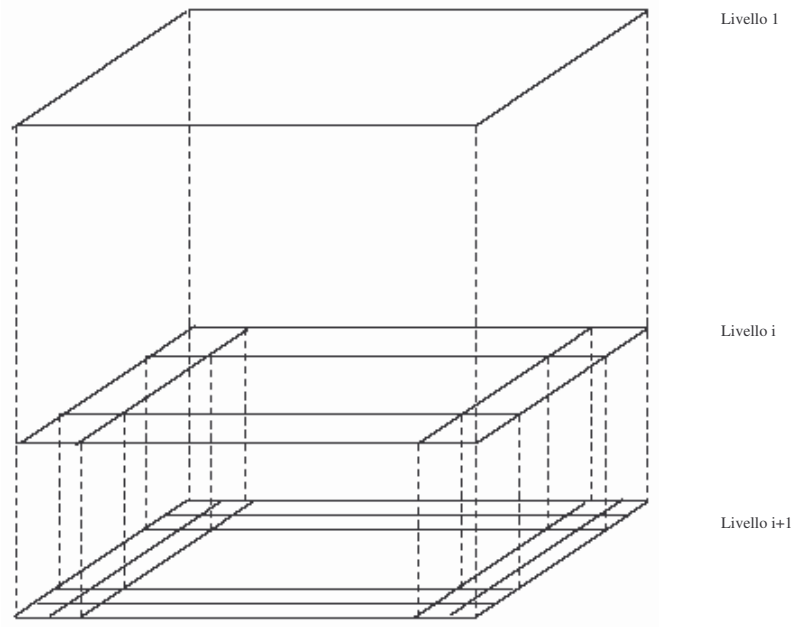


Figura 34: Esempio di celle gerarchiche.

I parametri delle celle di livello superiore sono facilmente calcolabili da quelli di livello inferiore; le distribuzioni di una cella di livello *i* è calcolata dalla distribuzioni più frequenti al livello *i+1*, più un ulteriore processo di filtraggio basato su una soglia. Se le distribuzioni in

una cella sono disomogenee e non superano il test sulla soglia, il tipo della cella superiore è settato a *"none"*.

Dopo che tutti i valori sono stati computati e memorizzati, sono sottoposte delle interrogazioni al sistema. Esso risponde alla query analizzando le informazioni statistiche delle celle a partire da un certo livello della gerarchia; si selezionano le celle calcolando un intervallo di confidenza della probabilità che una cella sia rilevante ai fini della query, le celle non rilevanti sono rimosse ed il processo continua in profondità analizzando solo quelle rilevanti. Una volta raggiunto il livello più basso, il sistema ritorna le celle che verificano le condizioni richieste, altrimenti si recuperano i dati relativi alle celle rilevanti e si effettuano analisi supplementari.

Un primo vantaggio di questo metodo è l'indipendenza dall'interrogazione al sistema, in quanto le informazioni memorizzate in ogni cella sono intrinseche alla sua conformazione, inoltre la struttura a griglia favorisce processamenti paralleli ed aggiornamenti incrementali. Come accennato il vantaggio maggiore è la velocità: infatti il processo di generazione dei cluster ha complessità $O(N)$ (N = numero di oggetti), mentre il processo di analisi della query è $O(G)$ (G = numero di celle al livello più basso, tipicamente molto minore di N), entrambi lineari. Il grado di accuratezza dell'algoritmo dipende comunque dal livello di granularità raggiunto alla fine della gerarchia: se le celle sono a grana fine, il costo del processamento più incrementarsi sostanzialmente; se la grana è troppo grossolana, i risultati potrebbero essere troppo approssimativi. Lo scotto da pagare per la velocità è rappresentato dalla forma "innaturale" dei cluster che avranno solo confini orizzontali e verticali, a causa della conformazione delle celle.

Algoritmo WaveCluster

L'algoritmo è stato proposto da (Sheikholeslami, Chatterjee e Zhang, 1998). Anch'esso si basa su una struttura a griglia multi-risoluzione e successivamente trasforma questo spazio tramite le trasformazioni wavelet per trovare le regioni ad alta densità. In una struttura a griglia, gli attributi degli oggetti sono rappresentati da vettori di features, dove ogni elemento del vettore rappresenta un attributo. Per un oggetto con n attributi vi sarà un vettore n-dimensionale. Le trasformazioni wavelet sono una tecnica di processamento di segnali che decompone il segnale in sottobande di frequenza. Per segnali n-dimensionali si applica una trasformazione monodimensionale n volte. Una volta nel dominio delle frequenze i cluster naturali diventano molto più distinguibili.

Una caratteristica interessante di queste tecniche è che in qualche modo effettuano un clustering non supervisionato, in quanto i filtri wavelet enfatizzano le regioni più dense ma simultaneamente tendono a tagliare le informazioni meno rilevanti situate ai confini. Così queste regioni ad alta densità si comportano da attrattori per i punti più vicini e respingono i punti non vicini abbastanza. Secondo, i filtri passa-basso usati nella trasformazione rimuovono automaticamente gli outliers.

Un algoritmo di clustering basato su filtri Wavelet può operare in questo modo:

```
INPUT (vettori di features)
OUTPUT (clusters)
      Quantizza lo spazio e assegna oggetti alle celle;
      Applica trasformazioni wavelet;
      Trova i cluster connessi nelle sottobande a vari livelli di
      risoluzione;
      Assegna etichette alle unità;
      Assegna oggetti ai cluster;
```

La tecnica ha una complessita lineare $O(N)$.

Nella fig. 35 è rappresentata una struttura di cluster a tre differenti livelli di risoluzione. Ad ogni livello sono riportati i cluster relativi alle varie sottobande di frequenza che definiscono la struttura interna (media), i confini orizzontali, verticali e gli angoli.

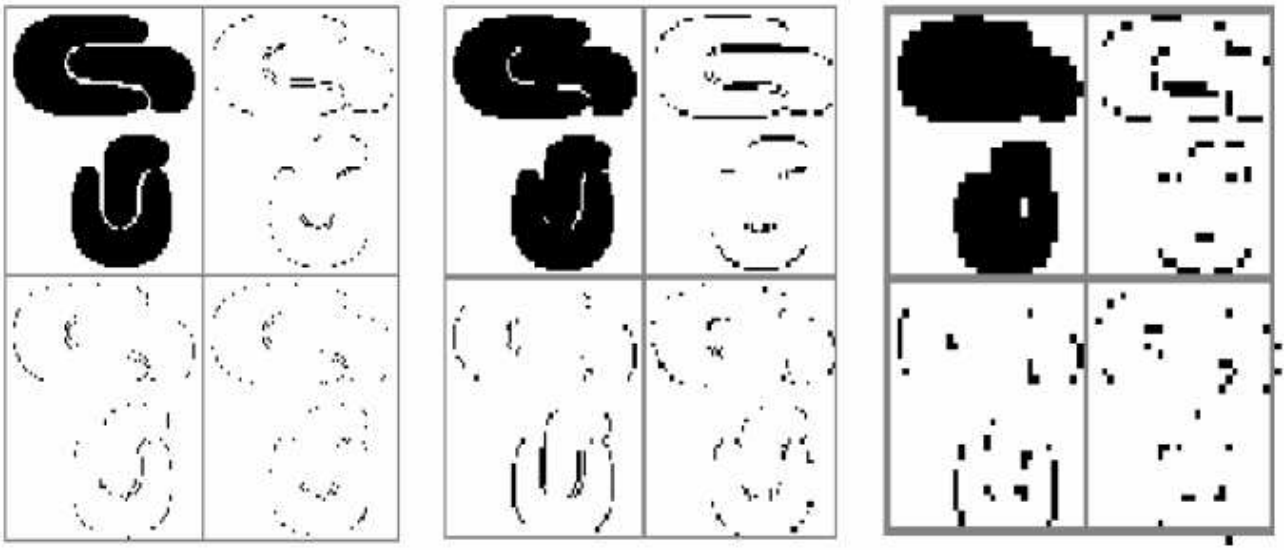


Figura 35: WaveCluster: rappresentazione dei cluster.

WaveCluster è un algoritmo basato su griglia e basato su densità; esso può operare su DB di grandi dimensioni, scovare cluster di forma arbitraria e gestire efficientemente gli outliers.

Algoritmo CLIQUE

CLIQUE è un altro algoritmo basato su griglia e densità e si presta bene al trattamento di dati a dimensionalità elevata. E' stato proposto da (Agrawal et al., 1998). Dato un ampio spazio ad alta dimensionalità, esso individua le sia le regioni sparse che le regioni dense, quindi scopre la distribuzione complessiva di tutto l'insieme. Un unità della struttura è definita densa se una frazione dei punti totali contenuti in essa eccede una certa quantità, determinata da un parametro in input. Un cluster è una connessione di unità dense. CLIQUE partiziona l'insieme di dati *n*-dimensionali in unità rettangolari non sovrapposte, determina le unità dense e trova i cluster nei vari sottospazi sfruttando il principio APRIORI: *se una unità nello spazio k-dimensionale è densa, allora è densa ogni sua proiezione nello spazio k-1-dimensionale; di conseguenza, se nessuna delle proiezioni k-1-dimensionali è densa, allora la relativa unità k-dimensionale non è candidata ad essere densa.* Dopodiché esso dà una descrizione minimale dei cluster individuando le zone di copertura massima e minima che racchiudono ogni cluster. Quindi la procedura determina sottospazi ad alta dimensionalità tali che contengono cluster densi. Il metodo è semplice, insensibile all'ordinamento e non richiede nessuna distribuzione iniziale dei dati; la complessità scala linearmente sia con l'aumentare del numero di oggetti che del numero di dimensioni. Per contro l'accuratezza può non essere sufficiente a causa della semplicità del metodo.

3.4.6 Metodi basati sui modelli

Un metodo basato su modello ipotizza un modello per ogni cluster e cerca di trovare il sottoinsieme dati che meglio si adatta ad esso in tutto lo spazio. Esso localizza i cluster tramite le funzioni di densità, come visto prima, che descrivono la distribuzione dei punti nello spazio; inoltre costituisce un metodo di determinazione automatica del numero di cluster tramite le misure statistiche standard, prendendo in conto anche rumore ed outliers.

Ad esempio COBWEB (Fisher, 1997) effettua una clustering incrementale su attributi categorici sfruttando una misura (category utility) per assegnare un punteggio ad ogni nuovo inserimento. Il suo limite più evidente, però, è basato sull'assunzione che le distribuzioni su attributi distinti sono statisticamente indipendenti, mentre, in qualche misura, una correlazione fra attributi e valori degli attributi esiste sempre. Inoltre anche l'albero costruito per la classificazione può risultare non bilanciato in altezza per dati molto deviati. Un altro esempio è CLASSIT che lavora analogamente a COBWEB ma su dati continui (valori reali); usa una misura di category utility che è l'integrale sui valori continui invece che la somma su valori discreti, come in COBWEB.

3.5 Regole di associazione

Le regole di associazione cercano le relazioni di correlazione, che sono interessanti ai fini del problema, in grandi volumi di dati. Tali dati sono collezionati dall'azienda che intende sviluppare il progetto dai propri sistemi transazionali. Una tipica applicazione che viene usata a scopo esemplificativo è la market basket analysis, che analizza le tendenze e il comportamento di acquisto dei consumatori esaminando i loro carrelli della spesa. I risultati di questa analisi possono essere usati per pianificare strategie di marketing e pubblicitarie così come lo sviluppo del catalogo dei prodotti, ma anche suggerire il posizionamento dei prodotti sugli scaffali all'interno del locale, ovvero la progettazione del layout del negozio. Ad esempio se una regola dice che chi compra un computer probabilmente comprerà anche un software di gestione finanziaria, un approccio è quello di disporre su scaffali vicini i due prodotti per incentivare il cliente all'acquisto di entrambi; un'altra strategia è quella di posizionare i prodotti alle estremità del locale, cosicché il cliente intenzionato all'acquisto degli articoli possa notare e comprare altri articoli durante il tragitto. Ancora, le regole di associazione possono suggerire anche le politiche promozionali ovvero quali articoli mettere in vendita a prezzo ridotto e come comporre i pacchetti in promozione; se è noto da una regola che chi compra un computer compra anche una stampante si possono vendere i due articoli in un'unica promozione, ciò verosimilmente incoraggerà l'acquisto di entrambi.

3.5.1 Concetti di base

Indichiamo con $I = \{i_1 ... i_m\}$ un insieme di oggetti, con D un insieme di transazioni T tali che $T \subseteq I$, dove ogni transazione è identificata da un TID (Transaction ID). Siano A e B due sottoinsiemi qualunque di oggetti $A, B \subseteq I$ tali che $A \cap B = \varnothing$, una regola di associazione è espressa nella forma $A \Rightarrow B$. La regola è descritta da due parametri caratteristici: il *supporto* e la *confidenza*. Il supporto *s* di una regola è la percentuale con cui si trovano congiuntamente gli insiemi A e B rispetto a D, più formalmente $\sup porto(A \Rightarrow B) = prob(A \cup B)$; la confidenza *c* è la percentuale con cui si trova B rispetto alle volte che si è trovato A, ovvero $confidenza(A \Rightarrow B) = prob(B \mid A)$.

Se la regola soddisfa sia una soglia minima di supporto (min_sup) che di confidenza (min_conf) è detta forte. Nella letteratura specializzata un insieme di oggetti è chiamato itemset, se contiene k oggetti è detto *k*-itemset; la frequenza di occorrenza *f* di un itemset (o il conteggio del supporto) è il numero di transazioni che contengono l'itemset nell'insieme D.

Se il supporto di un itemset supera una soglia, l'itemset è detto frequente. L'insieme dei k-itemset frequenti è denotato con L_k.

Le regole sono desunte dall'insieme dati in due passi: (1) trovare tutti gli itemset frequenti basati su una certa soglia, definita dall'utente o dall'esperto di dominio e (2) generare una regola forte dagli itemset frequenti, ovvero una regola che soddisfi supporto e confidenza predeterminati.

3.5.2 Tassonomia delle regole

Le regole possono essere classificate in vari gruppi secondo vari criteri:

TIPO DI VALORI

Se una regola concerne la presenza o l'assenza di un oggetto, essa è una regola di associazione Booleana. Ad esempio la regola:

`compra(X, "computer")` $\Rightarrow$ `compra(X, "software gestionale").`

Se descrive l'associazione fra attributi quantitativi è detta regola di associazione quantitativa, un esempio è la seguente:

`età (X, "30-34") AND reddito (X, "40000-50000")` $\Rightarrow$
`compra (X, "TV alta definizione").`

In queste regole gli attributi coinvolti (età e reddito) sono opportunamente discretizzati in intervalli.

DIMENSIONI DEI DATI

La prima delle regole precedenti è anche monodimensionale in quanto riguarda la sola dimensione compra, mentre la seconda è multidimensionale (età, reddito, compra).

LIVELLO DI ASTRAZIONE

Prendiamo ad esempio un insieme di due regole del tipo:

`età(X, "30-35")` $\Rightarrow$ `compra ("notebook")` e `età(X, "30-35")` $\Rightarrow$ `compra("computer").`

Questa è una regola di associazione multilivello in quanto i livelli di astrazione sono diversi, come si può notare dai secondi membri. Se ci si attiene ad un solo livello la regola è detta a singolo livello.

3.5.3 Algoritmo APRIORI

L'algoritmo APRIORI è la tecnica più utilizzata per costruire regole di associazione booleane monodimensionali. Esso impiega una ricerca iterativa a livelli dove la conoscenza dei k-itemset frequenti è utilizzata per esplorare e trovare i k+1-itemset frequenti, sfruttando la cosiddetta proprietà Apriori: tutti i sottoinsiemi non vuoti di un k-itemset frequente sono

anch'essi frequenti. Questa proprietà appartiene alla categoria delle proprietà anti-monotone nel senso che se un insieme non supera un test, qualsiasi suo superinsieme non supererà il test. Vediamo come l'algoritmo trova l'insieme L_k a partire da L_{k-1}. Esso procede in due passi:

1. **Passo di join**: costruzione di un insieme C_k di k-itemset candidati ad appartenere ad L_k effettuando il seguente join: $L_{k-1} \times L_{k-1} = \{A \times B \mid A, B \in L_{k-1}, |A \cap B| = k-2\}$.
2. **Passo di taglio**: C_k è un superinsieme di L_k, ora si determinerà il supporto di ogni elemento di C_k effettuando una scansione del DB e contando le occorrenze di ogni k-itemset. Prima però, per ridurre l'insieme C_k si sfrutta la proprietà Apriori escludendo da esso ogni k-itemset per il quale si sia trovato almeno un k-1-itemset contenuto in esso che sia non frequente, ovvero che non appartenga a L_{k-1}.

Riportiamo di seguito lo pseudo-codice dell'algoritmo organizzato in un main program e due procedure ausiliarie.

```
main_Apriori()
INPUT (database delle transazioni: D; soglia di supporto minima: min_sup)
OUTPUT (insieme degli itemset frequenti: L)

L1=trova_1-itemset_frequenti(D);
FOR (k=2; L_{k-1} ≠ ∅ ;k++)
      Ck=join_taglia(Lk-1,min_sup);
      FOR EACH transazione t {      //scansione DB
            Ct=sottoins(Ck,t)       //trova sottoinsiemi di t candidati
            FOR EACH candidato c c.count++
      }
      L_k = {c ∈ C_k | c.count ≥ min_sup};
}
***
procedure_join_taglia(Lk-1,min_sup)
FOR EACH itemset l1
      FOR EACH itemset l2
            IF (l1[1]=l2[1]) ∧ (l1[2]=l2[2]) ∧ ... ∧ (l1[k-1]<l2[k-1]) {
                  THEN c = l1 × l ;     //passo di join:genera candidati
                  IF sottoins_non_frequenti(c,Lk-1)
                  THEN delete c;          //passo di taglio
                  ELSE aggiungi c a Ck;
            }
return Ck
***
procedure sottoins_non_frequenti(c,Lk-1) //usa conoscenza pregressa
FOR EACH k-1-itemset s di c
      IF (s ∉ L_{k-1})
      THEN return TRUE;
return FALSE;
```

Si noti che l'algoritmo assume che gli oggetti negli itemset siano ordinati lessicograficamente. La funzione `sottoins`, usata nel main e non definita altrove, trova tutti i sottoinsiemi di una data transazione t che sono candidati, per poi consentire il conteggio delle occorrenze.

Generazione delle regole

Una volta trovati gli itemset frequenti si può procedere ad esprimere le regole forti che li riguardano. Ricordando le proprietà delle regole forti (e cioè che devono rispettare una soglia minima di supporto e confidenza), possiamo esprimere la confidenza in termini di supporto grazie alla equazione: $confidenza(A \Rightarrow B) = prob(B \mid A) = \frac{\sup porto(A \cup B)}{\sup porto(A)}$, dove $\sup porto(A \cup B)$ è il numero di transazioni contenenti l'itemset $(B \cup A)$ e $\sup pporto(A)$ è il numero di transazioni contenenti l'itemset A.

Le regole possono essere costruite in questo modo:

- Per ogni itemset frequente *l*, generare tutti i sottoinsiemi non vuoti di *l*.
- Per ogni sottoinsieme non vuoto *s*, generare una regola $(s \Rightarrow l - s)$ se $\frac{\sup porto(l)}{\sup porto(s)} > \min\_conf$.

Siccome le regole sono derivate da itemset frequenti, esse soddisferanno automaticamente la soglia di supporto. Gli itemset frequenti possono essere memorizzati in memoria centrale insieme ai conteggi, così da essere velocemente accessibili.

Miglioramenti dell'algoritmo

Sono state proposte molte varianti per l'algoritmo APRIORI che si basano sulle seguenti tecniche:

TABELLA HASH

Questa tecnica riduce l'insieme dei k-itemset candidati C_k per k > 1 . Quando si effettua la scansione di ogni singola transazione per costruire L_1 da C_1, si possono allo stesso tempo generare tutti i possibili 2-itemset della transazione; poi si mappano, grazie ad una opportuna funzione hash, in una struttura tabellare e si contano le occorrenze di ogni campo della tabella. Un 2-itemset il cui conteggio in tabella non supera la soglia di supporto viene escluso da C_2. Nella tabella seguente è riportata un esempio di tabella hash che utilizza la funzione h(x,y) per la mappatura.

Indirizzo=h(x,y)	0	1	2	3	4	5
Conteggio	3	5	8	1	2	3
Contenuto	{I1,I4}…	…	…	…	…	…

RIDUZIONE DELLE SCANSIONI

Si ricordi che è necessaria una scansione dell'intero insieme dati per costruire L_k da C_k. Si può avere riduzione delle scansioni effettuando operazioni extra durante una scansione. Infatti, è noto che C_3 è costruito da $L_2 \, join L_2$ ma esso potrebbe ottenersi anche da $C_2 \, join C_2$. Diciamo C'_3 l'insieme dei candidati ottenuti da $C_2 \, join C_2$, sicuramente $|C'_3| > |C_3|$, ma se C'_3 non è molto più grande di C_3 e sia C_2 che C'_3 possono essere mantenuti in memoria centrale, allora è possibile calcolare L_2 ed L_3 simultaneamente durante la stessa scansione, risparmiandone una. Con questa strategia possiamo calcolare tutti gli L_k con 2 sole scansioni: una per determinare L_1 ed un'altra per i rimanenti $L_2 ... L_k$, assumendo che $C'_k = C'_{k-1} \, join C'_{k-1}$ e che per k > 2 essi possano essere mantenuti in memoria centrale.

RIDUZIONE DELLE TRANSAZIONI

Una transazione che non contiene k-itemset frequenti non potrà contenere k+1-itemset frequenti. Quindi una transazione potrà essere marcata per non prenderla più in considerazione nelle future iterazioni.

PARTIZIONAMENTO

Effettuando un partizionamento dei dati sono sufficienti 2 sole scansioni per trovare gli itemset frequenti. In primo luogo viene suddiviso l'intero insieme D in n partizioni non sovrapposte ed in ognuna di queste vengono trovati gli itemset frequenti locali (1° scansione), usando tipicamente una soglia di supporto minore della soglia "globale". Infatti un itemset potenzialmente frequente rispetto a D risulta frequente in almeno una delle n partizioni. Per ognuna di esse il processo è eseguito completamente in memoria centrale, memorizzando i risultati. Nella seconda fase (2° scansione) vengono trovati gli itemset frequenti globali con il parametro assoluto di supporto. Se in tale fase qualcuno degli itemset frequenti globali viene mancato si deve effettuare un ulteriore scansione.

3.5.4 Estrazione di regole multilivello

In molte applicazioni può risultare difficile ritrovare associazioni fra articoli ad un basso livello di astrazione, per questo si cerca di estrarre informazioni e formare le regole a livelli superiori della gerarchia concettuale. Inoltre, sondare i dati da un punto di vista più generale ne facilita l'interpretazione perché ci si avvicina alla conoscenza di senso comune.

Se, ad esempio, si stanno analizzando i dati di vendita di una grossa casa di elettronica di consumo, una possibile gerarchia strutturata ad albero può essere quella in Figura 36.

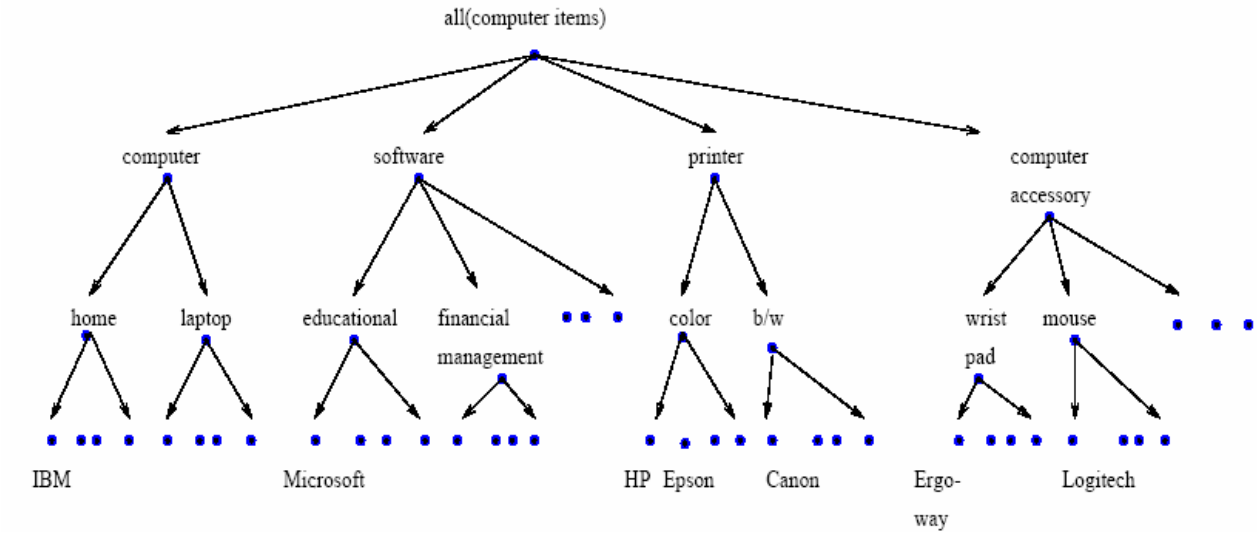


Figura 36: Gerarchia concettuale per una casa venditrice di articoli elettronici.

cercando le possibili correlazioni fra un "mouse Logitech" e un "home computer IBM" (livello 3: dettaglio massimo della struttura) è probabile che non si raggiungerà la soglia di supporto minimo; ma se ci si muove al livello di astrazione superiore si hanno grandi possibilità di soddisfare il supporto analizzando le relazioni fra "home computer" e "mouse" (livello 2), in quanto i due articoli sono quasi sempre acquistati insieme.

Si capisce quindi fin da ora l'utilità di strutturare i dati a disposizione in gerarchie concettuali.

Vi sono vari approcci per estrarre regole multilivello, ma tutte si basano su una strategia di indagine top-down della struttura ad albero, accumulando i conteggi degli itemset ad ogni livello utilizzando l'algoritmo APRIORI o una delle sue varianti.

SUPPORTO UNIFORME

La stessa soglia di supporto minimo è usata per ogni livello si astrazione. Tale metodo è il più semplice in quanto richiede all'utente la specifica di un unico parametro. Nella figura seguente si vede che gli itemset contenenti computer da tavolo sono considerati non frequenti.

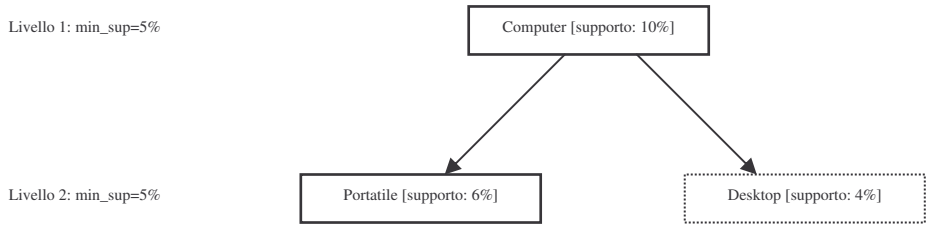


Figura 37: Medesimo supporto per astrazioni differenti.

un metodo di ottimizzazione potrebbe basarsi sulla conoscenza che ogni nodo padre è un superinsieme dei propri figli, e quindi evitare di analizzare itemset relativi ai figli che contengono articoli che non verificano il supporto a livello padre.

Questo approccio comunque ha delle controindicazioni in quanto è verosimile che articoli che occorrono con una certa frequenza ad un certo livello non si presenteranno ugualmente a livelli inferiori, quindi una soglia costante può far perdere associazioni importanti ai livelli più bassi.

SUPPORTO RIDOTTO

Ogni livello ha la propria soglia di supporto minimo. Più basso è il livello di astrazione minore è la soglia. Vi sono varie strategie di ricerca:

- *Indipendente*: è una ricerca completa lungo tutto l'albero, senza nessun taglio di nodi figlio.
- *Filtraggio su singolo articolo*: un articolo a livello i è esaminato solo se è trovato frequente a livello i-1.

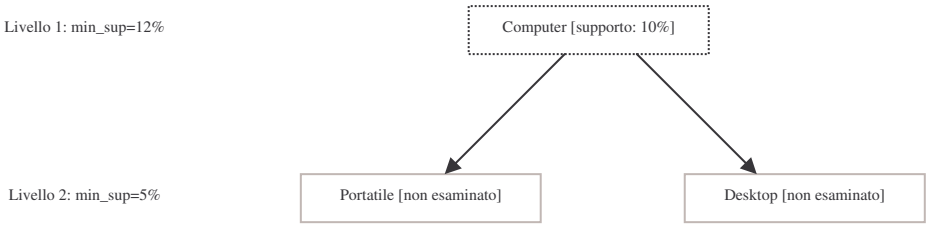


Figura 38: Supporto minimo dipendente dal livello gerarchico. Caso di articolo singolo.

questo metodo può perdere associazioni perché se un item a livello i è frequente ma il suo superset a livello i-1 non lo è, esso verrà escluso da ulteriori analisi.

- *Filtraggio su k-itemset*: un k-itemset a livello i è analizzato se e solo se il suo k-itemset padre a livello i-1 è trovato frequente.

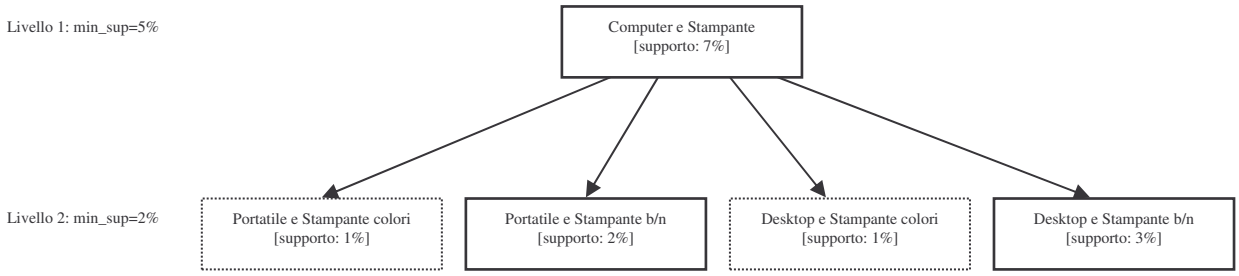


Figura 39: Supporto minimo dipendente dal livello gerarchico. Caso di 2-itemset.

guardando la figura si deduce che i figli del 1° e del 3° nodo non saranno ulteriormente esaminati.

Una versione modificata del filtraggio su singolo articolo è il filtraggio controllato su singolo articolo. Per rendere meno restrittivi i vincoli viene introdotta la cosiddetta soglia di passaggio per far passare gli articoli "abbastanza" frequenti e consentire analisi più approfondite. In altre parole si permette l'analisi dei figli di item che sono non frequenti ma che superano la soglia di passaggio. Tale soglia si sceglie fra i valori compresi fra il supporto minimo del nodo figlio (o la media se vi sono più figli) e il supporto minimo del nodo in questione.

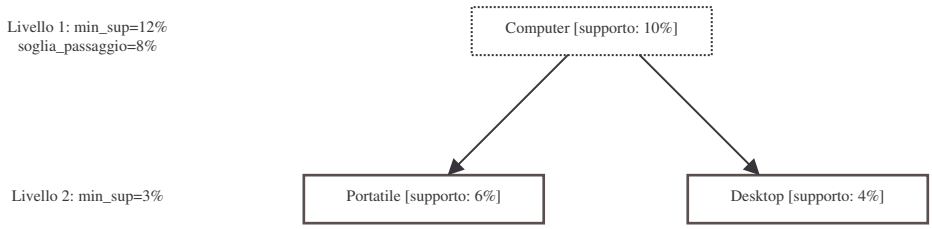


Figura 40: Ricerca con "soglia di passaggio".

3.5.5 Estrazione di regole multidimensionali

Siccome la maggior parte dei dati che sono sottoposti ad attività di mining proviene da sistemi Data Warehouse, le regole più interessanti sono quelle che coinvolgono più informazioni di varia natura, ovvero lungo le diverse dimensioni della struttura multidimesionale. Le regole con predicati differenti e non ripetuti sono dette *interdimensionali*:

età(X, "19-24") AND occupazione(X, "studente") $\Rightarrow$ compra("portatile")

Le regole con più predicati dove qualcuno di essi è ripetuto sono dette *a dimensione ibrida*:

età(X, "19-24") AND compra("portatile") $\Rightarrow$ compra("stampante b/n")

E' noto che gli attributi di un warehouse possono essere sia categorici che numerici; i metodi di estrazione di regole multidimensionali differiscono sulla base del trattamento degli attributi numerici o quantitativi, ovvero a valori continui.

1. il primo approccio consiste nella discretizzazione statica degli attributi numerici, ovvero nel partizionamento dell'intero range di valori in vari intervalli, identificati poi

da un attributo nominale. Ad esempio, per l'attributo reddito si può pensare ad una decomposizione negli intervalli: "0-20000" – "21000-30000" ecc. La composizione degli intervalli non cambia nel tempo.

2. nel secondo approccio gli attributi vengono suddivisi in "recipienti" (bins) a seconda della loro distribuzione, secondo un processo dinamico per venire incontro a specifiche esigenze di estrazione, come massimizzare la confidenza. Tale metodo è detto quantitativo.
3. nel terzo, gli attributi vengono suddivisi in modo da catturare il significato semantico degli intervalli, nel senso di misurare la distanza relativa tra essi. Infatti il metodo è detto basato sulla distanza.

In questa ricerca, a cambiare è l'oggetto della ricerca. Prima si cercavano i k-itemset frequenti, ora invece i *k-predicateset* frequenti. Un k-predicateset è un insieme di k predicati congiunti, ad esempio{età, reddito, compra} è un 3-predicateset.

DISCRETIZZAZIONE STATICA

Se i dati oggetto dello studio sono contenuti in una tabella relazionale, sono necessarie lievi modifiche per adattarla all'estrazione (come la ricerca degli attributi rilevanti).

La struttura che più si presta a questo tipo di analisi sono i cubi multidimensionali (cfr. par. 1.4.3), infatti ogni singola cella di un cubo rappresenta il conteggio (supporto) di un n-predicato. Per trovare i k-predicateset frequenti è usata una strategia molto simile all'algoritmo APRIORI che sfrutta una proprietà di monotoni analoga: "ogni sottoinsieme di un k-predicateset frequente è anch'esso frequente".

APPROCCIO QUANTITATIVO

Gli attributi numerici possono avere un grande dominio di valori, per trattarli più agevolmente si possono suddividere in "bins" (recipienti) che possono avere stesso intervallo (equi-width binning), stessa cardinalità (equi-depth binning) e dimensione tale che ogni recipiente abbia i valori uniformemente distribuiti (homogeneity based binning).

Si pensi ad una regola generica del tipo: $A_{q1} \wedge A_{q2} \Rightarrow A_c$, dove A_{q1}, A_{q2} sono attributi numerici e A_c è di tipo categorico. Una tecnica per estrarre delle regole di questo tipo è ARCS (Association Rule Clustering System), essa mappa su una griglia bidimensionale le occorrenze degli attributi A_{q1}, A_{q2} dato l'attributo A_c e poi precede alla clusterizzazione. I cluster trovati racchiudono i punti (dove ogni punto rappresenta un'occorrenza di un predicateset) che contribuiscono alla formazione di una regola forte.

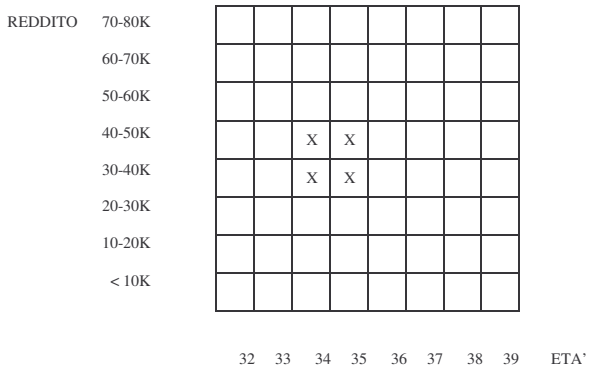


Figura 41: Griglia bidimensionale rappresentante le occorrenze di un 2-predicateset.

Se le "x" nella griglia rappresentano i cluster rettangolari trovati da un algoritmo di clustering basato su griglia (cfr. par.) la corrispondente regola derivata è:

età(X, "33-34") AND reddito (X, "40000-60000") ⇒
compra (X, "TV alta definizione")

APPROCCIO BASATO SULLA DISTANZA

La suddivisione in intervalli basata sulla distanza è la più intuitiva ma anche la più realistica. Infatti i metodi equi-width e equi-depth possono creare intervalli privi di dati o separare valori molto vicini tra loro. L'approccio basato sulla distanza forma i recipienti sulla base della loro vicinanza, misurata con le varie metriche, ed inoltre permette l'approssimazione dei valori. Una regola del tipo:

tipo_articolo(X, "elettronico") AND produttore(X, "Sony") ⇒ prezzo(X, "€200")

prenderà in considerazione solo gli articoli che costano esattamente 200 euro, mentre sarebbe desiderabile un minimo di elasticità considerando un intervallo di prezzi di una certa ampiezza, avente 200 come punto medio.

Le metriche adoperate sono quelle citate nel par. 3.4.1, fra cui la distanza Euclidea e di Manhattan.

3.5.6 Analisi di correlazione

Avendo a disposizione un insieme di regole forti (ovvero che soddisfino le soglie minime di supporto e confidenza) estratte da un insieme dati, ci si potrebbe chiedere se esse sono tutte significative per l'applicazione e se possono rappresentare una valida base per scelte strategiche dell'utente. Ciò in generale non è vero, infatti alcune regole forti possono essere fuorvianti. Per illustrare questo si faccia riferimento al seguente esempio. Si vogliano

analizzare 10000 transazioni di vendita di videogames e DVD; i dati sono riportati in una tabella di contingenza:

	$game$	$\overline{game}$	$\sum_{riga}$
dvd	4000	3500	7500
$\overline{dvd}$	2000	500	2500
$\sum_{col}$	6000	4000	10000

Tabella 2: esempio di tabella di contingenza su dati di vendita di videogames e dvd.

Nelle celle sono contenuti il numero di transazioni che contengono o non contengono i due articoli.

La correlazione fra due eventi A e B può essere misurata tramite l'espressione: $C_{A,B} = \frac{P(A \cap B)}{P(A)P(B)}$; se il risultato è minore di 1, gli eventi si dicono negativamente correlati, ovvero l'occorrenza di uno scoraggia l'occorrenza dell'altro; se è maggiore di 1, essi sono positivamente correlati; mentre un rapporto unitario significa che gli eventi sono statisticamente indipendenti.

Supponiamo ora di usare soglie minime pari a min_sup = 30%, min_conf = 40% e di aver estratto la regola:

`compra(X, "game")` $\Rightarrow$ `compra (X, "dvd")` [supporto = 40%, confidenza = 66%]

essa è forte in quanto soddisfa i limiti minimi ma la regola può essere fuorviante, infatti calcolando la correlazione: $C_{game,dvd} = \frac{0.4}{0.6 \cdot 0.75} = 0.89$ si vede che il rapporto è minore dell'unità e quindi gli eventi sono negativamente correlati, ciò confuta la regola forte.

Questo motiva l'estrazione di regole in cui le occorrenze degli oggetti costituenti siano correlati statisticamente, dette regole di correlazione. Un vantaggio della proprietà di correlazione è che è chiusa superiormente, ovvero ogni superinsieme di un insieme correlato (cioè un insieme ove gli elementi siano positivamente correlati) è anch'esso correlato. Ciò significa che aggiungendo arbitrariamente elementi ad un insieme correlato, la proprietà si mantiene. Sfruttando questa osservazione, per ricercare insiemi correlati da dove estrarre regole di correlazione gli algoritmi preposti lavorano in questo modo: si parte da un insieme vuoto e si aggiunge un elemento alla volta nello spazio degli item, calcolando dinamicamente la correlazione si giunge ai cosiddetti itemset correlati minimi, ovvero quegli insiemi che sono correlati ma che nessun loro sottoinsieme è correlato. A questo punto, vista la proprietà di chiusura, è inutile continuare la ricerca verso l'alto. In questi insiemi trovati e nei loro superinsiemi è possibile applicare le tecniche classiche per la ricerca delle regole forti.

3.5.7 Estrazione guidata da metaregole

Come si è già fatto notare non tutte le regole estratte da una magazzino dati può interessare l'utente per i suoi scopi, specialmente quando il numero di regole diventa consistente. Non è raro che, in grosse basi di dati, il numero di regole sia nell'ordine delle migliaia.

Per limitare il campo e guidare la ricerca del sistema l'utente può specificare dei limiti, o meglio, dei vincoli (constraint-based mining) di varia natura:

1) Vincoli sul tipo di conoscenza: una specifica sul tipo di conoscenza da estrarre, come una associazione.
2) Vincoli sui dati: specifiche sull'insieme dei dati rilevanti.
3) Vincoli di dimensione / livello: specifiche sulla dimensione dei dati o sul livello gerarchico in cui essi sono collocati.
4) Vincoli di interesse (interestingness): le specifiche di soglie minime o misure statistiche, come supporto e confidenza
5) Vincoli sulle regole: questo tipo di vincoli specifica la forma delle regole da estrarre. Essi sono espressi tramite le metaregole.

L'ultima tipologia di vincoli sono espressi tramite metaregole ovvero modelli generali delle regole (rule templates). Essi permettono all'utente di specificare la forma sintattica delle regole alle quali sono interessati, aumentando sia l'efficienza che l'efficacia del processo di scoperta. Le metaregole sono perlopiù basate sulla conoscenza e l'esperienza dell'analista, su intuizioni sui dati oppure generate automaticamente sullo schema dati.

Il modello generale di una metaregola è: $P_1 \wedge P_2 \wedge ... \wedge P_l \Rightarrow Q_1 \wedge Q_2 \wedge ...Q_r$, dove $P_i (i = 1...l)$ e $Q_j (j = 1...r)$ sono variabili predicato che comporranno i predicateset frequenti. Sia $p = l + r$ il numero di predicati della regola. Per trovare una regola che soddisfi il modello si ha bisogno di due informazioni principali:

- Tutti i p-predicateset frequenti, L_p.
- Il supporto (o conteggio) di tutti gli l-predicateset sottoinsiemi di L_p, per computare la confidenza della regola.

Ad esempio, si supponga di avere accesso a tutti i dati sui clienti di una multinazionale (nome, età, sesso, reddito, ecc.) e piuttosto che essere interessati alle tipologie di prodotti acquistati, si vuole indagare sul profilo del cliente che compra software educativo. Una metaregola che descriva le informazioni desiderate può essere

$P_1(X,Y) \wedge P_2(X,W) \Rightarrow$ `compra (X, "software educativo")`

dove X è una variabile che rappresenta il cliente, come un identificatore, mentre Y e W sono attributi istanziati dal processo di ricerca associati ai predicati P_1 e P_2. Tipicamente è l'utente a specificare una lista di attributi da associare ai predicati P_1 e P_2, altrimenti è usato un insieme di default.

Questo è un tipico caso di estrazione di regole multidimensionali. La struttura dati più diffusa e che meglio si presta a questo scopo è il cubo multidimensionale, vista la sua caratteristica di contenere informazioni già aggregate. Sovente negli strumenti OLAP si hanno a disposizione cubi n-dimensionali già preparati sui dati, con n numero di attributi. In tal caso si ha la necessità di esaminare solo i cubi p-dimensionali relativi alle variabili dei predicati e comparare i valori delle celle con i valori minimi di supporto per trovare L_p. I cubi l-dimensionali sono già calcolati e contenuti nei p-dimensionali, quindi il calcolo della confidenza non è complesso. A questo punto può essere invocata una procedura che generi la regola che sia conforme alla modello.

3.5.8 Classificazione basata sulle associazioni

Recentemente sono state sviluppate tecniche di DM che applicano la deduzione di regole di associazione a scopi di classificazione. Un metodo di questi è la classificazione associativa. Essa consiste in due fasi. In un primo momento vengono generate le regole di classificazione sfruttando una versione modificata dell'algoritmo APRIORI; durante il secondo passo viene costruito un classificatore basato sulle regole estratte.

Sia D l'insieme dei dati di apprendimento e Y l'insieme delle classi in D. l'algoritmo mappa attributi categorici e continui in interi positivi consecutivi. Ogni campione d in D è rappresentato da una coppia {attributo, valore intero}, detta item, e da un'etichetta di classe y. Sia I l'insieme di tutti gli item (coppie) in D. Una regola di associazione di classe (CAR, Class Association Rule) ha una forma generale del tipo condset $\Rightarrow$ y, dove condset è un sottoinsieme di item, $condset \subseteq I$ e $y \in Y$. Una regola di questo tipo può essere rappresentata nella forma < condset, y >, detta ruleitem.

Una CAR ha confidenza c se il c% dei campioni in D che contengono condset hanno etichetta y. Una CAR ha supporto s se il s% dei campioni in D contengono condset ed hanno etichetta y. Il conteggio di supporto di un condset (condsupCount) è il numero assoluto di campioni contenenti il condset; il conteggio delle regole di un ruleitem è il numero assoluto di campioni contenenti il condset ed etichettati con y. Un ruleitem che soddisfi supporto minimo è chiamato ruleitem frequente. Un ruleitem che soddisfi confidenza minima è detto accurato. Se

un insieme di ruleitem ha lo stesso condset, allora viene scelta la regola avente confidenza più alta come rappresentante dell'insieme e viene chiamata regola possibile (PR, Possible Rule).
L'algoritmo trova tutte le PR sia frequenti che accurate, con un approccio iterativo simile a quello descritto nella sezione 3.5.3, dove, piuttosto che essere processati i k-itemset, la ricerca prende in esame i k-ruleitem (il cui condset contiene k item). I k-ruleitem sono usati per esplorare i (k+1)-ruleitem sfruttando la proprietà di anti-monotonicità.
Il secondo passo processa le CAR trovate per costruire il classificatore. Siccome l'insieme di regole può essere molto grande, è effettuato un ordinamento preliminare basato su precedenza: una regola r_i ha precedenza su una regola r_j se:

- $confidenza(r_i) > confidenza(r_j)$
- $confidenza(r_i) = confidenza(r_j)$ ma $\sup porto(r_i) > \sup porto(r_j)$
- $confidenza(r_i) = confidenza(r_j)$, $\sup porto(r_i) > \sup porto(r_j)$ ma r_i è stata generata prima di r_j.

Il classificatore mantiene le CAR secondo l'ordine di precedenza da alta a bassa, e quando deve classificare un nuovo oggetto gli assegna l'etichetta della prima regola che esso soddisfa.

3.6 Alberi decisionali

Un albero decisionale è un digramma di flusso avente struttura ad albero, dove ogni nodo interno rappresenta un test su un attributo di partizionamento, ogni ramo denota il risultato del test ed ogni nodo foglia rappresenta una classe di appartenenza. Di seguito si riporta una procedura iterativa di base per la costruzione di un albero dato un insieme di apprendimento (training set) a scopi di classificazione, ciò significa che tutti gli attributi coinvolti sono categorici o discretizzati da valori continui.
L'algoritmo riportato è una versione di ID3, uno dei più conosciuti per la costruzione di alberi.
`campioni` indica l'insieme di apprendimento e `lista_attributi` la lista degli attributi degli oggetti coinvolti.

```
Procedure Genera_albero_dec
INPUT (campioni, lista_attributi);
OUTPUT (albero);

Crea un nodo N;
IF campioni sono tutti della stessa classe C
      THEN ritorna N come nodo foglia ed etichetta con C;
IF lista_attributi è vuota
```

```
      THEN ritorna N come nodo foglia ed etichetta con la classe più
comune;
seleziona attributo_test in lista_attributi con il guadagno più alto;
etichetta N con attributo_test;
PER OGNI valore ai di attributo_test
      Inserisci ramo da N per la condizione attributo_test=ai;
      memorizza in si il sottoinsieme di campioni con attributo_test=ai;
IF si è vuoto
      Attacca al ramo un nodo foglia con la classe più comune in campioni;
ELSE attacca al ramo il nodo generato da Genera_albero_dec(si,
lista_attributi);
```

La strategia alla base è la seguente: l'albero parte come degenere ovvero con un solo nodo che rappresenta l'intero insieme di apprendimento. Se gli oggetti appartengono tutti alla stessa classe il nodo viene etichettato con l'etichetta più comune (votazione a maggioranza), altrimenti l'algoritmo usa una misura, detta guadagno di informazione, per selezionare l'attributo che meglio effettua il partizionamento dei dati, sul quale eseguire il test. Viene creato un ramo per ogni valore noto dell'attributo di test e l'insieme dati è partizionato di conseguenza. L'algoritmo usa questa procedura ricorsivamente per ogni partizione dei dati; una volta che un attributo è stato testato in un nodo non verrà preso in considerazione nei nodi discendenti.

Le condizioni di arresto si verificano quando una delle 3 condizioni seguenti è vera:

I. Tutti i campioni ad un dato nodo appartengono alla stessa classe.
II. Non vi sono attributi residui sui quali effettuare ulteriori partizionamenti.
III. Non vi sono campioni aventi il valore `attributo_test=ai`.

Di seguito è raffigurato (fig. 42) la struttura generica di un albero per classificare oggetti in due classi (Classe 1, Classe 2), in cui sono effettuati test su 4 attributi diversi. Si noti che da un nodo interno possono partire un numero arbitrario di rami, a seconda dei valori distinti dell'attributo.

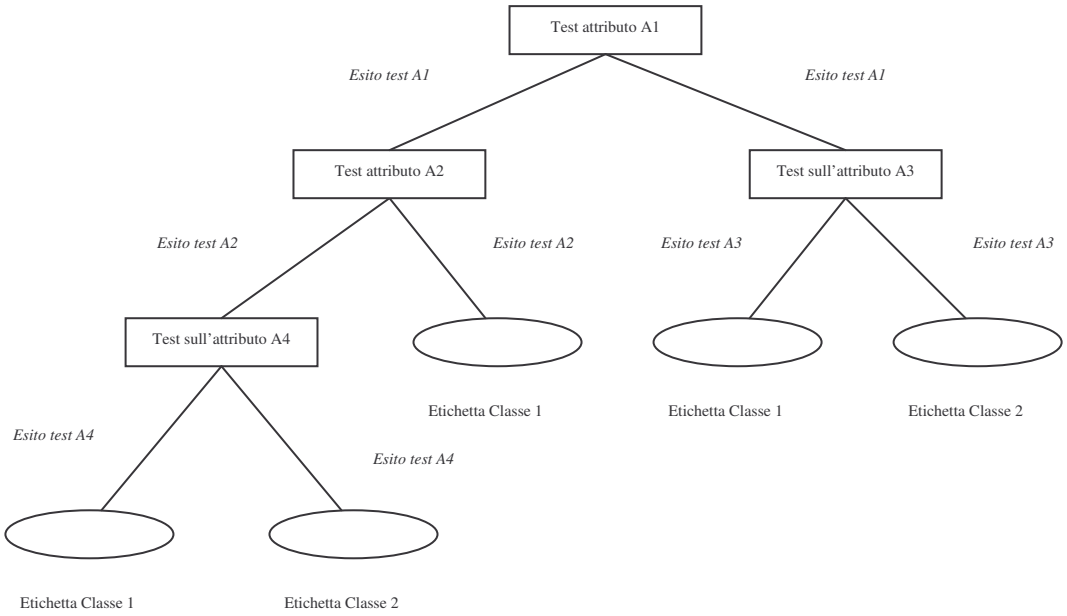


Figura 42: Albero per la classificazione in 2 classi.

IL GUADAGNO DI INFORMAZIONE

Per scegliere l'attributo più adatto ad effettuare il partizionamento a livello di ogni nodo si ha bisogno di una misura della bontà della suddivisione. Questa misura è basata sulla riduzione dell'entropia di ogni attributo per minimizzare la casualità o l'impurità di ogni partizionamento. Si cerca quindi l'attributo che richieda il minor numero di test da eseguire per classificare l'oggetto.

L'approccio teorico è il seguente: sia S un insieme di s oggetti campione, supponiamo che l'etichetta di classe abbia m valori distinti cosicché le classi siano $C_i (i = 1...m)$. Sia s_i il numero di campioni di S appartenenti alla classe C_i, l'informazione necessaria per classificare un dato oggetto è: $I(s_1, s_2, ..., s_m) = -\sum_{i=1}^{m} p_i \log_2(p_i)$, dove $p_i = s_i / s$ è la probabilità che un oggetto arbitrario appartenga a C_i. Si è usato il logaritmo in base 2 in quanto l'informazione è codificata in bit.

Supponiamo ancora che l'attributo A abbia v valori distinti $\{a_1, a_2, ..., a_v\}$. Tale attributo può essere usato per suddividere S in v sottoinsiemi $\{S_1, S_2, ..., S_v\}$, dove S_j contiene i campioni di S aventi per A il valore a_j. Qualora A fosse scelto come attributo di test, questi sottoinsiemi rappresentano i rami che partono dal nodo contenente S. Sia ora s_{ij} il numero di campioni della classe C_i racchiusi in S_j, l'*entropia* o l'informazione necessaria per il

partizionamento nei j sottoinsiemi è: $E(A) = \sum_{j=1}^{v} \frac{s_{1j} + s_{2j} + ... + s_{mj}}{s} I(s_1, s_2, ..., s_j)$. Il primo termine del prodotto rappresenta il peso relativo del j-esimo sottoinsieme, ovvero il numero di campioni in esso contenuti rispetto ai campioni totali. Minore è il valore dell'entropia maggiore è la purezza del partizionamento in base all'attributo. Infatti il guadagno è così calcolato: $G(A) = I(s_1, s_2, ..., s_m) - E(A)$.

3.6.1 Potatura dell'albero

Quando un albero viene costruito può riflettere le anomalie nei dati nell'insieme di apprendimento dovute a rumore o outliers. I metodi di potatura degli alberi intendono evitare o almeno attenuare il fenomeno del sovradattamento. Tali metodi usano misure statistiche per rimuovere i rami meno affidabili e migliorare la classificazione dei dati di test sia in velocità che in efficacia.

Gli approcci principali sono due:

PREPOTATURA

Questo approccio, più che potare l'albero ne arresta la crescita. Infatti si decide di non partizionare ulteriormente un nodo; quindi tale nodo diviene foglia e viene etichettato con la classe più frequente nel suo sottoinsieme o con la distribuzione di probabilità degli elementi. Per operare questa decisione vengono usate delle misure statistiche fra cui anche il guadagno di informazione. Se partizionando l'insieme i nodi risultanti cadono sotto delle soglie prefissate, i partizionamenti successivi sono inibiti. Le difficoltà si presentano nella scelta delle soglie adatte; valori troppo alti produrranno alberi sovrasemplificati, al contrario, soglie troppo basse innalzeranno la complessità dell'albero e di conseguenza il suo costo computazionale e la codifica.

POSTPOTATURA

Secondo questa metodologia si effettua la potatura vera e propria. Infatti i nodi sono rimossi da alberi completamente sviluppati tagliando i relativi rami.

La strategia più usata è l'algoritmo *cost complexity*. Il nodo potato diventa foglia ed assume l'etichetta più frequente fra i campioni. Per ogni nodo non foglia l'algoritmo calcola il tasso di errore atteso nel caso il sotto albero sia potato. In seguito, viene calcolato l'errore atteso in presenza del sottoalbero prendendo in considerazione una combinazione pesata delle osservazioni lungo ogni ramo del nodo. Se la potatura porta ad un tasso di errore maggiore, il sottoalbero viene mantenuto, altrimenti è potato.

Ogni volta che è operata una scelta di questo tipo (potare / mantenere) viene stimata l'accuratezza di ogni albero su test set indipendenti e viene scelto l'albero che massimizza tale parametro.

La decisione di potatura, invece che essere basata sugli errori attesi, può effettuarsi sul costo di codifica della struttura ovvero sul numero di bit necessari a codificarlo. Tale metodo adotta il principio MDL (Minimum Description Length), infatti è scelto l'albero dalla codifica più semplice.

ESTRAZIONE DELLE REGOLE DI CLASSIFICAZIONE

La conoscenza contenuta negli alberi decisionali può essere rappresentata da regole di classificazione nella forma di espressioni IF – THEN. Questa forma di rappresentazione risulta comprensibile alla maggior parte degli utenti. Una regola è creata per ogni percorso dalla radice ad una foglia dell'albero. Ogni valore dell'attributo di test rappresenta una congiunzione nella parte IF della regola. Ogni nodo foglia mantiene il valore della classe predetta e forma la parte THEN della regola.

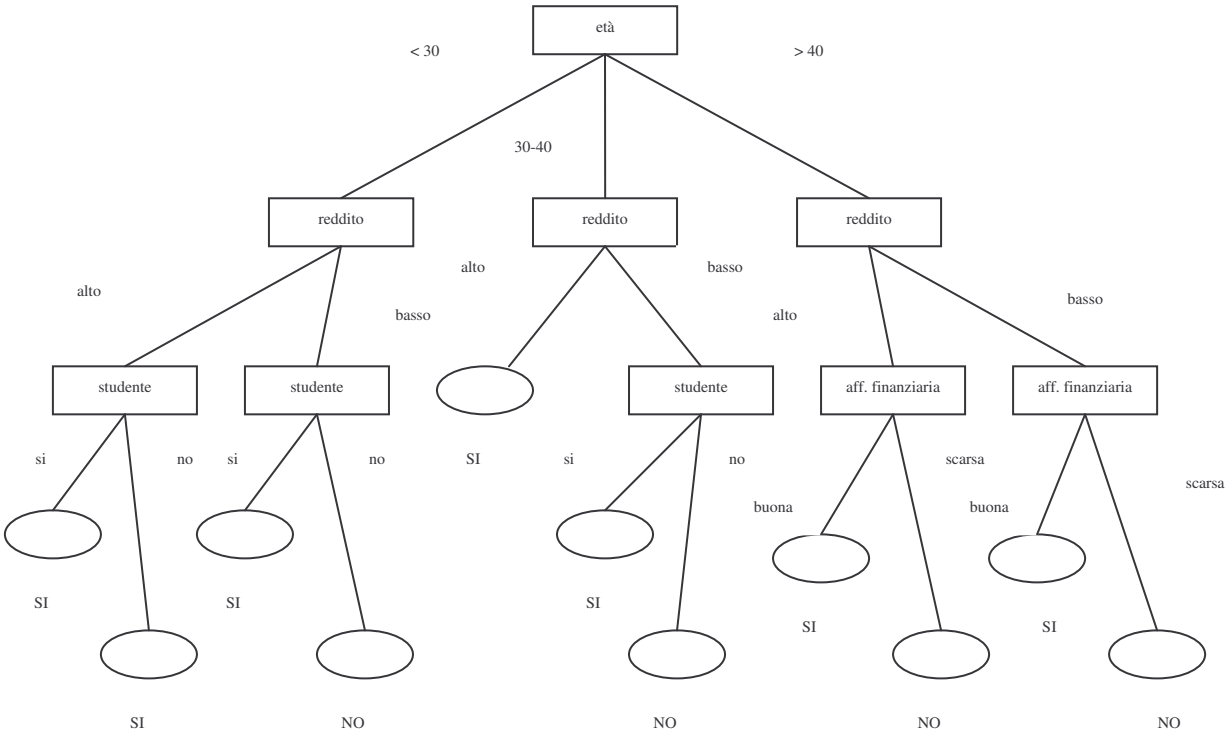


Figura 43: Albero per la classificazione di profili.

se ad esempio si è ottenuto un albero come in fig. 43 inerentemente ad un problema di profiling dei consumatori, ovvero la determinazione dei profili degli acquirenti di computer in questo caso, alcune delle regole estraibili dalla struttura sono le seguenti:

```
IF età "< 30" AND reddito = "alto" AND studente = si THEN acquista_pc = si
```

```
IF età "30-40" AND reddito = "alto" THEN acquista_pc = si
IF età "> 40" AND reddito = "basso" AND aff_fin = "scarsa" THEN acquista_pc
= no
```

3.6.2 Miglioramenti dell'algoritmo base

L'algoritmo descritto nel par. 3.6 richiede che tutti gli attributi siano categorici o discretizzati. Si possono apportare modifiche per permettere la gestione di attributi continui. Sia A un attributo continuo, un test su A produrrebbe due rami rispettivamente per A < V e A > V. Se si hanno a disposizione a priori v valori di A sono possibili $v-1$ suddivisioni. Tipicamente per effettuare i test di separazione vengono utilizzati i valori medi fra valori adiacenti.

L'algoritmo di base produce un ramo per ogni valore distinto dell'attributo di test, ciò porta alla formazione di partizioni sempre più piccole che, da un certo punto in poi, potrebbero essere statisticamente insignificanti per insufficienza di dati. Una possibile soluzione potrebbe essere il raggruppamento di valori categorici sotto un unico nome. Un'alternativa sarebbe l'esecuzione di soli test binari sui dati, in tal caso ogni ramo rappresenta il valore di una variabile booleana e di conseguenza ogni nodo ha al massimo due rami. Infatti, secondo studi empirici effettuati, gli alberi binari portano ad una minore frammentazione dei dati e sono più accurati degli alberi tradizionali.

Sono state proposte varie alternative anche per quanto riguarda le misure per la selezione degli attributi di test in luogo del guadagno di informazione: indice gini, tabelle di contingenza statistica, G-statistic ecc.

Altri metodi sono stati introdotti per al gestione di valori mancanti. Un valore mancante o sconosciuto per un attributo A può essere rimpiazzato dal valore più comune per l'attributo, per esempio. Diversamente, il guadagno di informazione dell'attributo può essere ridotto in base alla proporzione dei suoi valori mancanti. Altri metodi possono guardare al valore più probabile o fare uso delle relazioni note fra A e gli altri attributi.

Sono state testati anche versioni incrementali degli alberi nel senso che, avendo a disposizione nuovi insiemi di apprendimento quando l'albero è gia stato costruito, essi sono in grado di "ristrutturarsi" in base alle nuove informazioni piuttosto che sviluppare un altro albero ex novo.

Molti di questi miglioramenti sono incorporati in C4.5, l'algoritmo successore di ID3.

3.6.3 Algoritmi scalabili

La maggior parte degli algoritmi per l'induzione di alberi, fra cui C4.5 e ID3, ha la notevole restrizione di far risiedere in memoria centrale l'intero insieme di apprendimento. Quando tale

insieme consiste in milioni di tuple (dimensione comune nei grossi DW reali), le prestazioni sono drammaticamente inficiate dalla pesante attività di swapping da memoria centrale a memoria di massa.

Un primo semplice metodo può essere quello di partizionare l'insieme di apprendimento in n sottoinsiemi tali da risiedere in memoria centrale e sviluppare n alberi differenti, uno per ogni partizione. Poi si procede all'integrazione degli alberi nell'albero finale, ma l'accuratezza di classificazione dell'albero finale è sensibilmente ridotta rispetto ai classificatori di partenza.

SLIQ e SPRINT sono due versioni migliorate verso la scalabilità, che permettono di gestire sia attributi categorici che continui. Essi fanno uso di particolari strutture dati per facilitare la costruzione dell'albero e richiedono un iniziale preordinamento dei dati residenti su memoria di massa. SLIQ impiega una lista attributi residente su disco e una lista classi contenuta in memoria centrale. La lista attributi è del tipo:

Affidabilità_Fin	rid
Buona	1
Buona	2
Scarsa	3
Buona	4
…	…

Età	rid
25	2
38	4
44	1
31	3
…	…

Tabella 3: Lista attributi gestita da SLIQ.

Ogni attributo è associato ad un identificatore del record. La lista classi è del tipo:

rid	Acquista_pc	nodo foglia
1	Si	5
2	Si	2
3	No	1
4	No	3
…	…	…

Tabella 4: Lista classi gestita da SLIQ.

Dove sono contenuti il valore della classe, l'identificatore e il nodo foglia al quale la classe è associata. Ogni tupla dell'insieme è rappresentata dal collegamento delle entrate delle liste attributi alla lista classi seguendo l'identificatore di record, cosicché ogni tupla sarà collegata al nodo foglia che la contiene nella classificazione.

La lista classi è mantenuta in memoria centrale per via dei frequenti accessi a modiche durante le fasi di costruzione e sviluppo dell'albero. Se la lista non può essere contenuta interamente in memoria, le prestazioni decadono vertiginosamente.

SPRINT usa delle strutture che mantengono tutte l'informazione sulla classe di appartenenza, oltre all'identificatore di record.

Affidabilità_Fin	Acquista_pc	Rid
Buona	Si	1
Buona	Si	2
Scarsa	No	3
Buona	No	4
…		…

Età	Acquista_pc	rid
25	Si	2
38	No	4
44	Si	1
31	No	3
…		…

Tabella 5: Struttura gestita da SPRINT. In ogni lista attributi è presente l'informazione sulla classe.

Quando un nodo è diviso i record delle tabelle sono partizionati di conseguenza ma l'ordinamento è mantenuto. SPRINT è stato progettato per una facile parallelizzazione, ulteriore contributo alla scalabilità. Esso rimuove tutte le restrizioni di memoria ma richiede la gestione di un albero hash in memoria centrale proporzionale alla dimensione del training set.

3.7 Classificatori Bayesiani

I classificatori bayesiani sono classificatori statistici. Essi possono predire l'appartenenza alla classe, come la probabilità che un dato oggetto faccia parte di una determinata classe.
Questo tipo di classificazione ha il suo fondamento teorico sul *teorema di Bayes*. La tecnica più nota in questo settore è il classificatore di Bayes semplice (naive bayesian classifier). Studi di comparazione di algoritmi di classificazione hanno verificato che tale classificatore è paragonabile in efficienza agli alberi decisionali e alle reti neurali. Data la sua semplicità, il processo ha esibito velocità ed accuratezza anche su grandi volumi di dati. Esso si basa sull'assunzione cosiddetta di indipendenza condizionale di classe che asserisce che l'effetto del valore di un attributo su una certa classe è indipendente dal valore degli altri attributi. Ciò semplifica, come vedremo, la computazione di talune misure di probabilità ed in questo senso il classificatore è da ritenersi "semplice".

3.7.1 Teorema di Bayes

Sia X un insieme di oggetti di cui sia sconosciuta l'etichetta di classe; sia H un insieme di ipotesi sull'appartenenza di un certo oggetto in X ad una certa classe C. A scopi di classificazione, si ha bisogno di calcolare $P(H \mid X)$, ovvero la probabilità che l'insieme osservato ,X, verifichi le ipotesi H. $P(H \mid X)$ è la probabilità condizionata di H dato X, detta probabilità *a posteriori*. Ad esempio si supponga che l'insieme campione consista in frutti, descritti da due attributi: colore e forma. Si supponga ancora che X sia rosso e rotondo e H è l'ipotesi che X sia una mela, allora $P(H \mid X)$ è la probabilità che X sia una mela sapendo che

X è rosso e rotondo. In contrasto con la probabilità a posteriori, è la probabilità *a priori* $P(H)$, che è la probabilità che X sia una mela non avendo nessuna informazione aggiuntiva. Analogamente $P(X \mid H)$ è la probabilità che X sia rosso e rotondo sapendo che X è una mela. Il teorema di Bayes fornisce un modo per calcolare $P(H \mid X)$ dalla $P(X \mid H)$ e dalle probabilità a priori secondo la seguente formula: $P(H \mid X) = \frac{P(X \mid H)P(H)}{P(X)}$.

3.7.2 Classificatore Bayesiano semplice (Naive Bayesian classifier)

Il classificatore lavora come segue:

1) Ogni campione dati è rappresentato da un vettore di features $X = (x_1, x_2, ..., x_n)$ contenente n misure su n attributi dell'oggetto $A_1, A_2, ..., A_n$.
2) Si supponga che vi siano m classi $C_1, C_2, ..., C_m$. Dato un oggetto campione X sconosciuto (ovvero privo di etichetta di classe), il classificatore assegnerà X alla classe avente la probabilità a posteriori più alta, condizionata su X. Cioè l'oggetto X sarà associato alla classe C_i se e solo se: $P(C_i \mid X) > P(C_j \mid X)$ per $j = 1...m, j \neq i$. La classe C_i per cui la $P(C_i \mid X)$ è massimizzata è detta ipotesi a posteriori massima. Secondo il teorema di Bayes essa risulta: $P(C_i \mid X) = \frac{P(X \mid C_i)P(C_i)}{P(X)}$.
3) Siccome $P(X)$ è un temine costante per tutte le classi, si deve massimizzare il solo numeratore $P(X \mid C_i)P(C_i)$. Se non si conosce la probabilità di ogni singola classe, esse possono essere ritenute equamente probabili, ovvero $P(C_1) = P(C_2) = ... = P(C_m)$, e si procede a massimizzare solo $P(X \mid C_i)$. Se si hanno informazioni in proposito, la probabilità a priori di ogni classe può essere stimata come $P(C_i) = s_i / s$, dove s_i è il numero di campioni nell'insieme di apprendimento aventi etichetta C_i, e s è il numero di campioni totale.
4) Dati insiemi con molti attributi può essere molto costoso calcolare $P(X \mid C_i)$. A questo punto viene in soccorso l'assunzione di indipendenza condizionale di classe. Questa assume che ogni valore di un attributo di un oggetto avente una certa etichetta di classe sia condizionalmente indipendente da ogni altro valore, ovvero che non vi sia nessuna relazione di dipendenza fra gli attributi. Ciò si traduce nella:

$P(X \mid C_i) = \prod_{k=1}^{n} P(x_k \mid C_i)$, dove le probabilità $P(x_1 \mid C_i), P(x_2 \mid C_i), ..., P(x_n \mid C_i)$ possono essere calcolate dall'insieme di apprendimento nei modi seguenti:

a) Se A_k è categorico, $P(x_k \mid C_i) = s_{i,k} / s_i$ dove $s_{i,k}$ è il numero di oggetti con etichetta C_i aventi x_k come valore di A_k, ed s_i il numero di oggetti totali in C_i.

b) Se A_k è continuo, si assume per esso una distribuzione gaussiana con media μ_{C_i} e deviazione standard σ_{C_i}:

$$P(x_k, C_i) = g(x_k, \mu_{C_i}, \sigma_{C_i}) = \frac{1}{\sqrt{2\pi}\sigma_{C_i}} e^{-\frac{(x-\mu_{C_i})^2}{2\sigma_{C_i}^2}}.$$

5) Per classificare ogni oggetto non osservato, $P(X \mid C_i)P(C_i)$ è calcolato per ogni classe. Un oggetto X è assegnato alla classe C_i se e solo se: $P(X \mid C_i)P(C_i) > P(X \mid C_j)P(C_j)$ per $j = 1...m, j \neq i$. In altre parole è assegnato alla classe che rende massima la probabilità condizionata espressa dal teorema di Bayes.

In teoria, il classificatore bayesiano semplice ha il tasso di errore minimo rispetto a tutti gli altri classificatori. In pratica ciò non è vero a causa delle approssimazioni implicite nell'assunzione dell'indipendenza condizionale di classe o per la mancanza di informazioni sulla probabilità dei dati. Comunque, studi empirici hanno provato che il classificatore può essere paragonato, come prestazioni generali, agli alberi decisionali o alle reti neurali in alcuni domini applicativi.

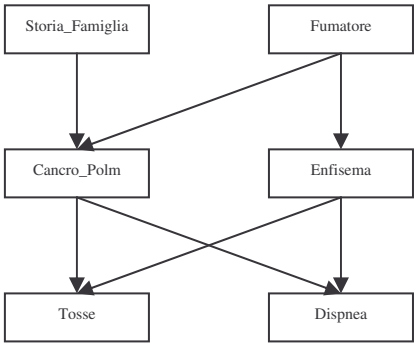
Il classificatore bayesiano semplice fonda la sua robustezza sulla giustificazione teorica del teorema di Bayes rispetto ad altri classificatori che non usano esplicitamente il teorema. Infatti, sotto certe ipotesi, alcune reti neurali producono l'ipotesi a posteriori massima, proprio come il classificatore bayesiano.

3.7.3 Reti Bayesiane

L'assunzione di indipendenza condizionale di classe, sulla quale fa affidamento il classificatore semplice, semplifica la computazione, ma può non rispecchiare la reale natura dei dati di apprendimento. In pratica, le dipendenze fra attributi possono esistere ed anche in maniera marcata. Le reti bayesiane lavorano con distribuzioni di probabilità congiunte e permettono la specifica di dipendenze condizionali fra sottogruppi di variabili. Esse

forniscono un modello grafico di relazioni causali sul quale effettuare l'apprendimento della rete.

Una rete bayesiana consiste di due elementi principali. La prima è un grafo aciclico, dove ogni nodo rappresenta una variabile ed ogni arco la dipendenza probabilistica. Secondo le usuali notazioni dei grafi è instaurata quindi una gerarchia genitori – discendenti, come in fig. 44, dove si è riportato un esempio su dati medici.



	S_F, F	S_F, ~F	~S_F, F	~S_F, ~F
C_P	0.8	0.5	0.7	0.1
~C_P	0.2	0.5	0.3	0.9

Figura 44: Grafo delle dipendenze e tabella di probabilità condizionale.

Ogni variabile è dipendente dai suoi genitori. Ogni variabile è indipendente dai suoi non-discendenti, dati i suoi genitori, nel senso che essa è indipendente sia dai suoi non - discendenti che dai suoi non-genitori. Ogni variabile può essere continua o discreta, e rappresenta un particolare attributo dell'oggetto o un valore nascosto, riferendosi a valori sconosciuti o mancanti.

Il secondo elemento di una rete bayesiana consiste in una tabella di probabilità condizionale (conditional probability table, CPT) per ogni variabile coinvolta. Una CPT per una variabile Z specifica le probabilità condizionali $P(Z \mid G(Z))$, dove $G(Z)$ sono i genitori di Z. In fig. 44 è riportata una la CPT per la variabile C_P (Cancro_Polmone) sulla base delle possibili combinazioni delle variabili genitore. Ad esempio, la cella in alto a sinistra segnala che: $P(Cancro\_Polmone = si \mid Storia\_Famiglia = si, Fumatore = si) = 0.8$

La probabilità congiunta di ogni tupla avente i valori $(z_1,...,z_n)$ per gli attributi $Z_1...Z_n$ è calcolata come: $P(z_1,...,z_n) = \prod_{i=1}^{n} P(z_i \mid G(Z_i))$.

Ogni nodo della rete può essere considerato come nodo di output e rappresentare un'etichetta di classe. Vi possono essere più nodi di output. Il processo di classificazione, piuttosto che restituire una singola etichetta di classe, ritornerà una distribuzione di probabilità delle etichette per l'oggetto in questione.

Per l'apprendimento della rete sono possibili diversi scenari. Se lo schema della rete è disponibile in anticipo, l'apprendimento consiste solo nel calcolo dei valori nelle celle delle CPT. Un processo simile al calcolo delle probabilità nel classificatore bayesiano semplice.
Se la struttura della rete è nota ma vi sono attributi nascosti, nel senso di mancanti, un metodo a gradiente discendente può essere usato per istruire la rete. L'obiettivo è sempre di calcolare i valori della CPT.
Sia S un insieme di s campioni di apprendimento $(X_1, X_2, ..., X_s)$. Sia w_{ijk} il valore della CPT per l'attributo $Y_i = y_{ij}$ con genitori $U_i = u_{ik}$. Ad esempio se w_{ijk} è la cella in alto a sinistra in fig. 44 allora $Y_i = y_{ij}$ è C_P = si, mentre $U_i = u_{ik}$ è la lista {S_F, F}={si, si}. I pesi w_{ijk} sono inizializzati a valori casuali, la strategia a gradiente discendente effettua un algoritmo avido (greedy) hill-climbing, ad ogni iterazione i valori w_{ijk} sono aggiornati e vi sarà una convergenza eventuale verso un ottimo locale.
Il metodo mira a massimizzare $P(S \mid H)$, ma lo fa su $\ln P(S \mid H)$ perché più semplice e procede come segue:

1. calcola il gradiente per ogni w_{ijk} come: $\frac{\partial \ln P(S \mid H)}{\partial w_{ijk}} = \sum_{d=1}^{s} \frac{P(Y_i = y_{ij}, U_i = u_{ik} \mid X_d)}{w_{ijk}}$.

   La probabilità a secondo membro è calcolata per ogni campione X_d in S. Quando i valori per gli attributi Y_i e U_i sono nascosti, vengono usati algoritmi inferenziali per reti bayesiane (come nel package Hugin).
2. i pesi sono aggiornati a piccoli passi nella direzione del gradiente: $w_{ijk} \leftarrow w_{ijk} + l \frac{\partial \ln P(S \mid H)}{\partial w_{ijk}}$, dove l è una costante detta passo di apprendimento.
3. normalizzazione dei pesi w_{ijk} nell'intervallo [0,1].

3.8 Reti neurali

Un'altra tecnica che può essere implementata a scopi di classificazione riguarda le reti neurali. Una rete neurale è un insieme di unità di input / output connesse fra loro, dove ogni connessione ha un peso associato. Durante la fase di apprendimento la rete regola i pesi per classificare correttamente gli oggetti. I tempi di apprendimento di una rete neurale sono abbastanza lunghi. Esse richiedono un set di parametri che sono determinati empiricamente, come la struttura e la topologia della rete. Le reti neurali sono state criticate per la loro scarsa

interpretabilità, per cui è umanamente difficoltoso estrarre il significato simbolico che portano con se, in special modo se le reti sono complesse. I loro vantaggi stanno nella buona gestione del rumore nei dati e nell'abilità di produrre modelli non riflessi dai training set sui quali sono state istruite.

3.8.1 Struttura della rete

Vi sono 3 tipi di unità, dette anche neuroni, che si possono interconnettere in una rete: le unità di input, che corrispondono agli attributi misurati per ogni oggetto; questi input sono alimentati simultaneamente e formano il cosiddetto livello di input (*input layer*). Gli output pesati di questo livello sono passati simultaneamente al livello successivo, detto livello nascosto (*hidden layer*). Le uscite di un livello nascosto possono alimentare un altro livello nascosto, ma nelle applicazioni reali, di norma, è usato un solo hidden layer. Le uscite pesate dell'ultimo livello nascosto fanno da ingresso al livello di output che emette la predizione. In fig. 45 si riporta uno schema generico di rete multilivello alimentata in avanti (multilayer feed-forward net).

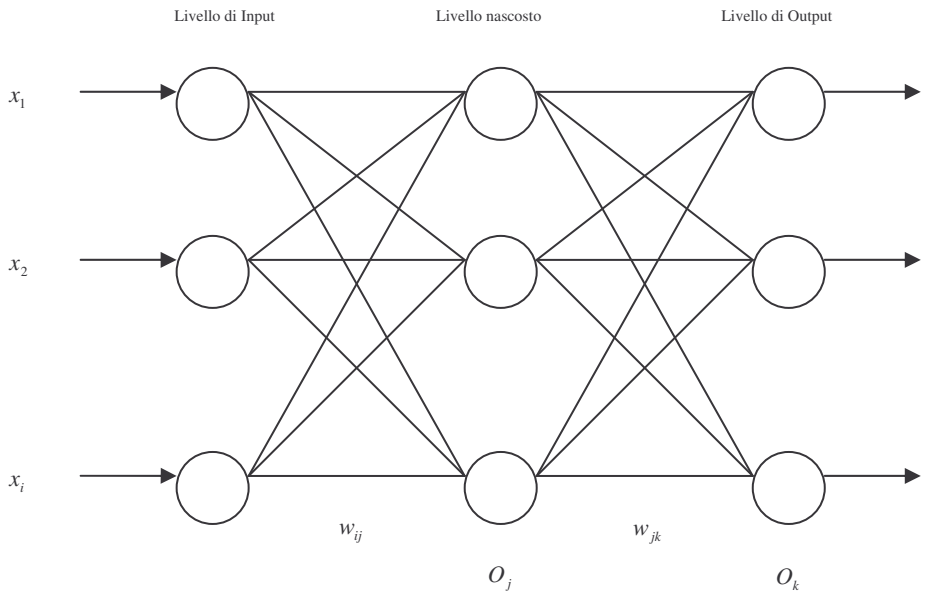


Figura 45: Struttura di una rete multilivello.

La rete in riportata è una rete a due livelli di output (*two layer net*); una rete con 2 livelli nascosti è detta rete a 3 livelli. La rete è alimentata in avanti nel senso che nessun peso ritorna indietro ad un unità dei livelli precedenti. Essa è pienamente connessa, infatto ogni nodo è connesso a tutti i nodi del livello successivo. Reti con funzioni soglia lineari e con un numero sufficiente di unità possono approssimare qualsiasi funzione.

3.8.2 Topologia della rete

Prima di iniziare l'apprendimento, si deve decidere sul numero di neuroni nel livello di input, sul numero di livelli nascosti e di unità nei livelli nascosti, sul numero di unità nel livello di uscita.

La normalizzazione in [0,1] dei valori di input aiuterà sensibilmente la fase di apprendimento, che risulterà più veloce. I valori discreti possono essere codificati in modo che vi sia un unità di input per ogni valore del dominio. Ad esempio, se l'insieme dei valori di un attributo A è $\{a_0, a_1, a_2\}$, si possono utilizzare 3 neuroni di ingresso I_0, I_1, I_2 in modo tale che se $A = a_0$ allora $I_0 = 1$ e gli altri sono settati a 0; se $A = a_1$, $I_1 = 1$ e gli altri a 0, e così via. Una unità di uscita può rappresentare due classi a seconda dei valori 0 / 1, se vi sono più classi, si usa un'unità per ogni classe.

Non vi sono regole generali per decidere sul numero dei livelli nascosti, ma si procede per tentativi misurando di volta in volta l'accuratezza della rete. Anche il settaggio iniziale dei valori e dei pesi della rete può influenzare l'accuratezza. Se l'accuratezza non è considerata accettabile si effettuano nuove fasi di apprendimento ricalibrando i valori iniziali e/o modificando la topologia della struttura.

3.8.3 Algoritmo di propagazione all'indietro (backpropagation)

La propagazione all'indietro è la procedura per effettuare l'apprendimento su una rete neurale miltilivello con alimentazione in avanti. Essa acquisisce informazioni iterativamente dai campioni del training set e modifica i pesi al fine di minimizzare l'errore quadratico medio fra il valore della predizione e il valore effettivo della classe dell'oggetto. Queste modifiche sono fatte all'indietro, a partire dal livello di uscita, attraverso i livelli nascosti, fino al livello di input. In generale i pesi eventualmente convergono ad un valore ottimo o subottimo, sebbene questo non sia garantito, ed il processo termina. I passi dell'algoritmo sono i seguenti:

1. **inizializzazione dei pesi** (e dei valori di bias) con valori casuali compresi negli intervalli [-1,1] o [-0.5,0.5]

ogni campione X è processato nel modo seguente:

2. **propagazione in avanti degli ingressi**. Sono calcolati per ogni unità i valori di input ed output di ogni livello. L'output di ogni neurone è calcolato come combinazione lineare dei suoi ingressi, dove i coefficienti della combinazione sono i pesi delle connessioni. Data un'unità j di un livello nascosto o di output, il suo ingresso è: $I_j = \sum_i w_{ij} O_i + \vartheta_j$, dove O_i è l'uscita dei nodi connessi in ingresso e ϑ_j è il valore di bias del nodo j. Il bias serve a calibrare l'attività del nodo. Ogni unità, quindi, prende

il valore di input e procede alla sua attivazione, ovvero al calcolo dell'output tramite la funzione di attivazione. Come funzione di attivazione è usata la funzione logistica: $O_j = \frac{1}{1+e^{-I_j}}$. Essa mappa un range di valori ampio, scaturente dalla combinazione lineare, in un intervallo [0,1]. La funzione logistica è non lineare e differenziabile.

3. **propagazione all'indietro dell'errore**. Ciò viene fatto aggiornando i pesi ed i bias per riflettere l'errore di predizione della rete. Per ogni unità j nel livello di uscita, l'errore è calcolato come: $Err_j = O_j(1-O_j)(T_j - O_j)$, dove T_j è l'uscita vera, basata sulla conoscenza dell'etichetta di classe del campione.

   L'errore per ogni unita j nei livelli nascosti è computata come: $Err_j = O_j(1-O_j)\sum_k Err_k w_{jk}$, dove Err_k è l'errore della k-esima unità connessa al livello successivo e w_{jk} è il peso relativo alla connessione. A questo punto i pesi vengono aggiornati calcolando lo scostamento ad ogni iterazione $\Delta w_{ij} = l \cdot Err_j O_i$, con l passo di apprendimento compreso in [0,1]. Siccome l'algoritmo adotta il metodo del gradiente discendente per trovare la combinazione ottima dei pesi, il passo di apprendimento serve a non bloccare l'algoritmo presso un ottimo locale, ma procedere la ricerca verso l'ottimo globale. Se l è troppo piccolo, la fase di apprendimento sarà lenta; se è troppo grande vi saranno grandi oscillazioni fra soluzioni successive. Una regola empirica è porre $l = 1/t$, con t = numero di iterazioni sul training set. L'aggiornamento vero e proprio avviene secondo la $w_{ij} = w_{ij} + \Delta w_{ij}$.

   I valori di bias sono aggiornati calcolando $\Delta \vartheta_j = l \cdot Err_j$ e modificando il vecchio valore $\vartheta_j = \vartheta_j + \Delta \vartheta_j$.

   Si noti che l'aggiornamento dei valori dei pesi e dei bias avviene ad ogni passo dell'algoritmo, cioè al presentarsi di ogni esempio. In questo caso ci si riferisce al case updating. Alternativamente i diversi incrementi possono essere accumulati in varibili differenti ed aggiornare i valori solo alla fine del processo iterativo. In quest'ultimo caso ci si riferisce all'epoch updating, dove ogni epoca è rappresentata da un passo dell'algoritmo.

L'algoritmo si arresta nei seguenti casi:

- tutti gli scostamenti Δw_{ij} relativi all'epoca precedente sono sotto una soglia specificata, oppure

- la percentuale di campioni non correttamente classificati all'epoca precedente è inferiore ad un certo limite, oppure
- è stato raggiunto un certo numero di iterazioni.

In pratica è frequente che siano necessarie epoche nell'ordine di 10^5 finchè i valori convergano.

Si riporta di seguito, per completezza, lo pseudocodice dell'algortimo di backpropagation.

```
Procedure BACKPROPAGATION
Input(campioni, l, rete)
Output(rete istruita)

Inizializza pesi e bias;
WHILE cond_terminazione = false {
      FOR EACH campione X in campioni {
            FOR EACH unità j di input o nascosta
                  //propagazione in avanti input
                  I_j = Σ_i w_ij O_i + ϑ_j // calcola input unità j
            FOR EACH unità j di input o nascosta
                  O_j = 1 / (1 + e^(-I_j)) // calcola output unità j
                  //propagazione all'indietro errore
            FOR EACH unità j di uscita
                  Err_j = O_j (1 - O_j)(T_j - O_j) //calcolo errore unità j
            FOR EACH unità j nascosta
                  Err_j = O_j (1 - O_j) Σ_k Err_k w_jk //calcolo errore unita j
            FOR EACH peso nella rete      {
                  Δw_ij = l · Err_j O_i // incremento
                  w_ij = w_ij + Δw_ij } // aggiornamento
            FOR EACH bias nella rete      {
                  Δϑ_j = l · Err_j // incremento
                  ϑ_j = ϑ_j + Δϑ_j } // aggiornamento
      }
}
```

3.8.4 Interpretabilità delle reti

Lo svantaggio più evidente delle reti neurali sta nella loro scarsa interpretabilità, infatti la conoscenza acquisita durante l'apprendimento è umanamente difficile da rappresentare. Questo problema ha motivato molte attività di ricerca per estrarre la conoscenza insita nelle reti istruite e rappresentarla simbolicamente.

Come si è già accennato reti con centinaia di neuroni completamente connessi sono difficili da articolare. Un primo passo verso la loro semplificazione è quello di effettuare dei tagli (pruning) alla struttura, ovvero di eliminare i collegamenti che non decrementano durante il

processo. Una volta che la rete è stata potata, un metodo per dedurre la conoscenza può essere quello di analizzare i valori di attivazione comuni dei neuroni negli strati nascosti. Una volta determinati tali valori, ad esempio con il clustering, vengono messi in relazione con le combinazioni dei valori dei nodi di output e si tenta di generare delle regole di associazione. Similmente si cercano le relazioni notevoli fra i neuroni nascosti e i neuroni di input, formando altre regole. Infine i due insiemi di regole ottenuti vengono combinati per formare delle regole globali nella forma IF-THEN.

Un esempio di questo metodo inferenziale applicato alla rete raffigurata è riportato in fig. 46.

Identify sets of common activation values for each hidden node, H_i:
for H_1: (-1,0,1)
for H_2: (0,1)
for H_3: (-1, 0.24, 1)

Derive rules relating common activation values with output nodes, O_j:
IF (a_2 = 0 AND a_3 = -1) OR
(a_1 = -1 AND a_2 = 1 AND a_3 = -1) OR
(a_1 = -1 AND a_2 = 0 AND a_3 = 0.24)
THEN O_1 = 1, O_2 = 0
ELSE O_1 = 0, O_2 = 1

Derive rules relating input nodes, I_i, to output nodes, O_j:
IF (I_2 = 0 AND I_7 = 0) THEN a_2 = 0
IF (I_4 = 1 AND I_6 = 1) THEN a_3 = -1
IF (I_5 = 0) THEN a_3 = -1
...

Obtain rules relating inputs and output classes:
IF (I_2 = 0 AND I_7 = 0 AND I_4 = 1 AND I_6 = 1) THEN class = 1
IF (I_2 = 0 AND I_7 = 0 AND I_5 = 0) THEN class = 1

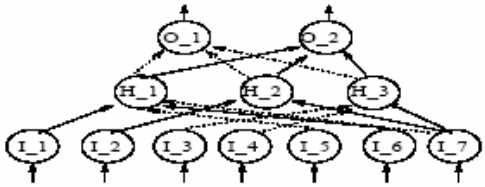


Figura 46: Esempio di generazione di regole di associazione da una rete neurale.

3.9 Regressione

La predizione di un attributo numerico continuo può essere modellata statisticamente con la tecnica della regressione. Ad esempio, si vuole sviluppare un modello che predica il salario medio dei laureati con almeno 10 anni di esperienza; oppure che stimi le vendite sul mercato di un certo prodotto dato il suo prezzo. Molti questioni possono essere affrontate con la regressione lineare, ma possono essere apportate trasformazioni alle variabili per ricondurre problemi non lineari a lineari. Per ovvie ragioni non si può dare una trattazione esauriente delle tecniche di regressione che convolgerebbero la trattazione di complessi modelli matematici.

3.9.1 Regressione lineare

Nella regressione lineare i dati sono modellati su una retta nello spazio bidimensionale. Quella lineare è la forma più semplice di regressione. Una variabile casuale sconosciuta Y, detta *response variable*, viene modellata come funzione lineare di un'altra variabile casuale, ma nota, X, detta *predictor variable*, secondo la relazione: $Y = \alpha + \beta X$. La varianza della Y è

assunta costante e i coefficienti di regressione α e β definiscono il coefficiente angolare e l'intercetta sulle ordinate della retta. Tali coefficienti possono essere trovati con il metodo dei minimi quadrati. Dati s campioni $(x_1, y_1)...(x_s, y_s)$, i coefficienti possono essere stimati con le seguenti equazioni: $\beta = \frac{\sum_{i=1}^{s}(x_i - x_m)(y_i - y_m)}{\sum_{i=1}^{s}(x_i - x_m)^2}$; $\alpha = y_m - \beta x_m$, con x_m e y_m medie dei relativi valori.

3.9.2 Regressione multipla

È un'estensione della regressione lineare che coinvolge più di una predictor variable. Essa permette la modellazione di una variabile di responso su molteplici variabili predittrici, ovvero su vettori multidimensionali di features. La forma generale della relazione di regressione è la seguente: $Y = \alpha + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_n X_n$. Per la stima dei coefficienti è possibile applicare il metodo dei minimi quadrati.

3.9.3 Regressione non lineare

Questo tipo di regressione è usata per modellare dati che esibiscono un legame non lineare, ad esempio polinomiale. La regressione polinomiale può ottenersi dalla lineare aggiungendovi termini polinomiali. Questa relazione può poi essere trasformata in lineare ed essere risolta con il metodo dei minimi quadrati. Ad esempio una regressione cubica: $Y = \alpha + \beta_1 X + \beta_2 X^2 + \beta_3 X^3$, applicando le semplici trasformazioni $X_1 = X; X_2 = X^2; X_3 = X^3$, viene trasformata nella regressione lineare $Y = \alpha + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3$.

3.9.4 Altri modelli di regressione

La tecniche di regressione possono essere usate anche per modellare attruibuti categorici, in questo caso esse si chiamano modelli lineari generalizzati. In tali modelli, la verianza della response variable Y non è costante come nella regressione lineare, ma è funzione del suo valor medio. I metodi generalizzati usati più comunemente sono la regressione logistica e la regressoine di Poisson. La regressione logistica modella la probabilità di occorrenza di eventi come funzione lineare di un insieme di predictor variables.

I modelli log-lineari, invece, approssimano distribuzioni multidimensionali di probabilità su attributi categorici. In tali modelli gli attributi contunui devono essere prima discretizzati.

Tale metodo, ad esempio, può essere usato per stimare la probabilità di ogni singola cella di un cubo n-dimensionale analizzando i sottocubi di dimensione inferiore, ovvero (n-1), (n-2) ecc.

Cap. IV

L'APPROCCIO OLEDB for DM

Le tecniche di Data Mining basate sulla statistica e sull'apprendimento delle macchine possono accrescere significativamente la capacità di analizzare i dati. Nonostante sia questo l'obiettivo ultimo del DM, questa tecnologia è destinata a rimanere confinata in "nicchie applicative", a meno che non sia fatto uno sforzo per integrare questa tecnologia con i sistemi di basi dati tradizionali. Questo perché l'analisi dei dati ha bisogno di essere unificata e "fusa" nel warehouse di dati a scopi di integrità dei dati e capacità di gestione. Per tutto questo, una delle sfide principali è permettere l'integrazione delle tecnologie di DM all'interno del framework dei sistemi tradizionali senza "cuciture" visibili.
Nell'ultima decade sono stati fatti molti progressi verso il miglioramento degli algoritmi di mining: la scalatura delle tecniche più note verso il trattamento di grandi moli di dati, ad esempio rendendo gli algoritmi coscienti della presenza di memorie di massa (disk - aware), invece che assumere che tutti i dati risiedano in memoria centrale.
Un'altra direzione di ricerca che è stata inseguita è stata la possibilità di considerare gli algoritmi di mining come implementazione di applicazioni tradizionali per DBMS. Queste implementazioni assicurano che le applicazioni di tecniche mining siano non solo "disk - aware", ma anche "SQL - aware", ovvero trarre vantaggio dalle funzionalità offerte dal motore SQL e dalle API.
Sicuramente quest'ultimo è un grande contributo, ma non si deve perdere di vista l'obiettivo fondamentale del DM, ovvero permettere agli sviluppatori di applicazioni per database di costruire il proprio modello DM (ad es. un albero decisionale, un modello regressivo) a partire dal proprio database, usare tale modello a fini analitici o predittivi, permettere la condivisione del modello con altri sistemi o applicazioni adoperando una lingua franca (ad es. PMML).
Tutto ciò che si è detto è il presupposto del successo della tecnologia DM, obiettivo che Microsoft Corporation si è posta nello sviluppo dello standard OLEDB for DM.

Riconoscendo le questioni citate, il primo aspetto chiave che si presenta in questo tentativo di integrazione è la necessità di trattare un modello di mining come oggetto di classe nel database. Ricordando che un modello è ottenuto dall'applicazione di algoritmi di mining su insiemi di apprendimento, ciò può essere realizzato da applicazioni SQL che implementino

degli algoritmi di training. Il sistema, dal canto suo, è ancora ignaro della semantica e ancor di più della presenza del modello in quanto non esplicitamente rappresentato. Senza la possibilità di una rappresentazione esplicita del modello all'interno del DB non si potrà fare leva sulle funzionalità e sulle capacità del gestore del sistema per condividere, riusare e manipolare il modello. Ancora, non vi è modo di ricercare in base a delle proprietà un modello nell'insieme di quelli creati; non si può indicare un modello per la predizione di una certa colonna di un insieme dati sconosciuto e poi interrogarlo per ottenere il risultato; non si potranno comparare predizioni effettuate da modelli di diversa natura per valutarne l'accuratezza.
In modo da rappresentare effettivamente tali modelli nei database, si ha bisogno di catturare la *creazione* del modello tramite un algoritmo; l'*esplorazione* del modello e del suo contenuto; la *selezione* di un modello per scopi predittivi. **OLEDB for DM**, noto anche come **DMX to SQL** (*Data Mining Extension to SQL*) permette di catturare questi passi fondamentali in metafore SQL, questo permetterà allo sviluppatore di sfruttare tutte le funzionalità del sistema, evitando al tempo stesso di modificare il paradigma di sviluppo dell'applicazione.
Inoltre è necessaria la presenza di strutture di metadati che descrivano le proprietà di una colonna di predizione cosicché gli strumenti analitici possano interpretarla.

In questo capitolo si vuole focalizzare l'attenzione sul supporto fornito da Microsoft per lo sviluppo di applicazioni di Data Mining integrate in ambienti DBMS, esaminando lo standard OLEDB for DM introdotto dal Data Mining Group e sviluppato presso il Microsoft Research Center.

4.1 Il modello Microsoft di accesso ai dati: OLEDB

La soluzione ODBC (Open DataBase Connectivity) è stata introdotta da Microsoft nei primi anni 90 e costituisce il cuore dell'architettura di accesso ai dati della piattaforma della casa di Redmond. ODBC è una soluzione sviluppata con l'intento di consentire l'accesso a sorgenti dati relazionali in un contesto eterogeneo e distribuito. Tramite un interfaccia ODBC, le applicazioni possono accedere a dati presenti su sistemi relazionali generici, eventualmente situati su sistemi remoti connessi in rete. Il linguaggio supportato è un SQL particolarmente "ristretto", causa principale dei pro e contro della soluzione.
Nell'architettura ODBC, il collegamento fra l'applicazione e il server di dati avviene tramite un driver che viene collegato dinamicamente all'applicazione e da essa di volta in volta invocato.

ODBC è una struttura a 4 livelli, e sono:

- *Applicazione*: essa richiama le funzioni SQL per eseguire interrogazione ed acquisirne i risultati. L'applicazione è trasparente rispetto al protocollo di comunicazione, al server DBMS e al sistema operativo ove il DBMS è installato, tutti mascherati dal driver.
- *Driver Manager*: è responsabile del caricamento dei driver a seconda delle esigenza dell'applicazione.
- *Driver*: essi sono delegati all'esecuzione delle funzioni ODBC e, pertanto, sono in grado di eseguire interrogazioni SQL, traducendole opportunamente rispetto alla sintassi e alla semantica degli specifici prodotti cui viene fatto accesso. Essi sono anche responsabili della restituzione dei risultati.
- *Sorgente Dati*: è il sistema che esegue effettivamente le funzioni trasmesse dall'applicazione client.

Si è detto che questa soluzione utilizza SQL come linguaggio base per la comunicazione. La struttura di SQL limita fortemente i tipi di dati che possono essere acceduti, infatti esso si presta bene alla gestione di fonti dati relazionali e tabellari, ma non può essere utilizzato per l'accesso a DB multidimensionali (gerarchici) o a semplici file preformattati.

Per di più vi sono molte interpretazioni o "dialetti" di SQL utilizzati da driver diversi a seconda del costruttore, ciò può causare problemi nella formulazione di query alla base dati. Infatti i driver sono sviluppati dai diversi vendors al fine di supportare strutture dati proprietarie. Comunque, per venire incontro a tali inconvenienti ODBC propone diversi livelli di adesione allo standard: Core, Level 1, Level 2. Ogni driver aderisce ad uno specifico livello a seconda delle funzionalità che fornisce.

Per superare le evidenti limitazioni di ODBC fu successivamente introdotto OLEDB.

OLE DB (Object Linking and Embedding for Data Bases) rappresenta una componente importante della strategia Microsoft per l'accesso universale ai dati, in quanto consente l'accesso in modo efficiente da qualsiasi origine dati. Tramite OLE DB è possibile visualizzare dati tabellari di qualunque tipo, anche non provenienti da un database (caselle di posta elettronica, sistemi CAD/CAM), garantendo così una maggiore flessibilità.

OLE DB si basa sul concetto di consumer e provider. Il consumer effettua richieste di dati mentre il provider restituisce al consumer i dati in formato tabellare. Dal punto di vista della programmazione, l'implicazione principale di questo modello è rappresentata dal fatto che il provider deve implementare tutte le chiamate effettuate dal consumer.

In termini tecnici, un consumer è qualsiasi sistema o codice di applicazione (non necessariamente un componente OLE DB) che accede ai dati tramite interfacce OLE DB. Le

interfacce vengono implementate in un provider. Pertanto, per provider si intende qualsiasi componente software che implementa interfacce OLE DB per incorporare l'accesso ai dati ed esporre tali dati ad altri oggetti, ovvero ai consumer.
In termini di ruoli, quindi, un consumer richiama i metodi su interfacce OLE DB, mentre un provider OLE DB implementa le interfacce OLE DB necessarie.
In OLE DB non vengono utilizzati i termini "client" e "server", in quanto questi ruoli non sono sempre validi, in particolare in situazioni a più livelli. Dal momento che un consumer può essere un componente su un livello che viene utilizzato da un altro componente, la definizione di componente client sarebbe inesatta. Inoltre, i provider funzionano in alcuni casi più come driver di database che come server.
Per provider OLE DB si intende un gruppo di oggetti COM (il modello a oggetti per tutto il software Windows) che risponde alle chiamate delle interfacce eseguite da un oggetto consumer, trasferendo dati in formato tabellare da un'origine durevole, detta archivio dati, al consumer.
I provider possono essere semplici o complessi. Possono supportare una quantità minima di funzionalità oppure implementare più interfacce e presentare una gamma completa di funzionalità. Un provider è in grado di restituire una tabella, consentendo al client di determinarne il formato, e di eseguire operazioni sui dati della tabella.
Ciascun provider implementa un gruppo standard di oggetti COM per la gestione delle richieste provenienti dal client. Gli oggetti sono definiti standard in quanto qualsiasi consumer OLE DB può accedere a dati da qualsiasi provider in modo indipendente dal linguaggio (C++, Basic e così via).
Ciascun oggetto COM contiene diverse interfacce, alcune delle quali sono obbligatorie mentre altre sono facoltative. Mediante l'implementazione delle interfacce obbligatorie, un provider garantisce un livello minimo di funzionalità, detto conformità, che potrà essere utilizzato da qualsiasi client. Un provider può implementare interfacce facoltative per fornire funzionalità aggiuntive. È opportuno che il client chiami sempre la primitiva **QueryInterface** per determinare se un provider supporta una determinata interfaccia.

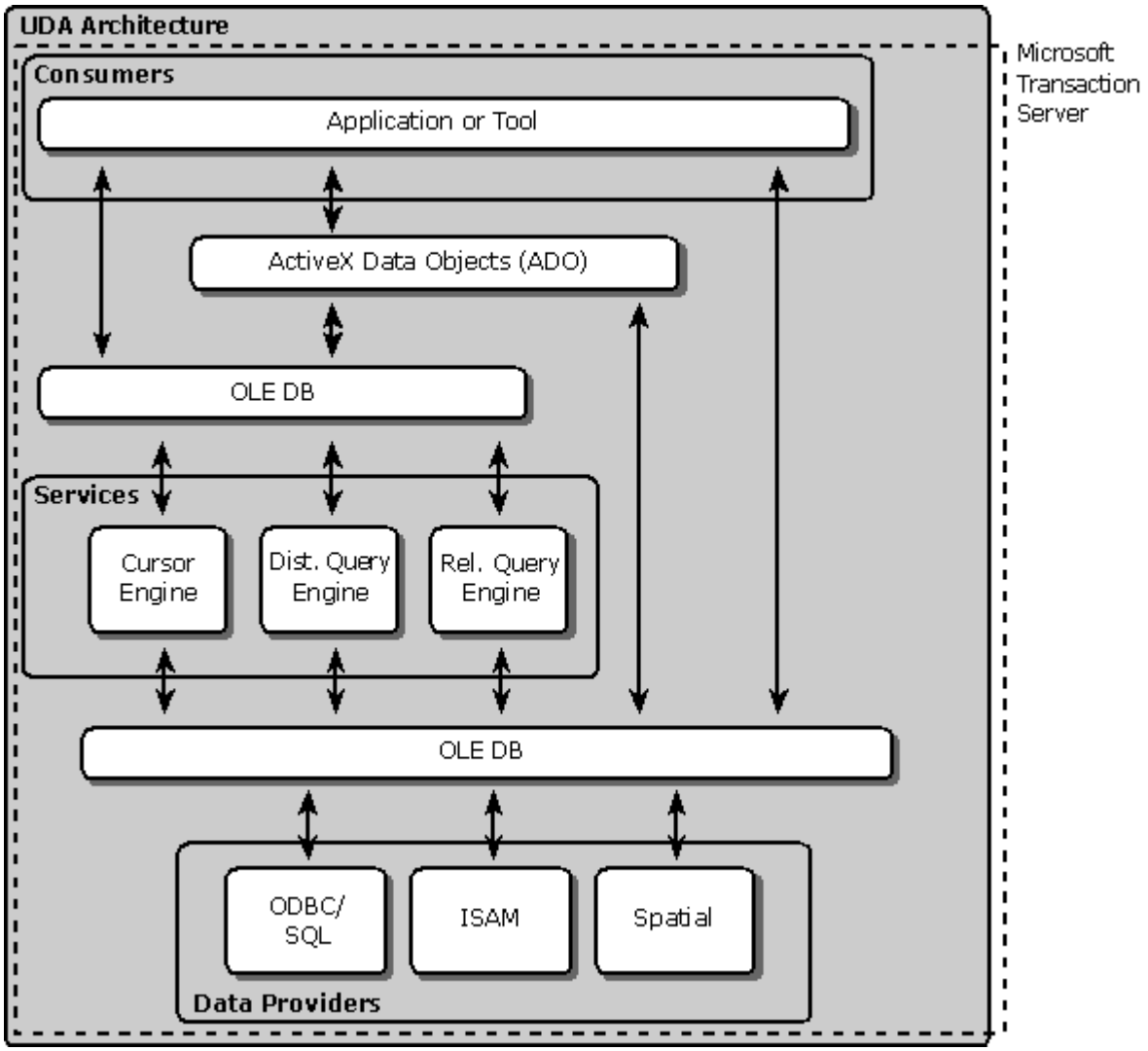


Figura 47: collocazione delle interfacce OLEDB nell'architettura di accesso universale ai dati (UDA).

Non è sempre necessario creare un provider personalizzato, in quanto Microsoft fornisce diversi provider standard precostituiti nella finestra delle proprietà di Data Link di Visual C++. Si crea un provider OLE DB principalmente per potersi avvalere della strategia di accesso universale ai dati e poter quindi fruire di diversi vantaggi, tra cui:

- Accesso ai dati mediante qualsiasi linguaggio, quale ad esempio C++, Basic, Visual Basic, Scripting Edition e altri.
- Possibilità per diversi programmatori in una stessa organizzazione di accedere agli stessi dati nello stesso modo, indipendentemente dal linguaggio utilizzato.
- Esposizione dei dati ad altre origini dati, quali SQL Server, Microsoft Excel, Access e altre. Questa possibilità risulta molto utile se si desidera trasferire dati tra formati diversi.
- Possibilità di eseguire operazioni tra origini dati diverse (eterogenee), con un sensibile vantaggio per il data warehousing.

- Possibilità di mantenere i dati nel formato nativo e poter comunque accedere a essi con una semplice operazione.
- Aggiunta di capacità supplementari ai dati, quali l'elaborazione delle query.
- Miglioramento delle prestazioni di accesso ai dati grazie alla possibilità di controllarne la modifica.
- Maggiore affidabilità.

Una situazione in cui è presente un formato di dati proprietario a cui può accedere solo un programmatore è una situazione a rischio. Grazie ai provider OLE DB è invece possibile rendere disponibile tale formato proprietario a tutti i programmatori.

L'architettura OLEDB è stata costruita sul principio di una applicazione che accede a diverse fonti di dati tramite un'altra applicazione scritta appositamente a questo scopo: il *provider*.

I provider possono essere di due tipi: data providers e service providers.

Un provider di dati possiede i dati e li espone all'esterno in forma tabellare. Esso possiede i dati nel senso che un'applicazione che voglia far riferimento a quello specifico insieme dati può accedervi unicamente tramite il relativo provider. Inoltre per svolgere le sue funzioni, nel fornire dati al consumer, esso è indipendente da qualsiasi altro provider.

I provider di servizi incapsulano servizi di produzione e consumo dati attraverso le interfacce OLEDB. Essi non possiedono dati ed in realtà svolgono un doppio ruolo di provider / consumer (fig. 47).

Microsoft OLEDB Simple Provider (OSP) offre un framework che semplifica il compito di scrittura di un provider su dati in maniera abbastanza semplice. Il toolkit OSP è indipendente dal linguaggio e supporta gli ambienti di sviluppo Microsoft in Visual Basic, Visual C++ e Visual J++.

Comunque i provider scritti con OSP, il cui principale obiettivo è la rapidità, sono meno flessibili ed offrono minori funzionalità di un provider scritto usando le interface native OLEDB.

L'applicazione consumer viene scritta adoperando la tecnologia ADO (ActiveX Data Objects) perfettamente integrabile al "consumo" di dati forniti da interfacce OLEDB. ADO costituisce l'interfaccia di alto livello dei servizi offerti da OLEDB. Il modello ADO si basa su 4 concetti fondamentali: connessione, comendo, tupla e insieme di tuple. La connessione (connection) rappresenta il canale di comunicazione che deve essere stabilito per interagire con una sorgente dati, fornendo la locazione della stessa ed username e password per l'identificazione sul sistema; una componente importante della connessione è l'insieme di errori che possono verificarsi durante la comunicazione, definita anche collezione in quanto più anomalie

possono presentarsi simultaneamente. Il comando (Command) è la stringa di caratteri che contiene l'istruzione SQL che si vuole far eseguire. La tupla (Record) descrive la singola riga di una tabella e fornisce strumenti per il riconoscimento della struttura. L'insieme di tuple (RecorsSet) offre meccanismi per la scansioni di più tuple.

4.2 Introduzione a OLEDB for DM

Un componente per l'approccio a problemi di DM è presente in SQL Server 2000 e 2005, le proposte Microsoft nel settore DBMS. Vista la larga diffusione della piattaforma Windows, questo favorisce il coinvolgimento delle tecnologie di Data Mining in applicazioni di largo utilizzo nelle imprese. A parte gli algoritmi proposti per il training e la costruzione dei modelli, il grande contributo di SQL Server al Data Mining è l'implementazione dello standard OLEDB for DM, proposto da Microsoft come standard industriale e supportato da vari vendors. Esso fa leva su due tecnologie relazionali molto stabili: SQL e OLEDB.

L'industria del DM è molto frammentata, perciò risulta difficoltoso per i produttori di software applicativo e gli sviluppatori delle imprese integrare le differenti tecnologie degli strumenti per la scoperta della conoscenza. Si può considerare il mercato odierno del DM simile a quello dei DBMS prima che venisse introdotto il linguaggio SQL. Ogni fornitore ha il proprio package per il DM di tipo "stand alone", che non comunica con altri applicativi. Se uno sviluppatore ha basato il proprio progetto su uno specifico package ed a un certo punto vorrebbe utilizzare un certo algoritmo proposto da terze parti dovrebbe reistanziare l'intero processo a causa della impossibilità di comunicazione e scambio dati.

Con l'aiuto delle specifiche OLEDM for DM, qualsiasi algoritmo per il DM può essere acceduto tramite la tecnologia OLEDB, facilmente integrabile nelle applicazioni di utente.

Un ulteriore problema, da non sottovalutare, della maggior parte dei prodotti DM è la necessità di trasportare l'insieme dati verso formati specifici e, come si è già sottolineato, la fase di ETL (Extraction, Transformation and Loading) spesso è la più onerosa dell'intero processo (cfr. cap.2). OLEDBDM permette di effettuare tutte le operazioni necessarie direttamente su strutture relazionali, dove tipicamente sono mantenute le informazioni.

Microsoft ha iniziato lo sviluppo delle specifiche OLEDBDM con il supporto di una dozzina di venditori di software per la Business Intelligence, proponendole come interfaccia integrativa fra le diverse proposte sul mercato DM; una sorta di "lingua franca", in stile SQL, grazie alla quale i consumer di servizi DM possano dialogare con i provider che li mettono a disposizione, seguendo quindi il paradigma sul quale si fonda la tecnologia OLEDB.

Al solito, i pacchetti software che incorporano gli algoritmi DM sono detti DM Providers, mentre le applicazioni che intendono utilizzarne le funzionalità sono dette DM Consumer. OLEDB for DM specifica l'interfaccia comune fra le due entità. In fig. 48 è evidenziato il ruolo dell'interfaccia nella comunicazione fra i vari componenti del sistema.

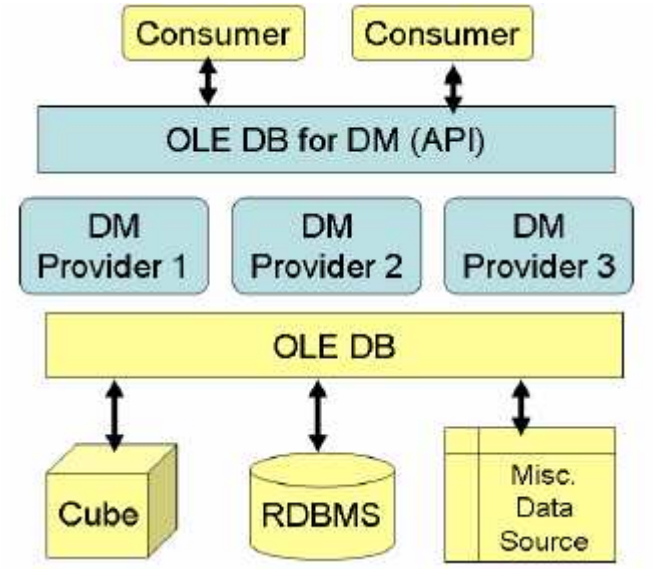


Figura 48: OLEDB for DM come interfaccia di programmazione di applicazioni in ambiente Windows.

OLEDB for DM è un'estensione di OLEDB che supporta le funzioni di Data Mining sui provider. L'obiettivo principale della specifica OLEDBDM è fornire uno standard industriale per lo sviluppo di progetti DM che permette l'integrazione in un unico ambiente di differenti tecnologie e soluzioni algoritmiche proposte da terze parti. OLEDB for DM, insomma, specifica l'interfaccia fra DM Consumers e DM Providers.

Essendo basato sul modello COM, OLEDB for DM introduce un nuovo oggetto virtuale, detto DMM (Data Mining Model), e alcuni nuovi comandi per la sua manipolazione. Nella sua struttura un DMM è molto simile ad una tabella creata con lo statement SQL CREATE. Essa viene popolata, ovvero istruita, con lo statement INSERT INTO ed il client usa lo statement SELECT per esplorarne il contenuto ed effettuare predizioni.

Tuttavia questa nuova entità non viene trattata come una tabella in senso tradizionale, infatti, una volta popolata attraverso il processamento dei dati da parte di un determinato algoritmo, viene mantenuta solo l'astrazione risultante del modello e non l'intero set dei dati di ingresso.

I dati da analizzare sono rappresentati logicamente da una collezione di tabelle relazionali. L'entità o il soggetto sulla quale si basa l'attività di predizione è detta case (ad esempio il case cliente se si sta trattando una analisi delle abitudini di acquisto) e l'insieme delle entità case set. Se sono presenti join fra tabelle dai quali risultano corrispondenze uno a molti fra il case ed altri record, OLEDB for DM fa uso delle tabelle innestate definite dal Data Shaping

Service, incluso nel MDAC (Microsoft Data Access Component). Questo offre una maggiore possibilità di manipolazione dati dato che lo stesso insieme di dati fisici può dare luogo a differenti case set.

Il contenuto di un DMM può essere pensato come una tabella di verità, ovvero una tabella contenente ogni possibile combinazione dei diversi attributi ed un valore di verità che ne attesta l'effettiva esistenza.

4.3 La struttura del Data Mining Model

Come detto il DMM è molto simile ad una tabella SQL. Lo statement SELECT ritorna può selezionare colonne dai dati di ingresso, colonne del modello e le predizioni effettuate dal modello.

4.3.1 Colonne di Modello

Le colonne di modello descrivono tutte le informazioni su uno specifico case. Ad esempio se il case rappresenta un cliente, le colonne riporteranno tutte le informazioni conosciute sul cliente. Un esempio è il seguente:

ID Cliente	Sesso	Colore Capelli	Età	Probabilità Età	Nome Prodotto	Quantità Prodotto	Tipo Prodotto	Auto Possedute	Probabilità Auto
1	M	Nero	35	100%	TV	1	Elettronico	Camion	100%
					PC	1	Elettronico	Furgone	50%
					Mais	2	Cibo		
					Birra	6	Bibite		

Come si vede non tutte le informazioni possono essere descritte da semplici tabelle relazionali, in quanto un case può avere non solo semplici attributi con relazioni univoche, ma anche tabelle multiple che possono avere un numero diverso e variabile di righe. La capacità di un case di contenere tabelle multiple è un requisito chiave per la maggior parte degli algoritmi di mining. Il concetto teorico di tabelle innestate, note anche come colonne tabella, è contemplato e supportato da MDAC tramite l'uso dello statement SHAPE.

Nell'esempio riportato vi sono degli attributi aventi relazioni uno a molti, in particolare possiamo identificare due tabelle innestate:

- "Prodotti Acquistati", contenente "Nome Prodotto", "Quantità Prodotto" e "Tipo Prodotto"
- "Proprietà Auto", contenente "Auto Possedute" e "Probabilità Auto"

La riga principale del case è detta case row; le righe nelle tabelle innestate sono dette nested rows.

Ogni colonna del modello può assumere i seguenti tipi (per brevità sono omesse alcune descrizioni):

- KEY: la colonna che identifica la riga. Nella sintassi si usa il flag di tipo KEY per identificare le colonne chiave.
- ATTRIBUTE: attributo semplice.
- RELATION: informazione usata per classificare attributi secondo particolari gerarchie. Ad esempio "Mais" è classificato come "Cibo". Nella sintassi viene usata la clausola RELATED TO.
- QUALIFIER: un valore speciale, associato ad un attributo, che ha un significato per il provider. Ad esempio la probabilità che un attributo è corretto. I qualificatori sono opzionali e vengono applicati se vi è incertezza sul valore dell'attributo o se il risultato di una predizione viene usato in fase di training di un altro DMM. Nella sintassi i qualificatori sono identificati da un OF.

  Esempi di qualificatori sono: PROBABILITY, VARIANCE, STDEV, SUPPORT, PROBABILITY_VARIANCE, PROBABILITY_STDEV, ORDER.
- TABLE: identifica una colonna contenente una tabella innestata. Per ogni case row, una cella di tipo TABLE contiene tutti i valori della tabella innestata relativi al case. Di conseguenza una colonna così definita contiene l'intera tabella innestata a cui fa riferimento. Nalla sintassi una tabella è definita dall'elenco delle sue colonne contenute nella definizione di una colonna di tipo TABLE.
- DISCRETE.
- ORDERED.
- CYCLICAL.
- CONTINUOUS: per un attributo continuo si può specificare anche il tipo di distribuzione dei valori, questo è un suggerimento che si dà al provider in mancanza del quale esso sceglie il più opportuno. Tipi di distribuzione supportati sono: NORMAL, LOG_NORMAL, UNIFORM, BINOMIAL, MULTINOMIAL, POISSON, T-DISTRIBUTION.
- DISCRETIZED: questo tipo di flag accetta argomenti per imporre il tipo di discretizzazione da eseguire in luogo del tipo di default.
- SEQUENCE_TIME: identifica una colonna contenente unità di misura temporali.

Altri suggerimenti possono essere dati al provider sulla natura dei dati di ingresso, ma la maggior parte dei flag sono dipendenti dal tipo di provider. I seguenti sono due esempi:

- MODEL_EXISTENCE_ONLY: vengono marcati da questo flag gli attributi esistenti ma che non sono rilevanti a scopi di modellazione, ovvero che hanno significato marginale rispetto all'analisi che si vuole effettuare.
- NOT NULL: è un'acquisizione dall'SQL standard. L'attributo relativo non può avere valore nullo altrimenti verrà generato un errore.

4.3.2 Colonne di Predizione

Una colonna di tipo ATTRIBUTE o TABLE può essere una colonna di input, output od entrambe. Il DM Provider costruirà un modello capace di predire e giustificare i valori delle colonne di output sulla base dei valori delle colonne di input.
Le predizioni possono trasportare non solo semplici informazioni del tipo "l'età stimata è 21", ma tante altre statistiche a corredo dell'informazione come il livello di confidenza, la deviazione standard ecc.
Inoltre la predizione può essere una collezione di predizioni come "il set di prodotti che un cliente verosimilmente acquisterà", ed ogni singola predizione può necessitare si statistiche addizionali.
Una predizione può essere espressa con un ISTOGRAMMA. Un istogramma fornisce la possibilità di predire valori multipli accompagnati da un vasto set di statistiche. Quando un istogramma è richiesto, ogni predizione può avere una collezione di valori possibili che costituisce l'istogramma stesso.
Siccome l'informazione convogliata in una predizione è molto ricca, spesso si ha necessità di estrarne solo una porzione; ad esempio "la migliore stima di…", "le prime 3 stime di…", "tutte le stime con probabilità non inferiore al 55%" ecc. Non tutti i Provider né tutti i DMM possono supportare tutti i tipi di richieste, quindi ogni colonna di output definisce i tipi di informazioni che possono essere estratte da essa e quali funzioni di trasformazione, fornite dallo standard, possono essere eseguite su di essa.

4.4 La specifica

Le operazioni fondamentali da effettuare per l'istanza di un processo DM sono le seguenti:

I. Creare un oggetto sorgente dati OLEDB ed ottenere un oggetto di sessione OLEDB. Connettersi ad una fonte dati tramite il meccanismo standard via OLEDB.
II. Creare un oggetto DMM.
    Usando un oggetto di comando OLEDB, il client crea la struttura del modello desiderato eseguendo lo statement CREATE MINING MODEL, del tutto simile al CREATE TABLE in SQL.

III. Inserire i dati di training nel modello.

In maniera analoga al popolamento di una tabella ordinaria, il client adopera lo statement INSERT INTO per istruire il modello con l'insieme dei dati di apprendimento, usando eventualmente lo statement SHAPE per le tabelle innestate.

IV. Usare il modello istruito per derivare predizioni.

Le predizioni sono effettuate con lo statement SELECT su uno speciale tipo di join, detto prediction join, che congiunge il modello con tutti i casi possibili, il DMM, con l'insieme dei case "attuali", ovvero i dati incompleti sui quali è desiderata l'attività di stima.

OLEDB for DM introduce un nuovo tipo di join, codificato con lo statement PREDICTION JOIN, in quanto il processo di combinazione di tutti i case attuali con i case del modello non è così semplice come la semantica di un normale join SQL. Se, ad esempio, il modello mappa perfettamente la struttura dati dei case attuali può essere usato il NATURAL PREDICTION JOIN, ovviando alla clausola ON del join. Tale mappatura avviene secondo i nomi delle colonne dei modelli, come avviene in SQL.

4.4.1 Connessione ad un DMP

Il processo di connessione ad un DM Provider avviene secondo le solite modalità di connessione a qualsiasi altro provider OLEDB, sia esso relazionale, multidimensionale o di qualsiasi altro tipo.

Per prima cosa si dichiara la variabile che rappresenta l'oggetto ADO delegato alla connessione, ad esempio:

```
Dim conn As ADODB.Connection
```

Dopodiché, all'interno della procedura, si attiva una sessione con la sorgente dati:

```
conn.Open "mioProviderDM","nicola","pwdsegreta".
```

Come tutti i provider OLEDB, il DMP supporta i tipi di oggetto data source, session, command e rowset.

Sebbene durante la sessione di connessione un DMP si comporta come un normale Provider OLEDB, può essere utile verificare se lo specifico provider fornisce supporto alle operazioni DM. A questo scopo è definita la costante DBSOURCETYPE_DATASOURCE_DMP e può essere usata quando si enumerano i vari providers per localizzare quello che adotta OLEDB for DM.

Un singolo provider può supportare diversi tipologie di magazzini dati e fornire concorrentemente sia operazioni relazionali che di data mining. Per verificare le tipologie di strutture dati incorporate da un provider si usa la costante SOURCE_TYPE.

Una volta che l'oggetto sessione è stato istanziato, l'applicazione consumer può interrogare il provider ed eseguire numerosi comandi.

4.4.2 Creazione di un DMM

Con il comando CREATE MINING MODEL si crea la struttura per un nuovo DMM. Questa operazione è molto vicina alla comune operazione relazionale CREATE TABLE, con la quale si istanzia un oggetto avente struttura tabellare. Come si vedrà in seguito la creazione ed il popolamento di un DMM segue lo stesso approccio della gestione delle tabelle nei RDBMS.
Questa similarità fra DMM e tabelle tradizionali non è accidentale ma voluta. Infatti lo standard guarda ad una futura integrazione delle tecniche di DM nei RDBMS di larga diffusione e propone la vista di un DMM come un tipo speciale di struttura tabellare.
Comunque, diversamente dalle tabelle, un DMM deve definire preliminarmente gli obiettivi e le tecniche di analisi. Diventa quindi utile determinare le capacità funzionali del Provider.

DETERMINARE LE CAPACITÀ DI UN PROVIDER

I differenti servizi di mining offerti da un Provider sono descritti da strutture di metadati detti ***schema rowset***, dove al fianco del nome dello specifico algoritmo sono riportate tutte le sue caratteristiche, gli obiettivi perseguibili e le limitazioni.
Algoritmi diversi sono capaci di predizioni diverse e la scelta della tecnica adatta ai propri scopi non è banale. Scorrendo lo schema rowset (che si è deciso di non riportare per brevità)si possono ottenere molte informazioni sulle tecniche proposte, fra cui:

Nome colonna	**Descrizione**
SERVICE_NAME	Il nome dell'algoritmo usato per creare il modello
SERVICE_TYPE_ID	I tipi di servizi offerti: clustering, regole di associazione, ecc.
PREDICTED_CONTENT	Il tipo di attributo che può essere stimato
PREDICTION_LIMIT	Il numero massimo di predizioni
SUPPORTED_DISTRIBUTION_FLAGS	I tipi di distribuzioni continue supportate
SUPPORTED_INPUT_CONTENT_TYPES	I tipi di dati di input accattabili
TRAINING_COMPLEXITY	Indicazioni sul tempo previsto per il training
PREDICTION_COMPLEXITY	Indicazioni sul tempo previsto per la predizione
EXPECTED_QUALITY	La qualità attesa del modello
ALLOW INCREMENTAL INSERT	Possibilità di training incrementali
ALLOW_DUPLICATE_KEY	Capacità di gestire chiavi multiple

Tabella 6: una parte dello schema rowset Mining Services.

DEFINIRE UN NUOVO MODELLO

Per creare la struttura di un DMM si usa lo statement CREATE MINING MODEL. Questo comando definisce solo le proprietà del modello e non il contenuto con il quale verrà istruito né tantomeno la particolare struttura grafica che apprenderà (alberi decisionali, tabelle, cluster, ecc.).
Con lo statement si definiscono:

1. Le colonne del DMM
2. L'algoritmo usato

La sintassi generica è:

```
CREATE MINING MODEL <nome_modello> (<definizione_colonne>) USING
<Algoritmo>[(<argomenti_algoritmo>)]
```

Nella definizione delle colonne del DMM sono richieste varie informazioni oltre al tipo, come i vari flag che qualificano l'attributo. Vediamo al tal proposito un esempio:

```
CREATE MINING MODEL [Predizione Età]
(
      [ID Cliente]                  LONG        KEY,
      [Sesso]                       TEXT        DISCRETE,
      [Colore Capelli]              TEXT        DISCRETE,
      [Età]                         DOUBLE      DISCRETIZED() PREDICT,
      [Probabilità Età]             DOUBLE      PROBABILITY OF [Età],
      [Prodotti Acquistati]         TABLE
      (
            [Nome Prodotto]         TEXT        KEY,
            [Quantità]              DOUBLE      NORMAL CONTINUOUS
            [Tipo Prodotto]         TEXT        RELATED TO [Nome Prodotto]
      ),
      [Auto Possedute]              TABLE
      (
            [Nome Auto]             TEXT        KEY,
            [Probabilità]           DOUBLE      PROBABILITY OF [Nome Auto]
      )
)
USING [Microsoft_Decision_Trees]
```

Il nome della colonna ed il tipo di dato sono obbligatori, tutti gli altri sono opzionali. L'attributo Età, che è l'oggetto della predizione, è qualificato con il flag PREDICT; inoltre esso, essendo continuo, è discretizzato con il metodo di default (il flag DISCRETIZED non ha argomenti). La sintassi completa della funzione è la seguente: DISCRETIZED([method,[n]]). Entrambi i parametri sono opzionali ma le parentesi sono obbligatorie. Il parametro "method" indica il metodo che il Provider userà per segmentare gli intervalli ed "n" è il numero raccomandato di segmenti. Le tecniche per la discretizzazione sono dipendenti dal Provider, alcuni valori plausibili sono: AUTOMATIC, EQUAL_AREAS, THRESHOLDS, CLUSTERS, e così via.

La colonna "Probabilità Età" indica la probabilità che la predizione sia esatta ed è qualificata dall'apposito flag. Discorso analogo per l'attributo "Probabilità" nella colonna "Auto Possedute".

Si notino le due colonne di tipo TABLE "Prodotti Acquistati" e "Auto Possedute" che a loro volta contengono altre colonne che formano le tabelle innestate.

Per "Quantità" è specificato anche il tipo di distribuzione e "Tipo Prodotto" è relazionato gerarchicamente a "Nome Prodotto".

Una colonna può avere valori mancanti. Vi sono diversi modi per gestire tali valori:

- NOT NULL: la colonna non può avere valori mancanti
- IGNORE NULL: i valori mancanti vengono ignorati
- NULL INFORMATIVE: l'algoritmo modella gli stati mancanti

L'opzione di default è NULL INFORMATIVE in quanto può essere utile ai fini dell'analisi che l'algoritmo modelli tali valori.

Dopo la definizione delle colonne è specificato il tipo di algoritmo usato in fase di training. La clausola USING può essere seguita da coppie parametro-valore specifiche del Provider. La lista completa dei parametri forniti è contenuta in un altro schema rowset detto SERVICE_PARAMETER dove sono riportate anche informazioni sul tipo di parametro, ad esempio se è obbligatorio o facoltativo, ed una maschera di bit che descrive le caratteristiche, ad esempio se di training o di predizione. Ogni Provider definisce i nomi dei propri parametri. Una possibile lista di parametri può essere:

- HOLDOUT_PERCENTAGE: la percentuale di dati mantenuti in fase di training.
- SAMPLE PERCENTAGE: la percentuale di dati di training ottenuti dal campionamento
- SAMPLE SEED: il seme usato nel campionamento.

Quando lo statement di creazione del DMM è eseguito, il modello apparirà nello schema rowset MINING_MODELS del Provider. Esso dovrà essere poi popolato per effettuare operazioni di analisi utili.

COPIARE UN MODELLO

In alcuni casi si può avere la necessità di eseguire algoritmi differenti per istruire lo stesso modello con il medesimo set di dati di ingresso. OLEDM for DM fornisce un meccanismo per creare facilmente un nuovo modello da uno esistente. La sintassi del comando è:

```
SELECT * INTO <nuovo_modello> USING <tipo modello> [( <lista parametri> )]
FROM <modello esistente>
```

Tutte le informazioni del modello non attinenti al particolare algoritmo adoperato verranno copiate nel nuovo modello. Se si vuole adoperare un algoritmo di apprendimento differente, una volta settato nella query di training, basterà eseguire questo statement per avviare il popolamento della nuova struttura. Se il modello non è popolato, verrà copiato solo lo schema.

CREAZIONE DI UN MODELLO DA PMML

La struttura e il contenuto di un qualsiasi DMM può essere espresso in PMML (Predictive Modeling Markup Language), un'estensione del linguaggio di gestione dati XML. Esso fornisce un metodo per definire statistiche e modelli DM e per condividerli fra applicazioni diverse compatibili.

Tale metodo è indipendente dai particolari ambienti di sviluppo proposti dai vari vendors, cosicché le questioni proprietarie e le incompatibilità che possono presentarsi non siano più una barriera allo scambio di modelli fra le applicazioni. In questo modo è possibile sviluppare i modelli all'interno di un'applicazione ed utilizzare un'altra applicazione per visualizzare, analizzare, valutare e manipolare ulteriormente il modello.

Essenso basato sullo standard XML, le sue specifiche sono nella forma di XML DTD (Document Type Definition).[3]

È plausibile, quindi, che un utente esperto abbia concepito il proprio modello in PMML e si vuole importare tale modello in una struttura DMM.

I Provider non devono necessariamente supportare l'inizializzazione di un modello da PMML, per attestare tale capacità si deve consultare lo schema rowset MINING SERVICES alla colonna ALLOW_PMML_INITIALIZATION. Se la variante è vera si può utilizzare il seguente statement:

```
CREATE MINING MODEL <nome modello> FROM PMML <stringa xml>.
```

4.4.3 Esplorazione di un DMM

Una volta che un modello è stato creato o identificato può essere utile esplorarne la struttura. La struttura di un DMM è del tutto simile alla struttura di una tabella che è rappresentata dall'insieme delle sue colonne, che specificano il tipo di input o di output di cui il modello è dotato. Così come una tabella, il modello è indipendente dalle particolari istanze dei dati che vi risiedono o che saranno inseriti in esso.

Tutte le informazioni relative alla struttura del modello sono riportate nello schema rowset MINING_COLUMNS.

COLONNE DI INPUT

La struttura di un DMM è costituita dalle colonne di input e dalle colonne di output (o di predizione). Nel MINING_COLUMNS schema rowset sono riportate, per ogni modello presente, tutte le informazioni relative ad ogni singola colonna, come la tipologia, il contenuto ed i flag qualificatori.

[3] La documentazione ufficiale è disponibile presso il Data Mining Group: http://www.dmg.org/v1-1/dtd_v1_1.html

Nome Colonna	Descrizione
COLUMN_NAME	Nome della colonna
DATA_TYPE	Indicatore del tipo di dato, ad es. TABLE, TEXT
DISTRIBUTION_FLAG	Indicatore del tipo di distribuzione continua. Ad es.:NORMAL, POISSON. Sono definibiliti tipi specifici del provider.
CONTENT_TYPE	Indicatore sul tipo di contenuto, ad es. KEY, PROBABILITY. Sono definibiliti tipi specifici del provider.
MODELING_FLAG	Lista di flag separati da virgole, ad es. MODEL_EXISTENCE_ONLY, NOT NULL. Sono definibiliti tipi specifici del provider.
RELATED_ATTRIBUTE	Nome della colonna a cui è relazionata o di cui essa è proprietà.
CONTAINING_COLUMN	Nome della colonna TABLE in cui essa è contenuta se si tratta di tabella innestata.
IS_INPUT	Valore VERO se la colonna è di input
IS_PREDICTABLE	Valore VERO se la colonna è di output

Tabella 7: Versione abbreviata dello schema rowset Mining Columns.

Ogni colonna del modello può essere di input ed acquisire dati dall'insieme di apprendimento.

COLONNE DI PREDIZIONE

Una colonna di tipo ATTRIBUTE o TABLE può essere di input, di output o entrambi. Come già visto nella sintassi di creazione del DMM una colonna di predizione è contraddistinta dal flag PREDICT, che indica il fatto che i dati di input verranno utilizzati per stimare i valori di tale colonna, inoltre suddetta colonna sarà coinvolta nelle stime di eventuali altre colonne di predizione (colonna di I/O). Se invece si marca la colonna con il flag PREDICT_ONLY essa non verrà usata per ulteriori predizioni. L'assenza del qualificatore indica una colonna di input.

Una colonna TABLE che contiene colonne PREDICT diviene essa stessa di predizione.

Se una colonna è di predizione, ovvero se IS_PREDICTABLE è settato a vero, nello schema rowset si trovano colonne addizionali che specificano le informazioni di predizione disponibili e quali particolari funzioni di estrazione possono essere eseguite sulla colonna.

4.4.4 Popolamento di un DMM

Dopo che la struttura del DMM è stata definita, si può procedere all'inserimento dei dati di training nel modello. Questa operazione si effettua tramite l'esecuzione dello statement INSERT INTO, del tutto simile al comando INSERT che si adopera per il popolamento di una normale tabella.

Lo stadio di popolamento del modello eseguirà l'algoritmo prescelto sull'insieme di ingresso e genererà un modello predittivo, detto anche contenuto del DMM (DMM content).

La differenza sostanziale con il popolamento di una tabella relazionale sta nel fatto che il modello non memorizzerà l'intero set di dati, case per case, ma solo i valori distinti delle colonne ed alcune altre informazioni necessarie.

Questa fase spesso implica un processamento intensivo di dati e può essere onerosa oltre che richiedere tempo. È disponibile un meccanismo di notifica sullo stato di avanzamento del processo oltre a delle interfacce OLEDB di cancellazione asincrona dell'esecuzione in corso. Le interfacce menzionate sono **IDBAsynchStatus** e **IConnectionPointContainer** (che permette all'utente di ottenere un punto di connessione con l'interfaccia **IDBAsynchNotify**); esse sono supportate dal command object del Provider.

INSERIRE I CASE

La sintassi del comando è:

```
INSERT [INTO] <nome modello>
      [ <mapping colonne> ]
      <query sorgente>
```

Come sarà descritto successivamente varie sintassi sono disponibili per la composizione della query di recupero dei dati sorgente. Tralasciando per ora questo aspetto, il legame fra le colonne del DMM e le colonne di input è basato sull'ordine delle stesse, seguendo così ancora l'analogia con SQL; nel caso in cui si vuole esplicitare il mapping lo si fa nella clausola facoltativa <mapping colonne>. In tali casi spesso si fa uso della sintassi SHAPE, che permette la gestione di tabelle innestate.

La sintassi generica è:

```
SHAPE {<query primaria>}
            APPEND ({ <query secondaria> }
            RELATE <colonna principale> TO <colonna secondaria>)
            AS <nome colonna tabella>
      [
            APPEND ({ <query secondaria> }
            RELATE <colonna principale> TO <colonna secondaria>)
            AS < nome colonna tabella>
            …
      ]
```

Lo statement permette l'aggiunta di colonne TABLE ad una query principale da una query secondaria specificando il "matching" fra le righe nella clausola RELATE e permettendo l'aliasing della colonna contenente.

Usando questa sintassi è possibile leggere i dati da case derivanti da interrogazioni multiple e modellarli in una singola tabella da fornire al DMM.

Riportiamo un esempio a scopi chiarificatori sul modello precedentemente definito

```
INSERT INTO [Predizione Età]
(
      [ID Ciente], [Sesso], [Età], [Probabilità Età],
      [Prodotti Acquistati] (SKIP, [Nome Prodotto], [Tipo Prodotto], [Quantità]),
      [Auto Possedute] (SKIP, [Nome Auto], [Probabilità Auto])
```

```
)

SHAPE { select [ID Cliente], [Sesso], [Età], [probabilità Età]
                  from [Clienti] order by [ID Cliente]}
      APPEND ( {select [ClienteID], [Nome Prodotto], [Tipo Prodotto] , [Quantità]
                           from [Vendite] order by [ClienteID] }
            RELATE [ID Cliente] TO [ClienteID])
      AS [Prodotti Acquistati],
       ( {select [ClienteID], [Nome Auto], [Probabilità],
                        from [Auto Clienti] order by [ClientiID] }
            RELATE [ID Cliente] TO [ClienteID])
      AS [Auto Possedute]
```

Si è supposto che il magazzino dei dati a disposizione sia costituito da 3 tabelle denominate "Clienti"(query principale), "Vendite" e "Auto Clienti" (query secondarie).

Osserviamo che si è usato il flag SKIP per "saltare" le chiavi delle tabelle secondarie per evitare ridondanza di dati.

Le colonne delle tabelle secondarie relazionate alla tabella primaria devono essere ordinate secondo lo stesso criterio della chiave nella query principale.

Come si vede la sintassi SHAPE è ricca ed i Provider sono incoraggiati a supportarla il più possibile. Un Provider che voglia essere definito OLEDB for DM deve almeno supportare la sintassi generica su riportata.

POPOLARE COLONNE SINGOLE

In generale, durante la fase di training, il modello apprende i valori distinti delle varie colonne. Vi sono casi in cui è opportuno immettere esplicitamente particolari valori nel modello indipendentemente dall'insieme di apprendimento.

Un esempio di tali valori può essere l'attributo ciclico che rappresenta i giorni della settimana, in cui le istanze sono ordinate come lunedì < martedì < mercoledì ecc. Infatti, non vi è garanzia che i valori siano immessi nell'ordine esatto dai dati di training.

Ancora, nel caso di relazioni gerarchiche fra attributi, non vi è bisogno di specificare ogni volta che l'istanza compare tale legame, ma lo si può comunicare al modello prima di istanziare il training vero e proprio.

OLEDB for DM offre il seguente comando per istruire specifiche colonne:

```
INSERT INTO <nome modello>.COLUMN_VALUES(<mapping colonne>)
<query sorgente>
```

A differenza del modello in sè, per i valori delle colonne è possibile un apprendimento incrementale. Tuttavia, se all'atto della creazione del modello sono state definite colonne legate dalla clausola RELATED TO, tali colonne devono essere caricate simultaneamente.

Riferendosi al solito modello, si riporta un esempio:

```
INSERT INTO [Predizione Età].COLUMN_VALUES([Prodotti Acquistati].[Nome Prodotto],
                                           [Prodotti Acquistati].[Tipo Prodotto])
OPENROWSET('SorgenteOLEDB', 'conn.Open "mioProviderDM", "nicola", "pwdsegreta"',
                  'SELECT DISTINCT [Nome Prodotto], [Tipo Prodotto] FROM Vendite')
```

Una volta inserite le colonne del caso, l'applicazione consumer può esplorarne il contenuto, ma ancora non può effettuare predizioni né esplorare il contenuto del modello. Inoltre, se le colonne relazionate con RELATED TO sono state inserite, esse si possono omettere in fase di apprendimento del DMM.

4.4.5 Dati sorgente

La parte <query sorgente> degli statement INSERT e SELECT FROM PREDICTION JOIN (che sarà illustrato in seguito) è descritta dalla colonna SUPPORTED_SOURCE_QUERY nello schema rowset MINING_SERVICES.

I possibili valori per questa colonna sono:

- SINGLETON CONSTANT
- SINGLETON SELECT
- OPENROWSET
- SELECT
- SHAPE

Se il DM Provider è integrato in un RDBMS che supporta la gestione di tabelle innestate, il processo di apprendimento avviene sotto il controllo del gestore. Spesso accade però che il Provider di servizi DM sia separato da esso e che il gestore non supporti nativamente il concetto di colonna TABLE.

Gli sviluppatori di DM Provider sono incoraggiati ad integrare almeno uno dei meccanismi di manipolazione di tabelle innestate. Ogni Provider deve pubblicare nello schema dei servizi offerti quali metodi esso supporta.

I meccanismi per fornire dati sorgente sono descritti di seguito.

SINGLETON CONSTANT

Questa sintassi permette l'inserimento di un case come un insieme di valori costanti espliciti in luogo della <query sorgente>.

```
<singleton constant> ::= (<valore o set di valori> [,<valore o set di valori>] )

<valore o set di valori> ::= <valore> | (<set di valori>)
```

una possibile instanza del comando è la seguente:

```
('1', 'Maschio', ('TV', 1), ('PC', 2), 'Furgone', 'Automobile')
```

SINGLETON SELECT

Questa sintassi permette l'inserimento di un case come selezione di valori costanti in luogo della <query sorgente>, permettendo selezioni innestate e l'aliasing dei nomi secondo lo standard SQL.

```
<singleton select> ::= <select costante composto> as <alias>

<select costante composto> ::= <select costante> |
            <select costante composto> UNION <select costante composto>

<select costante> ::=   (SELECT <lista alias>)

<lista alias> ::= <elemento alias> |
                        <lista alias>, <elemento alias>

<elemento alias> ::= <COSTANTE> |
                        <COSTANTE> as <alias> |
                        <singleton select>
```

L'esempio seguente è una valida sintassi:

```
(SELECT 21 as Età, 'Maschio' as Sesso,
      ((SELECT 'prosciutto' as Prodotto, 10 as Quantità) UNION
      (SELECT 'birra' as Prodotto, 1 as Quantità)) as Acquisti)
as Case
```

OPENROWSET

Se il Provider supporta il meccanismo OPENROWSET, è possibile eseguire un comando esterno in luogo della <query sorgente>.

Come si è già accennato, molti Provider non sono integrati nella base dati, quindi in moltissimi casi vi sarà l'esigenza di leggere i dati da una sorgente esterna. La funzione OPENROWSET supporta questa funzionalità ed ha la seguente sintassi:

```
OPENROWSET('nome provider','stringa connessione','sintassi query')
```

Il primo parametro è il nome del provider; il secondo è la stringa di connessione al provider (cfr par. 4.4.1); il terzo è l'interrogazione alla base dati e può essere sia semplice che modellata con SHAPE. Il DM Provider stabilisce una connessione con l'oggetto sorgente, esegue la relativa query e ritorna l'insieme di tuple desiderate (rowset).

SELECT

Se il Provider lo permette, il meccanismo di selezione standard di SQL può essere usato come interrogazione alla sorgente dati.

SHAPE

Se il Provider supporta SHAPE come valore per SUPPORTED_SOURCE_QUERY, è possibile usare la sintassi SHAPE per ottenere i dati modellati in tabelle innestate ed inserire i case tramite gli statement di popolamento.

Sui provider relazionali più comuni, una interrogazione singola non è in grado di ritornare dati struttrati in tabelle multiple necessarie per il popolamento della maggior parte dei DMM.

OLEDB for DM fornisce varie alternative per effettuare questa operazione.

MDAC Data Shaping Services

Il Data Shaping Service è un tipo di provider OLEDB che può essere visto come strato superiore agli altri provider. Esso può essere invocato via OPENROWSET nella maniera seguente:

```
OPENROWSET('MSDataShape','Data Provider=SQLOLEDB', 'query shape')
```

L'ultimo parametro della funzione è la stringa in stile SQL che identifica l'interrogazione innestata. Essa deve seguire la sintassi descritta nel par. 4.4.4.

Supporto integrato

Alcuni DM Provider possono adottare essi stessi lo standard della sintassi SHAPE. In tal caso, la query SHAPE non deve essere eseguita nel contesto del comando OPENROWSET.

Supporto nativo

Questo è il caso in cui il DM Provider è integrato nativamente nel provider di dati. Se il gestore coinvolge il trattamento di tabelle innestate, verosimilmente vi sarà l'adozione di una sintassi propria di specifica. OLEDB for DM non preclude questa possibilità.

4.4.6 Esplorazione del contenuto del DMM

Oltre ad esplorare la struttura di un DMM, può essere significativo per l'analista navigare nel contenuto che il modello ha acquisito nella fase di training. Questa navigazione nella struttura grafica del modello può fornire molti suggerimenti alla comprensione della natura dei dati.

Il contenuto di un modello è, in ultima analisi, l'insieme delle regole, delle formule, dei nodi, delle classificazioni, delle distribuzioni che esso mette a disposizione come rappresentazione esplicativa dello schema.

Ogni modello istruito possiede una sua peculiare struttura grafica in dipendenza della tecnica di mining adoperata. Un modello di classificazione basato su alberi decisionali avrà una rappresentazione diversa da un modello regressivo o da un modello di segmentazione.

Uno delle possibilità di esplorare il modello è estrarne una descrizione in XML. Tale descrizione è contenuta nello schema rowset TABLES. Le stringhe XML forniscono un semplice modo per ottenere, manipolare e memorizzare la struttura informativa del documento. Per contro, sono richieste notevoli potenzialità a supporto del consumer.

Un'alternativa, che è anche la più popolare, è la possibilità di navigare il modello nella sua rappresentazione come grafo diretto. L'albero decisionale ne è un classico esempio.

La navigazione ad albero è definita nella specifica OLEDB for OLAP e OLEDB for DM adotta un meccanismo analogo per l'attraversamento dei nodi di un DMM. Nello schema rowset MINING_MODEL_CONTENT è riportato un vasto set di funzioni a tale scopo.

Un'altra possibilità è data dall'interrogazione diretta al modello tramite la query:

```
SELECT * FROM <nome modello>.CONTENT
```

Essa restituirà lo schema rowset MINING_MODEL_CONTENT, che fornisce una descrizione esaustiva di ogni entità del modello, ognuna rappresentata da una riga della metatabella.

4.4.7 Esplorazione di tutti i case possibili e dei valori distinti di colonna

Come già detto in precedenza, il modello memorizzerà i dati in una struttura composta da tutte i possibili case dati da tutte le possibili combinazioni fra i valori degli attributi. A seconda dei valori che incontrerà nel set di training esso terrà nota dei relativi stati dei singoli attributi.

Il set di tutti i case possibili è quindi formato da varie entry, ognuna delle quali rappresenta uno dei possibili case ottenuti dal prodotto cartesiano degli insiemi degli attributi costituenti.

Per attributi discreti tale insieme è una lista dei valori osservati nella colonna con l'aggiunta dello stato "missing". Ad esempio, l'attributo "Sesso" ha gli stati: {"M", "F", "Missing"}.

Per gli attributi continui non vengono mantenuti tutti i valori distinti (nel caso di training set molto grandi ciò comporterebbe un significativo spreco di risorse), ma sono riportati solo i valori "Maximum", "Minimum", "Mean" e "Missing".

Per attributi discretizzati sono mantenuti i soli segmenti ottenuti dal processo, rappresentati dal valore medio fra il limite inferiore e superiore del segmento.

Per ottenere tale set dei case possibili si fa uso del SELECT sul modello. Insieme ad ogni case possibile il DMM riporta varie statistiche sull'attributo oggetto della predizione:

```
SELECT *, PredictProbability(Colore Capelli) FROM DMMPredizioneColoreCapelli
```

Per filtrare una struttura del genere, che può avere una dimensione notevole, si può immettere nella query la clausola WHERE:

```
SELECT Età, PredictProbability(Colore Capelli) FROM DMMPredizioneColoreCapelli
WHERE Sesso = 'M' and Colore Capelli = 'Nero'
```

Sesso	Età	Colore Capelli	P(Colore Capelli)
M	2	Nero	.667
M	91	Nero	.300
M	45	Nero	.667
M	NULL	Nero	.600

VALORI DI COLONNA DISTINTI

Per elencare i valori distinti di una colonna del DMM, si può usare lo statement SQL SELECT DISTINCT:

```
SELECT DISTINCT Colore Capelli FROM DMMPredizioneColoreCapelli
```

In teoria si potrebbe elencare il contenuto di una colonna TABLE, ma in realtà questa operazione è priva di senso pratico. Questo perché l'insieme dei valori possibili per una colonna TABLE sono tutte le possibili tabelle aventi le possibili combinazioni delle chiavi delle tabelle innestate. Non c'è ragione quindi che un Provider implementi un'operazione che fornisca tale risultato.

Comunque, selezionare i valori distinti di una colonna dall'insieme delle possibili tabelle innestate può essere un'operazione desiderabile. Considerando il solito modello esemplificativo, una possibile interrogazione è:

```
SELECT DISTINCT [Prodotti Acquistati].[Nome Prodotto] FROM [Predizione Età]
```

Si noti l'uso dell'operatore "." all'interno dello scope della tabella innestata.

4.4.8 Interrogazioni di predizione: applicazione di DMM su dati attuali

Una volta che il modello è istruito, ed avendo a disposizione il nuovo insieme di dati (tipicamente diverso da training set) obiettivo dell'analisi, si può procedere ad effettuare le interrogazioni di predizione.

1.4.2.2 Componenti

Le interrogazioni di predizione sono recuperate dal DMM tramite un comando di SELECT, la cui sintassi generica è:

```
SELECT [FLATTENED] <espressione SELECT>
FROM <nome modello> PREDICTION JOIN <query sorgente> ON <condizione di join>
[WHERE <espressione WHERE>]
```

Procediamo ora alla descrizione delle varie parti componenti la query.

QUERY SORGENTE

La clausola <query sorgente> identifica la stringa SQL di recupero dati, diretta verso la sorgente contenente i nuovi dati su quali attributi verrà effettuata la stima (cfr. par. 4.4.5).

PREDICTION JOIN

Durante l'attività di predizione, i valori dei nuovi case (detti anche case attuali) vengono confrontati con tutti i possibili case appresi dal DMM specificato nella clausola <nome modello>. Esaminiamo la semantica del Join di predizione, fondamentalmente diversa dal join tradizionale, facendo alcune osservazioni:

- Il DMM memorizza solo alcuni valori (massimo, minimo, medio e valore mancante) per attributi continui. Poniamo che nel nostro caso, per l'attributo "Età", siano stati dedotti i valori: Maximum = 95, Minimum = 2 e Mean = 35. Un comando del tipo seguente non restituirà alcun record:

```
SELECT * FROM DMMPredizioneSesso WHERE Sesso = 'M' AND Age = 30
```

Un Prediction Join usa la rappresentazione del modello (ad esempio una struttura ad albero) per trovare la distribuzione dell'attributo oggetto della predizione (Nero = 0.667, Grigio = 0.267, Missing = 0,067).

- I case di un DMM rappresentano tutti i possibili stati di una colonna di predizione, mentre l'utente spesso si aspetta la migliore predizione. Questo è proprio il risultato restituito dal join di predizione. Selezionando infatti l'attributo "Colore Capelli" usando il Prediction Join verrà restituito lo stato avente probabilità più elevata.
- Il join di predizione effettua alcune assunzioni ed aggregazioni quando deve effettuare il confronto fra i dati sorgente ed il suo contenuto.

  In generale, il Prediction Join prende un case attuale alla volta e recupera un set di case nel DMM sulla base della condizione ON. Questo insieme di case viene collassato in un unico case, in modo specifico dall'algoritmo, che contiene le migliori predizioni per tutti le colonne predicabili del modello. Questo case "collassato" porta con sé delle statistiche di predizione non direttamente osservabili nell'insieme di tutti i possibili case del modello, perché risultanti dal processo di aggregazione.

ESPRESSIONE DI SELEZIONE

La clausola <espressione SELECT> è una sequenza di espressioni separate da virgole, ognuna delle quali può essere un semplice riferimento ad una colonna o una successione di invocazioni a funzioni di predizione. I riferimenti possono essere sia al DMM che alla sorgente dati. Quando si presenta un conflitto di nomi fra DMM e sorgente, il riferimento deve essere prefissato ripettivamente con il nome del modello o con l'alias della colonna.

Al fine di valutare l'accuratezza del modello popolato, si possono effettuare delle predizioni su case di cui sono già noti i valori (tipicamente una parte dell'insieme di apprendimento riservata a tale scopo). Si può usare la clausola SELECT per recuperare e confrontare le colonne del modello con le colonne note della sorgente.

CONDIZIONE DI JOIN

L'esistenza di colonne chiave nelle strutture relazionali sono utili a molti scopi e necessarie a mantenere la consistenza della base dati. Tali chiavi possono non essere usate in fase di training, infatti il modello non mantiene i valori di tali attributi, in quanti ogni riga dell'insieme di tutti i possibili case è unica.

Il "matching" fra i case della query sorgente e del modello è effettuato sulla base della clausola <condizione di join> sulla parola chiave ON. La condizione di join è costituita da una successione di espressioni di uguaglianza ("=") unite dalla parola chiave AND. I riferimenti alle colonne possono essere semplici o prefissati da nomi di modello o alias, e

possono avere vari livelli di profondità per raggiungere i contesti di eventuali colonne TABLE innestate.

Una possibile codifica di una condizione di join può essere:

```
SELECT … ON M1.Sex = T2.Sesso AND
M1.[ProductPurchases].[ProductName]=T2.[Prodotti Acquistati].[Nome Prodotto]
```

Nella clausola FROM (non riportata) il DMM è stato rinominato come M1, mentre nella query sorgente la struttura è stata nominata T2.

Nella situazione favorevole in cui vi è un matching di nomi fra il DMM e lo schema della query di input, si può utilizzare la parola chiave NATURAL PREDICTION JOIN ed omettere la clausola ON.

FILTRAGGIO DI UNA PREDIZIONE

La clausola <espressione WHERE> supporta una forma semplificata della semantica SQL standard allo scopo di limitare i case restituiti da una predizione. I riferimenti alle colonne hanno la stessa semantica della espressione di SELECT.

Riportiamo di seguito un esempio di interrogazione di predizione sullo schema del modello creato in precedenza.

```
SELECT
      T1.[ID Cliente], T1.[Sesso], M1.[Età]
FROM
      [Predizione Età] as M1 PREDICTION JOIN
            OPENROWSET('MSDataShape',
                  'provider dati=Microsoft.Jet.OLEDB.4.0; sorgente
                                    dati=D:\clienti.mdb',
            'SHAPE { SELECT [ID Cliente], [Sesso]
                                          FROM [Clienti] ORDER BY [ID Cliente]}
                  APPEND      ( {SELECT [ClienteID], [Nome Prodotto], [Quantità]
                              FROM [Acquisti Clienti] ORDER BY [ID Cliente] }
                  RELATE [ID Cliente] TO [ClienteID]) AS [Prodotti Acquistati],
                              ( {SELECT [ID Cliente], [Nome Auto]
                              FROM [Auto Clienti] ORDER BY [ClienteID] }
                  RELATE [ID Cliente] TO [ClienteID]) AS [Auto Possedute]') as T1
      ON M1.Sesso = T1.Sesso AND
      M1.[Prodotti Acquistati].[Nome Prodotto] = T1.[Prodotti Acquistati].[Nome
      Prodotto] AND
      M1.[Prodotti Acquistati].[Quantità] = T1.[Prodotti Acquistati].[Quantità] AND
      M1.[Auto Possedute].[Nome Auto] = T1.[Auto Possedute].[Nome Auto]
WHERE PredictProbability(M1.Età) > .8
```

La query ritorna l'età stimata per un insieme di nuovi clienti dove la probabilità di predizione predizione è superiore all'80%.

1.4.3 Informazioni di predizione

Insieme alle stime, le query di predizione possono richiedere statistiche supplementari ricavate in fase di apprendimento. Nel DMM non vi sono colonne esplicitamente dedicate a

contenere tali valori, ma possono essere selezionate invocando le opportune funzioni, che di solito prendono come argomento la colonna di predizione.
Alcune di queste funzioni, come si vedrà in seguito, riportano semplici valori scalari sulla confidenza della predizione e forniscono un controllo a grana fine sull'attività del processo. Altre funzioni ritornano, invece, intere strutture tabellari contenenti dettagli caratteristici dell'attivitàdi stima.

1.4.3.1 Funzioni scalari

Selezionando semplicemente una colonna di predizione è una scorciatoia per eseguire il comportamento di default della funzione Predict(). Essa ritornerà la predizione migliore sulla colonna, ovvero quella avente maggiore probabilità (o quella che il Provider decide essere la migliore secondo i suoi criteri). Quando l'argomento della funzione è una colonna semplice (ovvero non TABLE), il valore restituito è uno scalare.
Come già osservato, il modello considera lo stato mancante ("Missing") come uno dei possibili modellabili. In alcuni casi il risultato di una predizione che dia il valore mancante può essere considerato un "non risultato", ma in taluni altri casi può essere informativo. Se si include il valore INCLUDE_NULL come argomento opzionale della funzione Predict, si forza la funzione a ritornare il valore "Missing" come potenziale risultato.
Insieme ai risultati di predizione sono supportate funzioni che descrivono la predizione. La funzione PredictSupport() ritorna il numero di case in supporto alla predizione; PredictProbability() ritorna la verosimiglianza del valore stimato, fra tutti i valori possibili del dominio, in termini di probabilità.
Ad esempio, una selezione del tipo:

```
SELECT [ID Cliente], Predict(Età), PredictProbability([Età]) as P …
```

fornirà il seguente risultato:

ID Cliente	Età	P
10001	43	.667
10203	43	.400

Si prenda in osservazione l'attributo "Età". Se esso è discretizzato, il risultato sarà uno dei punti medi dei segmenti ottenuti dalla discretizzazione. Per ottenere maggiori informazioni sulla conformazione dei segmenti sono a disposizione delle funzioni specifiche, come RangeMin, RangeMax, RangeMid.
Se, invece, il modello è stato istruito con valori continui per "Età", anche il valore restituito dalla predizione sarà del medesimo tipo. Verosimilmente, l'età stimata sarà la media di una qualche distribuzione locale sulla base delle specifiche date per i case. Sebbene

frequentemente sia sufficiente tale valore, è possibile ricavare informazioni addizionali, come ad esempio la deviazione standard, calcolabile con la funzione PredictStDev.
Non si confonda tale funzione con la funzione SQL standard STD, quest'ultima infatti è una funzione di aggregazione, mentre PredictStDev restituisce uno scalare per ogni case della predizione.

Se il DMM supporta algoritmi di clustering, è possibile recuperare informazioni descriventi l'attività di raggruppamento, come l'appartenenza dei case ai vari cluster (quest'ultima informazione è data dalla funzione Cluster).
La lista completa delle funzioni disponibili, per ogni colonna di predizione, è riportata nello schema rowset MINING_COLUMNS. Per completezza di trattazione ne riportiamo alcune.

Funzione	**Valore restituito**	**Descrizione**
Predict(<riferimento colonna>, opzioni, …)	<riferimento colonna>	Funzione generica di predizione. Ritorna il miglior valore, date le opzioni. Può essere anche usata come argomento di PredictHistogram
PredictSupport(<riferimento colonna>**)**	Scalare	Conteggio dei case a supporto della predizione.
PredictVariance(<riferimento colonna >**)**	Scalare	Varianza della distribuzione i cui il valore stimato è la media.
PredictionStdev(<riferimento colonna >)	Scalare	Radice quadrata di **PredictVariance**.
PredictProbability(<riferimento colonna>)	Scalare	Probabilità che il valore stimato sia corretto.
PredictProbabilityVariance(<riferimento colonna >**)**	Scalare	Probabilità che il valore di**PredictVariance** sia corretto.
PredictProbabilityStdev(<riferimento colonna >**)**	Scalare	Radice quadrata di **PredictProbabilityVariance**.
Cluster	Scalare o <riferimento colonna cluster>	Identificatore del cluster a cui appartiene il case di ingresso con la probabilità più alta. Può essere anche usata come argomento di PredictHistogram
ClusterDistance([espressione Cluster]**)**	Scalare	Distanza del case di input dal centro del cluster identificato dalla espressione.
ClusterProbability([espressione Cluster])	Scalare	Probabilità che il case appartenga al cluster identificato dall'espressione.
RangeMid(<riferimento colonna >**)**	Scalare	Ritorna il punto medio del segmento per un attributo discretizzato.
RangeMin(<riferimento colonna >**)**	Scalare	Ritorna l'estremo inferiore del segmento.
RangeMax(<riferimento colonna >**)**	Scalare	Ritorna l'estremo superiore del segmento.
PredictHistogram(<riferimento colonna >**)**	Colonna tabella	Genera un istogramma che contiene le statistiche di predizione. L'argomento può essere anche una chiamata ad altre funzioni, come Predict o Cluster

Tabella 8: Funzioni built-in descritte nello schema rowset Mining Columns.

1.4.3.2 Espansioni di predizioni scalari: gli Istogrammi

Le informazioni supplementari a supporto della predizione non sono necessariamente semplici valori scalari. Un'altra possibilità per fornire predizioni è la generazione di una struttura, detta istogramma. Un istogramma è una tabella, ritornata dalla funzione

PredictHistogram(), che continene al suo interno, oltre al valore predetto, tutte le informazioni statistiche collaterali al processo predittivo. Essa ha una riga per ogni possibile valore ritornato per la colonna, al fianco dei valori statistici che ne descrivono la verosimiglianza. Il formato della colonna tabella ritornata è di tipo TABLE.

L'istogramma ha un insieme predefinito di statistiche contenute nelle colonne: valore stimato, supporto, varianza, deviazione standard, probabilità della predizione, probabilità della varianza e probabilità della deviazione standard. Lo schema completo dell'istogramma è riportato, al solito, nello schema rowset MINING_COLUMNS del DMM.

Di seguito vi è un esempio:

```
SELECT [ID Cliente], PredictHistogram([Sesso]) AS GH …
```

La tabella restituita è:

ID Cliente	GH

Sesso	$Support	$Probability	
M	621	.621	
F	379	.379	

10001

Sesso	$Support	$Probability	
M	446	.446	
F	554	.554	

10203

Alcune colonne sono state omesse per semplicità.

La funzione Predict seleziona il suo valore da questa struttura, scegliendo il record con il valore più alto di $Probability.

A seconda delle proprietà del DMM sottostante, la distribuzione di un attributo continuo può assumere più forme, il che significa che il grafico della distribuzione complessiva può avere più picchi. In questo caso è possibile ottenere le statiche relative ad ogni forma (cioè valori di picco locali) invocando la funzione PredictHistogram su una colonna continua.

```
SELECT [ID Cliente], PredictHistogram([Età]) AS AH …
```

ID Cliente	AH

Età	$StdDev	$Probability	…
32.1	17.2	.621	
65.2	6.4	.379	

10001

Se il DMM supporta il clustering, la funzione PredictHistogram() espanderà la predizione in una tabella descrivente l'appartenenza del case ad ogni cluster trovato. In questo caso, l'argomento della funzione PredictHistogram sarà la funzione Cluster().

Un istogramma, nel trattamento di cluster, consiste nelle seguenti colonne: Cluster, $Support, $Probability, $Distance.

```
SELECT [ID Cliente], PredictHistogram(Cluster()) AS CH …
```

ID Cliente	CH			
	Cluster()	$Support	$Probability	…
10001	1	724	.55	
	2	1025	.05	
	3	20	.40	

Se si vuole che il processo di costruzione dell'istogramma prenda in considerazione anche gli stati mancanti degli attributi, l'argomento passato alla funzione deve essere una chamata alla funzione Predict, dove sia specificato nelle opzioni il valore INCLUDE_NULL

1.4.3.3 Predizioni su colonne tabella

Una colonna TABLE può essere di predizione. Il risultato di un PREDICTION JOIN su una colonna di questo tipo è una tabella innestata con una riga per ogni valore della chiave della tabella innestata. In ogni riga comparirà il miglior valore stimato per ciascuna delle colonne predicabili. Selezionando direttamente la colonna tabella predicibile si ottiene il comportamento di default della funzione Predict.

Siccome la struttura restituita è una tabella vera e propria, possono essere adoperati dei SELECT innestati per selezionare le colonne di interesse.

```
SELECT [ID Cliente], [Sesso], (SELECT * FROM [Prodotti Acquistati]) …
```

ID Cliente	Sesso	Prodotti Acquistati		
		Nome Prodotto	Quantità	Tipo Prodotto
10001	M	TV	1	Elettronico
		Mais	2	Cibo
		Birra	6	Bibite
		Nome Prodotto	Quantità	Tipo Prodotto
10203	F	TV	2	Elettronico
		Mais	1	Cibo
		Birra	0	Bibite

La struttura dei case attuali può o non può contenere una tabella innestata avente lo stesso schema della tabella oggetto della predizione. Se no, l'interpretazione della predizione sulla

colonna tabella è naturale: viene predetta l'appartenenza ed i valori della tabella sulla base dei fattori dati per i case.

Se lo schema della tabella di input contiene una tabella innestata corrispondente, sono possibili tre varianti di comportamento:

1. la predizione consiste nella lista completa dei case della tabella innestata, sia con i dati di training che con i case attuali. Nel nostro caso ne risulta una tabella con tutti i prodotti acquistati e le quantità insieme con i valori stimati. Per eseguire tale implementazione, l'utente dovrà specificare il valore INCLUSIVE come opzione di Predict.
2. la predizione mostra solo i valori stimati, ovvero i prodotti che un cliente è più probabile che acquisti dati gli acquisti già fatti. I prodotti presenti nella tabella di input non sono riportati. Tale comportamento è identificato dal valore EXCLUSIVE.
3. la predizione può consistere nella sola stima della quantità dei prodotti acquistati, oppure della probabilità di ogni prodotto nei dati di input. Nessun altro prodotto apparirà nella tabella di output. L'opzione INPUT_ONLY assicura che lo schema risultante conterrà solo le righe dei dati di ingresso.

Ogni riga della tabella innestata di predizione possiede alcune informazioni statistiche associate di inclusione o appartenenza alla struttura. Queste sono differenti dalle probabilità e statistiche associate ad ogni valore delle colonne predicibili individuali. Infatti, le prime sono statistiche sulla sola esistenza del record nella tabella innestata. Per chiarire, un modello può mostrare che ci sia l'80% di probabilità che un cliente acquisti della birra e, di fianco, il 70% che le unità acquistate siano 10.

Un altro valore opzionale per Predict è INCLUDE_STATISTICS, che aggiunge ad ogni case della tabelle le colonne $Support e $Probability.

Ad esempio;

```
SELECT [ID Cliente], [Sesso],
      Predict([Prodotti Acquistati], INCLUDE_STATISTICS, INPUT_ONLY) …
```

ID Cliente | **Sesso** | **Prodotti Acquistati**

10001 | M

Nome Prodotto	Quantità	Tipo Prodotto	$Support	$Probability
Mais	2	Cibo	725	.267

10203 | F

Nome Prodotto	Quantità	Tipo Prodotto	$Support	$Probability
Mais	1	Cibo	30	.34
Birra	0	Bibite	56	.83

La colonna $Probability della tabella innestata contiene la probabilità di esistenza della particolare riga della sottotabella. Nessuna assunzione può essere derivata per le differenti probabilità di record diversi, in quanto esse sono dedotte da parti indipendenti del modello, ed unificarle non porterebbe a nessun risultato significativo.

Per ottenere informazioni sui differenti valori che compongono una predizione si può formulare una selezione più complessa facendo uso di istogrammi. In questo caso, la predizione include un istogramma per ogni valore distinto stimabile. Un esempio di tale formulazione è il seguente:

```
SELECT [ID Cliente], [Sesso],
      (SELECT [Nome Prodotto], PredictHistogram([Quantità]) AS [Istogramma Quantità]
                  FROM Predict([Prodotti Acquistati]), INCLUDE_STATISTICS) …
```

ID Clienti | **Sesso** | **Prodotti Acquistati**

Nome Prodotto	**Istogramma Quantità**			**$Probability**
TV	**Quantità**	**$Variance**	**$Probability**	0.23
	1	1.3	0.60	
	2	1.8	0.10	
	3	3.2	0.30	
Mais	**Quantità**	**$Variance**	**$Probability**	0.267
	1	0.5	0.25	
	2	0.7	0.55	
	3	3.7	0.20	
Birra	**Quantità**	**$Variance**	**$Probability**	0.832
	1	1.1	0.15	
	2	0.7	0.15	
	3	0.2	0.70	

10001 | M

Se una colonna TABLE supporta la funzione Predict è riportato nello schema rowset MINING_COLUMNS.

1.4.3.4 Operazioni su tabelle innestate

Una tabella innestata, risultante da una predizione, può contenere un elevato numero di record, in special modo quando si trattano grossi data warehouse aziendali. Il setacciamento dei risultati alla ricerca di nozioni e suggerimenti utili può essere un compito oneroso sia per l'applicazione consumer che per il provider. Anche quando una tabella innestata è costituita da un numero modesto di record, la ricerca di predizioni opportune può risultare scomoda.

Per agevolare questo tipo di processo, OLEDB for DM mette a disposizione 2 famiglie di funzioni, TopX e BottomX, esplicitamente dedicate al filtraggio e alla manipolazioni di strutture annidate.

Esse possono operare su qualsiasi tipo di tabella innestata, anche su quelle basate sugli istogrammi (ovvero restituite dalla funzione PredictHistogram), e rendono possibile l'ordinamento delle liste secondo specificati criteri (valori di colonna) ed il troncamento.

Ad esempio si può utilizzare la funzione TopCount per recuperare i 3 più probabili colori dei capelli:

```
SELECT [ID Cliente], TopCount(PredictHistogram([Colore Capelli]), $Probability, 3)…
```

o per ottenere la stima dei 10 prodotti che un cliente comprerà in maggiore quantità:

```
SELECT [ID Cliente], TopCount([Prodotti Acquistati], [Quantità], 10) …
```

Se una tabella innestata contiene un gran numero di colonne, dove solo poche delle quali sono di interesse per la predizione, come nel caso di uso delle funzioni PredictHistogram o Predict dove la maggior parte delle informazioni automatiche non sono desiderate, una selezione innestata può essere effettuata sulla tabella innestata o sulla funzione stessa.

Il seguente è un esempio di quest'ultima possibilità:

```
SELECT [ID Cliente], (SELECT Colore Capelli, $Support as Sup, FROM
      TopCount(PredictHistogram([Colore Capelli]), $Probability, 3)) as PH …
```

ID Cliente	PH
200	**Colore Capelli** / **Sup**: Rosso 100; Marrone 57; Nero 13
220	**HairColor** / **Sup**: Grigio 675; Nero 453; Biondo 2

Ancora, si supponga di volere un set di record stimati di una colonna tabella insieme ad un numero di statistiche a supporto per ogni record ritrovato. Nel paragrafo 4.4.9.3 è stato già fornito un esempio del genere, ma la struttura risultante può essere ingombrante e la sua navigazione difficoltosa oltre che non necessaria nel caso in cui si desiderino solo le migliori predizioni per l'attributo "Quantità" ed alcune misure sulla bontà della predizione recuperate dal relativo istogramma:

```
SELECT [ID Cliente], Sesso,
            (SELECT [Nome Prodotto], [Quantità] as [Migliore Quantità],
                        PredictStdev(Quantità) AS [Deviazione Quantità],
                        $Probabilty
                  FROM Predict([Prodotti Acquistati], INCLUDE_STATISTICS)), …
```

ID Cliente	Sesso	Prodotti Acquistati			
		Nome Prodotto	**Migliore Quantità**	**Deviazione Quantità**	**$Probability**
10001	M	TV	1	1.3	0.23
		Mais	2	0.7	0.267
		Gassosa	3	0.2	0.832

NOTA: Il SELECT innestato seleziona alcune colonne della tabella istogramma generata dalla chiamata della funzione `Predict([Prodotti Acquistati], INCLUDE_STATISTICS)` e la colonna $Probability riporta la probabilità che il record esista nell'insieme, non la probabilità sulla quantità stimata.

Un SELECT innestato, con una clausola WHERE, può essere utile nell'estrarre alcuni tipi di record. Ad esempio, invece che la migliore stima, si desidera la probabilità che un cliente sia femmina. In tal caso la query sarebbe del tipo:

```
SELECT [ID Cliente],
            (SELECT $Probability FROM PredictHistogram([Sesso])
                  WHERE Sesso = 'F')
                  AS [Probabilità Femmina] …
```

ID Cliente	Probabilità Femmina
10001	.379
10203	.554

Un altro possibile uso della clausola WHERE è nel limitare la predizione su alcuni tipi di record. L'esempio seguente mostra una predizione solo sull'acquisto di birre da parte dei clienti:

```
SELECT [ID Cliente], (SELECT * FROM [Prodotti Acquistati]
      WHERE [Nome Prodotto] = 'Birra') …
```

ID Cliente	Prodotti Acquistati		
	Nome Prodotto	**Quantità**	**Tipo Prodotto**
10001	Birra	6	Bibite

10203

Nome Prodotto	Quantità	Tipo Prodoto
Birra	0	Bibite

La stessa idea si può applicare per effettuare la predizione sul set di record definito dal valore di una colonna relazionata alla chiave della sottotabella:

```
SELECT [ID Cliente], (SELECT * FROM [Prodotti Acquistati]
      WHERE [Tipo Prodotto] = 'Bibite') …
```

La lista completa delle funzioni applicabili ad una colonna tabella predicibile sono riportate nello schema rowset MINING_COLUMNS nella documentazione dello standard. Riportiamo la sintassi e la descrizione solo di alcune di queste:

Funzione	Valore restituito	Descrizione
TopCount(<espr tabella>, <expr ord >, <n-items>)	<espr tabella>	Ritorna le prime n righe in ordine descrescente dell'espressione di ordinamento.
TopSum(<espr tabella>, <espr ord >, <somma>)	<espr tabella>	Ritorna le prime N righe, in ordine descrescente su <espr ord>, tali che la somma dei valori della colonna di ordinamento sia almeno <somma>.
TopPercent(<espr tabella>, <espr ord >, <percentuale>)	<espr tabella>	Ritorna le prime N righe, in ordine decrescente su <espr ord>, tali che la somma dei valori della colonna di ordinamento sia almeno la <percentuale> sulla somma totale dei valori.
Sub-SELECT: (SELECT <espressioni SELECT> FROM <espr tabella> [WHERE <clausola>])	<espr tabella>	Espressione di selezione annidata. <espr tabella> può essere sia un riferimento ad una calonna tabella che una funzione restituente una tabella, ma non un ulteriore sub-SELECT.

Tabella 9: Altre funzioni (famiglia TopX) riportate nello schema rowset.

1.4.3.5 Livellamento di tabelle innestate

La tabella innestata è una forma molto utile di rappresentazione dati che va incontro alle esigenze degli algoritmi di mining. Sfortunatamente, non sempre vi è supporto diffuso a questo tipo di rappresentazione. Il metodo per convertire semplici tabelle relazionali in strutture innestate è stato già discusso, introducendo lo statement SHAPE. Questo meccanismo aiuta a fornire dati al Provider.

Può accadere che alcune applicazioni consumer non abbiano la capacità di accettare dati in forme gerarchiche. Questo può essere causato dal mancato supporto alla gestione di gerarchie

o dalla necessità, da parte del consumer, di memorizzare i dati nei tradizionali schemi relazionali.

Da qui l'esigenza di convertire le strutture a livelli multipli in strutture "piatte". Per operare questa conversione vi è bisogno di richiedere che l'interrogazione sia "flattened", appiattita. A questo scopo, la sintassi di selezione fornisce l'opzione FLATTENED:

```
SELECT FLATTENED <espressione SELECT> FROM …
```

L'opzione converte la tabella gerarchica risultante dalla selezione in una tabella ad un livello. Il set risultante conterrà una riga per ogni valore stimato, semplificando il processamento dei risultati.

Se le colonne nella clausola <espressione SELECT> provengono da diversi livelli di gerarchie di tabelle innestate, la tabella piatta non disporrà le predizioni sulla medesima riga. Ad esempio, una predizione FLATTENED su "Prodotti Acquistati" può fornire il seguente risultato:

Customer ID	Product Name	Quantity	Probability
1	TV	1	.25
1	TV	2	.1
1	TV	3	.02
1	Mais	2	.2
1	Mais	1	.05
1	Mais	3	.03

In questo set, ogni riga contiene una singola predizione sui prodotti e le relative quantità.

Se, invece, la clausola <espressione SELECT> contiene colonne provenienti da più colonne TABLE, la struttura sarà piatta e suggerirà la forma gerarchica del risultato. Ogni riga conterrà sempre una singola istanza di predizione, ma righe diverse possono contenere tipi diversi di predizione. Chiariamo con un esempio: se la predizione riguarda il sesso ed i prodotti acquistati, il risultato potrebbe così apparire:

ID Cliente	Sesso	Probabilità Sesso	Nome Prodotto	Quantità	Probabilità Quantità
1	F	.43	Null	Null	Null
1	M	.57	Null	Null	Null
1	Null	Null	TV	1	.25
1	Null	Null	TV	2	.1
1	Null	Null	TV	3	.02
1	Null	Null	Mais	2	.2
1	Null	Null	Mais	1	.05
1	Null	Null	Mais	3	.03

Ogni riga contiene una predizione; righe diverse possono contenere predizioni di attributi diversi.

1.4.4 Cancellazione di modelli esistenti

Sono possibili due modi per effettuare la cancellazione di un modello. queste due operazioni sono simili alle operazioni di cancellazione di tabelle in SQL standard.

1. Cancellare l'oggetto DMM. Rimuovere l'oggetto significa cancellare dal sistema sia la sua struttura che il suo contenuto. Lo statement è il seguente:
   ```
   DROP MINING MODEL <nome modello>
   ```
   Effettuando questo comando il modello scomparirà dallo spazio dei nomi.
2. Cancellare il contenuto dell'oggetto. Pulire la struttura del suo contenuto informativo, ma lasciare inalterato lo schema. Lo statement è:
   ```
   DELETE FROM <nome modello>
   ```
   Il comando cancellerà il contenuto ed i valori di colonna, ma la struttura del modello persisterà e sarà ancora disponibile, ad esempio per un eventuale ripopolamento.
   Una variante del comando è la sintassi:
   ```
   DELETE FROM <nome modello>.CONTENT
   ```
   Questo comando permette la concellazione del solo contenuto del modello, lasciando intatti i valori di colonna appresi.

1.4.5 Raffinamento del modello

I modelli esistenti possono essere raffinati ulteriormente. Il raffinamento si riferisce alla modifica del contenuto o alla composizione di regole mediante l'inserimento di nuovi training set.

Il raffinamento del modello basato su case addizionali è possibile solo per alcuni algoritmi di mining, che possono essere aggiornati su base incrementale. Nello schema rowset MINING_SERVICES, alla colonna ALLOW_INCREMENTAL_INSERT è specificato se il provider fornisce questo vantaggio. Se supportato, il modello può essere raffinato semplicemente eseguendo altri comandi INSERT INTO con i case supplementari.

Se il raffinamento incrementale non è possibile, il modello deve essere prima pulito del contenuto e poi popolato con i case vecchi e nuovi.

4.5 Data Mining in MS SQL Server 200x

Un componente DM è incluso in Microsoft SQL Server (2000 e 2005), uno dei più popolari DBMS. Questo dà una spinta all'emersione delle tecniche DM nelle applicazioni più frequenti. A parte gli algoritmi sviluppati dal Research Center, il contributo maggiore contributo di SQL Server Data Minino è l'implementazione di OLEDB for DM.

Microsoft ha realizzato un provider OLEDB nativo per il Data Mining basato sulle specifiche dello standard. Il provider includeva in un primo momento 2 algoritmi per l'apprendimento (Microsoft Clustering e Microsoft Decision Tree) poi aumentati a 7 con la release del 2005 che include filtri bayesiani, reti neurali, regole di associazione, clustering di sequenze e serie temporali, insieme a vari strumenti per la visualizzaione e la modellazione avanzata dei modelli di mining.

Il provider per il DM è parte integrante degli strumenti di Analysis Services e, come OLAP Services, è espressamente indirizzato ai DBAs.

Non vi sono sofisticate componenti GUI per il DM, anche se Microsoft sta lavorando a stretto contatto con molti partner per lo sviluppo di applicazioni consumer generiche. Le uniche componenti GUI in Analysis Services includono un wizard per la creazione dei modelli, un editor di modelli, un browser per i contenuti ed il task DTS (Data Transformation Services: un set di tool che permette la manipolazione ed il trasferimento di dati fra sorgenti diverse). La fig. 49 riporta i maggiori componenti di Analysis Services 200x.

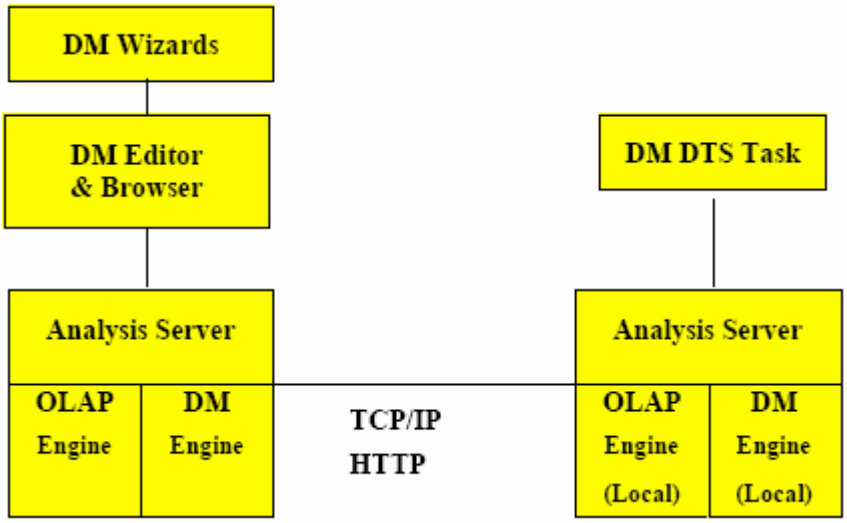


Figura 49: Disposizione dei i componenti Analysis Services in una coppia di nodi in rete.

Per iniziare lo sviluppo dell'applicazione si deve creare ed istruire il modello. Questo può essere fatto facilmente attraverso il wizard che instanzierà la struttura e provvederà alla generazione delle query SQL verso il provider via OLEDB.

Un altro metodo sicuramente più potente e flessibile è la scrittura di codice VB o C++ per connettersi al provider tramite ADO o le API DSO (Decision Support Objects) ed indirizzare il testo delle query come in una qualsiasi ambiente DBMS.

Analysis Serveices 2000 ha inoltre esteso il modello DSO per il DM. DSO è l'unico metodo per creare e gestire modelli persistenti sul server. I modelli creati tramite gli oggetti command di ADO hanno solo durata di sessione (*session mining models*). Inoltre con oggetti DSO è

possibile la creazione ed il settaggio di ruoli di sicurezza nell'ambiente. Per contro DSO richiede codice più complesso di ADO ed inoltre l'accesso è riservato agli amministratori.

Gli strumenti Microsoft per il Data Mining costituiscono un'alternativa all'approccio tradizionale al DM in vari modi.

Prima di tutto essi forniscono supporto all'intero ciclo di sviluppo di dati e relative applicazioni all'interno di un'organizzazione, al quale Microsoft si riferisce come ciclo IAR (*Integrate, Analyze and Report*). Questa possibilità libera i risultati di mining dalle competenza esclusive di pochi analisti esperti e li propone all'intero ambiente organizzativo.

Secondo, SQL Server 2005 Data Mining è una piattaforma per lo sviluppo intelligente di applicazioni, non un'applicazione *stand – alone*. Gli strumenti DM che spesso sono utilizzati dalle compagnie sono soluzioni mirate e spesso troppo legate al compito compito specifico. Esse realizzano un processo "*run and done*" sui dati, dopodiché i risultati sono osservati. Microsoft vuole proporre un'astrazione superiore dei propri strumenti svincolandoli dall'esecuzione di singoli task ed applicarli lungo tutto il processo, nel senso che i risultati raggiunti vengono immediatamente utilizzati come feedback per renderlo più accurato. Questo vantaggio di poter retroazionare i risultati nel processo consente all'organizzazione di dirigere più efficacemente le analisi successive o eliminare anomalie in fase di integrazione decidendo se i nuovi dati hanno senso alla luce degli altri dati. Ciò costituisce un vantaggio anche per il datawarehousing.

È possibile costruire la propria applicazione in modo che i modelli siano facilmente accessibili dal mondo esterno. I modelli sono estendibili cosicché terze parti possano effettuare il plug in dei propri algoritmi per specifiche esigenze.

Inoltre, gli algoritmi possono essere eseguiti in *tempo reale*, permettendo la validazione dei dati in linea. Ad esempio, l'immissione di dati da parte dell'utente tramite un web form sovente presenta varie incongruenze (come una combinazione di attributi "Età" = 7 e "Titolo" = laurea). Per risolvere tale problema si possono considerare i pattern validi dai dati propri dati ed assegnare, tramite il modello, un punteggio (scoring) a tali data entry che ne descriva il livello di confidenza. Se il punteggio è inferiore ad alcune soglie l'entry non viene accettata ed il form viene eventualmente riproposto. Questo esempio sottolinea il vantaggio che l'applicazione front – end può trarre dall'esecuzione in tempo reale di un motore DM.

Il DM, piuttosto che essere il mero risultato finale, diviene parte di un processo complessivo giocando un ruolo in ogni fase.

Un altro aspetto vantaggioso di un'unica piattaforma per la creazione di applicazioni intelligenti è la presenza di un server centralizzato per il mantenimento dei modelli e dei risultati. I modelli tendono ad essere altamente proprietari e ricchi di informazioni critiche e segrete per l'azienda. Un'unità server sicura protegge i modelli dalla diffusione nell'ambiente esterno. Altro beneficio implicito di una singola unità di memorizzazione è l'esitenza di una singola versione dei modelli e non di varianti multiple residenti sui singoli desktop.

4.5.1 I benefici di un processo integrato

INTEGRAZIONE

La fase di integrazione riguarda il reperimento dei dati da sorgenti disparate, la trasformazione ed il caricamento in altre unità. Gli strumenti stand – alone tradizionali non giocano alcun ruolo durante questa fase. All'interno dell'ambiente SQL Server gli algoritmi e le tecniche aiutano a scorgere i valori deviati che possono giacere nei dati o che possono essere stati inclusi durante la fase ETL tradizionale.

Gli strumenti DM sono integrati con i servizi offerti dai componenti di SQL Server Integration Services. Questo significa che, durante le operazioni di movimento e trasformazione, i dati possono essere analizzati e modificati sulle basi degli output dei modelli di mining.

ANALISI

Gli strumenti "run and done" generano i propri risultati al termine della costruzione dell'intero DW ed tali risultati sono analizzati indipendentemente dai risultati ottenuti sul warehouse (ad es. tramite strumenti OLAP).

I vantaggi del DM sono visibili sia in Analysis Services che in SQL Server. Grazie al modello universale di accesso ai dati UDM (Universal Data Model) le analisi possono essere effettuate sia su sorgenti relazionali che multidimensionali in maniera trasparente, e le funzionalità di mining forniscono una spinta al processo. Ad esempio, se alcuni clienti sono raggruppati in base ad alcune tendenze significative per l'analisi(es. le abitudini di navigazione) da un modello di clustering, tali risultati possono coadiuvare e suggerire lo *slice and drill* delle strutture dati sulla base dei cluster.

REPORTING

Una volta che la fase di modellazione è completa ed è stato creato un modello accurato, l'enfasi si rivolge alla traduzione delle analisi in risultati. Ciò è facilitato dall'inntegrazione fra DM e reporting in SQL Server 2005. Quest'ultima fase è realizzata dai componenti denominati SQL Server 2005 Reporting Services che permettono la consegna dei risultati significati in formati di facile comprensione.

Conclusioni

Con l'introduzione di uno standard specifico per le attività di Data Mining si muove un passo deciso verso la completa integrazione di questa tecnologia negli ambienti di basi di dati relazionali più familiari. Si superano le questioni legate all'incongruenza del formato dei dati dipendente dal produttore e si permette agli sviluppatori di applicazioni orientate ai database di far leva su tecnologie già stabili e robuste. Con uno sforzo relativo gli sviluppatori possono implementare altre tecniche algoritmiche all'interno di una piattaforma standard come SQL Server. Il motore relazionale viene mantenuto e migliorato con l'adattamento di tecniche specifiche in metafore relazionali. Tutti i servizi principali hanno un modello di estendibilità ben definito che consente agli sviluppatori di inserire propri componenti in qualsiasi punto del ciclo di vita dei dati nel sistema.

Il Data Mining diviene parte integrante dell'intero ciclo di sviluppo di applicazioni di Business Intelligence, integrazione dati, analisi dati e reporting.

BIBLIOGRAFIA

[1] AA.VV., 23 aprile 2000, *Comments on the Microsoft draft standard (specification) for data mining.*

[2] ACM SIGKDD, 11th International Conference on Knowledge Discovery and Data Mining (KDD), august 2005, Data Mining Standards, Services and Platforms, Chicago, Illinois.

[3] Atzeni P., Ceri S., Paraboschi S., Torlone R., 2002, *Basi di dati: modelli e linguaggi di interrogazione*. Mc Graw Hill.

[4] Atzeni P., Ceri S., Paraboschi S., Torlone R., 2002, *Basi di dati: architetture e linee di evoluzione*. Mc Graw Hill.

[5] Berson A., Smith S., Thearling K.,2001, *An overview of Data Mining Techniques*,from the book "Building Data Mining Application for CRM", Mc Graw Hill.

[6] Byard J., Schneider D.,1996, The *Ins & Ousts (and everything in between) of Data Warehousing*, Red Brick system Inc., www.redbrick.com.

[7] Chapman P. (NCR), Clinton J. (SPSS), Kerber R. (NCR), Khabaza T. (SPSS), Reinartz T. (DaimlerChrysler), Shearer C. (SPSS) e Wirth R. (DaimlerChrysler), august 2000, *CRISP-DM 1.0*, The CRISP-DM Consortium.

[8] Chapman P. (NCR), Clinton J. (SPSS), Kerber R. (NCR), Khabaza T. (SPSS), Reinartz T. (DaimlerChrysler), Shearer C. (SPSS) e Wirth R. (DaimlerChrysler), March 1999, *CRISP-DM Discussion Paper*.

[9] Eibe F., 2000, *Machine learning Techniques for Data Mining*, University of Waikato, New Zealand.

[10] Fayyad U., Piatetsky-Shapiro G., Smyth P., 1996, *From Data Mining to Knowledge Discovery in Databases*, American Association for Artificial Intelligence, www.aaai.org.

[11] Frawley J.W., Piatetsky-Shapiro G., Mathews C.J., 1991, *Knowledge Discovery in Database: an overview*. AAAI Press.

[12] Guha S., Rostogi R., Shim K., 1999,*CURE: An efficient Clustering Algorithm for Large Databases*, Stanford University.

[13] Grossman R.(University of Illinois at Chicago & Open Data Parters), Mark Hornik (Oracle Corporation), Meyer G. (IBM),2003, *Emerging Data Mining Standards and Interfaces*.

[14] Han J., Kamber M., 2000, *Data Mining: Concept and Techniques*. Morgan Kaufmann Publishers.

[15] Kluwer Academic, 1998, *Machine Learning*, Boston, Kluwer Academic Publishers, Manufactured in the Netherland, pp. 271-278.

[16] Lloyd-Williams M., 1997, *Discovering the hidden secrets in your data- the data mining approach to information*, Department of Information Studies, University of Sheffield Information Research, Vol 3, N° 2.

[17] Microsoft, 2000, *OLEDB for DM Specification*, www.micosoft.com/data/oledb/dm.

[18] Microsoft corporation, 2004, *SQL Server Data Mining*.

[19] Netezza Corporation, 2005, *Netezza Performance Server Appliance: An architectural Comparison*. www.netezza.com

[20] Nets A., Bernhardt J., Chaudhuri S., Fayyad U., 2004, *Integration of Data Mining and Reletional Databases*, Microsoft USA.

[21] Object Managment Group, february 2001, *Common Warehouse Metamodel (CWM) Specification*, version 1.0, www.omg.org.

[22] Parallel Computer Centre of the Queen's University of Belfast, 2002, *Data Mining Book*, mantained by Alan Rea.

[23] Riba S. D.,2003, *Extending The Data Warehouse: An Introduction to OLEDB*, Jade Tech Inc.,Clearwater FL, www.jadeteck.com.

[24] Rossini I., 2003, Metodologie per sistemi intelligenti- Costruzione di Modelli Previsionali, Politecnico di Milano, Polo Regionale di Como.

[25] Taft M., Krishnan R., Hornick M., Mukhin D., Tang G., Shiby T., 2003, *Oracle Data Mining: concept, 10g release 1*, Oracle Corporation.

[26] Tang Z., Kim P., 2000, *Building Data Mining Solution with SQL Server 2000*, Microsoft Corporation.

[27] Tang Z., Kim P., MacLennan J., 2005, *Building Data Mining Solution with OLEDB for DM and XML for Analysis*, Microsoft Corporation.

[28] Two Crows Corporation, 1999, *Introduction to Data Mining and Knowledge Discovery*, (Third ed.).

[29] Utley C., April 2005, Introduction to SQL Server 2005 Data Mining, www.microsoft.com/technet/

WEBGRAFIA

- www.iturls.com
- www.kdnuggets.com
- www.microsoft.com – the Microsoft Corporation Official Site.
- www.megaputer.com
- www.twocrows.com
- www.newarchitectmag.com
- www.spss.com
- www.cineca.it
- www.statsoft.com
- www.cs.cmu.edu
- www.apogeonline.com
- www.crisp-dm.org
- http://research.microsoft.com
- www.dmg.org - the Data Mining Group
- www.cio.com – the CIO Magazine Official Site
- http://www.datawarehousing.com

INDICE DELLE FIGURE

INDICE DELLE TABELLE