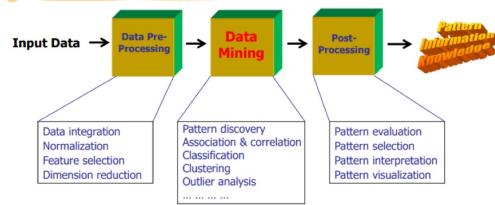


Data Mining e la conoscenza dei tuoi dati

Cos'è il Data Mining?

Data Mining è l'operazione di estrarre conoscenza dai dati.



Tipologie di Data Sets

Record	Graph and Network	Graph and Network	Spatial, image and multimedia
<ul style="list-style-type: none"> Relation records Data Matrix Document Data Transaction Data 	<ul style="list-style-type: none"> World Wide Web Social or information Networks Molecular Structures 	<ul style="list-style-type: none"> Video data: sequence of images Temporal Data: time-series Sequential Data: Transaction sequence Genetic sequence data 	<ul style="list-style-type: none"> Spatial data: maps Image data Video data

Descrizione statistica dei dati

1) Misura della tendenza centrale

Media

Media algebrica

$$\bar{x} = \frac{1}{N} \sum_{i=1}^N x_i \quad \text{where } N \text{ is sample size.}$$

Media algebrica pesata

$$\bar{x} = \frac{\sum_{i=1}^N w_i x_i}{\sum_{i=1}^N w_i}$$

Trimmed mean

Media ottenuta tagliando i valori agli estremi. Questo ci permette di rimuovere possibili outliers.

Mediana

Numero di valori dispari = Valore nel mezzo

Numero di valori pari = Media tra i valori nel mezzo

median = $I_1 + \frac{1}{2 - (\sum freq_{below}) / width}$
 Lower boundary of the median interval
 Sum of the frequencies of all the intervals that are lower than the median interval

Attenzione: La mediana è difficile da calcolare quindi si calcola una sua approssimazione

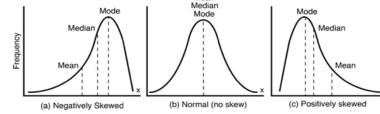
Modalità

Il mode è il valore che occorre più frequentemente nei dati.

I dati possono avere più valori frequenti e cambia il suo nome a seconda del numero: Unimodal, bimodal, trimodal.

$$mean - mode = 3 \times (mean - median)$$

Symmetric data and Skewed data



2) Misura della dispersione dei dati

kth percentile di un dataset > E' un valore x_k che ha la proprietà che il k percento dei dati è minore o uguale di x_k

- Quartili** = Le quartili sono VALORI che dividono un dataset in tre parti e rappresentano il
- 25th (Q_1) percentile = Il 25% dei valori è minore o uguale del valore Q_1
 - 50th (Q_2) percentile = Il 50% dei valori è minore o uguale del valore Q_2 = MEDIANA
 - 75th (Q_3) percentile = Il 75% dei valori è minore o uguale del valore Q_3

$$\text{Inter-quartile range (IQR)} = Q_3 - Q_1$$

Varianza = E' un valore algebrico che identifica la dispersione dei valori attorno al valore medio

$$\sigma^2 = \frac{1}{N} \sum_{i=1}^N (x_i - \bar{x})^2 = \frac{1}{N} \sum_{i=1}^N x_i^2 - \bar{x}^2$$

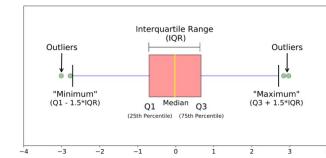
Deviazione standard = Essendo che la varianza è una quantità di secondo grado, si preferisce usare spesso la deviazione standard, essa misura sempre la dispersione dei valori attorno al valore medio

$$\sigma = \sqrt{\sigma^2}$$

↑
dev std. Varianza

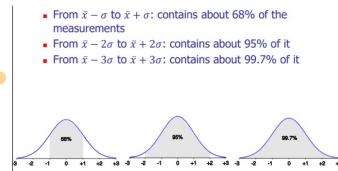
Boxplot Analysis

Rappresenta i dati con una box.



- From $\bar{x} - \sigma$ to $\bar{x} + \sigma$: contains about 68% of the measurements
- From $\bar{x} - 2\sigma$ to $\bar{x} + 2\sigma$: contains about 95% of it
- From $\bar{x} - 3\sigma$ to $\bar{x} + 3\sigma$: contains about 99.7% of it

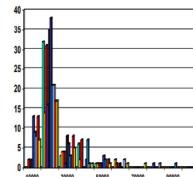
Proprietà della curva della distribuzione normale



3) Rappresentazione grafica della descrizione statistica dei dati

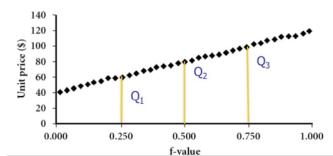
Histogrammi

Gli Histogrammi visualizzano la frequenza dei dati con delle barre.
 Nel caso di valori numerici essi vengono raggruppati in degli intervalli (bins) mentre nel caso di valori categorici viene mostrata la frequenza per ognuno di essi.



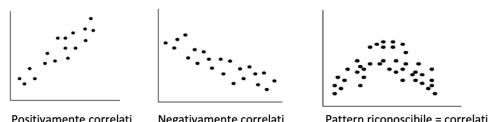
Quantile plots

I quantile plot visualizzano le informazioni riguardo i quantili sui dati.



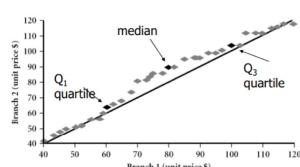
Scatter plot

Gli scatter plot sono grafici bivariati (utilizzano due attributi) e li mettono in correlazione tra di loro. Da uno scatter plot possiamo trovare delle correlazioni.



Quantile-Quantile plot

Nei quantile plots possiamo trovare anche il Quantile-Quantile plot il quale visualizza i quantili di una distribuzione univariata contro un'altra distribuzione univariata.



Similarità e Dissimilarità

Similarità = Misura numerica che indica quanto due data object sono simili.
 Dissimilarità = Misura numerica che indica quanto due data object sono differenti.

Viene solitamente usata la dissimilarità.

Matrice di dati = Ogni riga rappresenta un'istanza ed ogni colonna un attributo.

$$\begin{bmatrix} x_{11} & \dots & x_{1f} & \dots & x_{1p} \\ \dots & \dots & \dots & \dots & \dots \\ x_{i1} & \dots & x_{if} & \dots & x_{ip} \\ \dots & \dots & \dots & \dots & \dots \\ x_{n1} & \dots & x_{nf} & \dots & x_{np} \end{bmatrix}$$

istanza = oggetto

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

oggetto riga i
con oggetto riga j
oggetto riga i
con oggetto riga j

Matrice di dissimilarità (Dissimilarity matrix) = E' una matrice simmetrica che rappresenta quanto sono diversi gli oggetti tra loro. (due righe della matrice)

Si può rappresentare come matrice triangolare perché $d(2,1) = d(1,2), d(3,1) = d(1,3) \dots$

Dissimilarità per attributi nominali

Method 1: Simple matching

$$d(i,j) = \frac{p-m}{p}$$

Method 2: Use a large number of binary attributes
 - creating a new binary attribute for each of the M nominal states (for instance, for color, create binary attributes red, yellow, blue, green, and so on)

e confrontare
0 o 1

Dissimilarità per attributi numerici

Minkowski distance

$$d(i,j) = \sqrt[h]{|x_{i1} - x_{j1}|^h + |x_{i2} - x_{j2}|^h + \dots + |x_{ip} - x_{jp}|^h}$$

where $i = (x_{i1}, x_{i2}, \dots, x_{ip})$ and $j = (x_{j1}, x_{j2}, \dots, x_{jp})$ are two p -dimensional data objects, and h is the order (the distance so defined is also called L-h norm)

Inoltre, scegliendo specifici h otteniamo:

- $h=1$: Manhattan (city block, L₁ norm) distance
 E.g., the Hamming distance: the number of bits that are different between two binary vectors
- $h=2$: (L₂ norm) Euclidean distance
 $d(i,j) = \sqrt{|x_{i1} - x_{j1}|^2 + |x_{i2} - x_{j2}|^2 + \dots + |x_{ip} - x_{jp}|^2}$
- $h \rightarrow \infty$: supremum (L_{max} norm, L_∞ norm) distance.
 This is the maximum difference between any component (attribute) of the vectors

$$d(i,j) = \lim_{h \rightarrow \infty} \left(\sum_{f=1}^p |x_{if} - x_{jf}|^h \right)^{\frac{1}{h}} = \max_f |x_{if} - x_{jf}|$$

Dissimilarità per attributi di tipo diverso

Un dataset può contenere attributi di diverso tipo, per calcolare la dissimilarità tra i vari data object si può usare questa formula pesata:

$$d(i,j) = \frac{\sum_{f=1}^p \delta_{ij}^{(f)} d_{ij}^{(f)}}{\sum_{f=1}^p \delta_{ij}^{(f)}}$$

Vale 0 se:

- uno dei due obj non ha l'attributo f
- f vale 0 per entrambi nel caso asymmetric binary

where the indicator $\delta_{ij}^{(f)} = 0$ if either (1) x_{if} or x_{jf} is missing (i.e., there is no measurement of attribute f for object i or object j), or (2) $x_{if} = x_{jf} = 0$ and attribute f is asymmetric binary; otherwise, $\delta_{ij}^{(f)} = 1$. The contribution of attribute f to the dissimilarity between i and j (i.e., $d_{ij}^{(f)}$) is computed dependent on its type:

- If f is numeric: $d_{ij}^{(f)} = \frac{|x_{if} - x_{jf}|}{\max_{k,f} x_{kf} - \min_{k,f} x_{kf}}$, where h runs over all nonmissing objects for attribute f .
- If f is nominal or binary: $d_{ij}^{(f)} = 0$ if $x_{if} = x_{jf}$; otherwise, $d_{ij}^{(f)} = 1$.
- If f is ordinal: compute the ranks r_{if} and $z_{jf} = \frac{r_{jf}-1}{M_f-1}$, and treat z_{jf} as numeric.

$$\cos(d_p, d_q) = (d_p \bullet d_q) / ||d_p|| ||d_q||$$

Dove il "punto" indica il dot product

Data Preprocessing

Il processo di **Data Preprocessing** serve per andare a **sistemare i dati** per poter fare operazioni di **data mining** correttamente. Esso si divide in:

- 1) **Data Cleaning**
- 2) **Data Integration**
- 3) **Data Reduction**
- 4) **Data Transformation and Discretization**

Data Cleaning

I dati nel mondo reale possono essere sporchi: **Incompleti, rumorosi, inconsistenti o intensionali.**

1) **Dati incompleti**

I dati non sono sempre disponibili e bisogna gestire quando mancano. Bisogna **gestire i dati mancanti** perché se ci sono dei dati mancanti potrebbero essere presi dei dati non esatti e falsare le nostre operazioni.

- 1) **Ignorare le tuple** = ignorare quei data object incompleti. Viene solitamente fatto quando il class label è mancante. Non è effettivo se la % dei valori mancanti per attributo varia considerabilmente.
- 2) **Inserire i valori manualmente** = Non il miglior metodo, può essere tedioso e infattibile
- 3) **Inserire i valori automaticamente** = Possiamo andare a gestire la mancanza di valori andando a inserire in automatico dei valori. Possibilità:
 - a. Costante globale
 - b. Media dell'attributo
 - c. **Media dell'attributo per tutti i campionamenti che appartengono alla stessa classe** (è più intelligente rispetto al punto b.) -> Solitamente viene fatto questo
 - d. **Il valore più probabile ottenuto tramite inferenza di un modello**

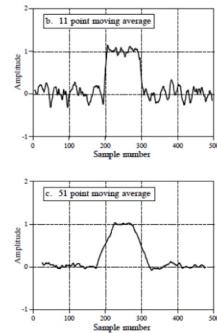
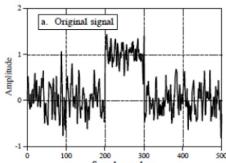
2) **Dati rumorosi**

I dati possono avere del **rumore** causato da un errore randomico o dalla varianza. Per andare a gestire il rumore si possono applicare diversi algoritmi.

- a) **Smoothing** = I dati points di un segnale sono modificati in modo che i punti che sono più vicini rispetto a quelli adiacenti vengono ridotti, e viceversa aumentati se più bassi.
Algoritmi di Smoothing

1) Rectangular or unweighted sliding-average smooth

Rimpiazza ogni punto del segnale con la media di m punti adiacenti, dove m è un intero positivo chiamato **smooth width**.



Filtered with 11 and 51 point moving average filters

Problema: Decidere m perché se troppo grande si ottiene un segnale piatto, se troppo piccolo si applica poco smoothing.

2) Triangular smooth

Implementa una funzione di smoothing pesata.

Per m = 5:

$$S_j = \frac{Y_{j-2} + 2Y_{j-1} + 3Y_j + 2Y_{j+1} + Y_{j+2}}{9}$$

Questo smooth è più effettivo nel ridurre high-frequency noise nel segnale. Inoltre è un 5-point triangular smooth corrisponde a due passi di un 3-point rectangular smooth.

Problema: Sempre il problema di decidere m.

Generalizzazione del rectangular filter

↑ usa un polinomio per filtrare i dati, cioè invece di usare una funzione lineare usa un polinomio per l'approssimazione.

Rimpiazza ogni data point Y_j con una combinazione lineare S_j se stesso è qualche numero vicino.

$$S_j = \sum_{n=-n_L}^{n_R} c_n Y_{j+n}$$

Dove:

- n_L è il numero di punti usati a sinistra della data point
- n_R è il numero di punti usati a destra della data point

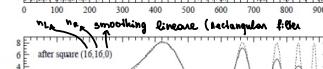
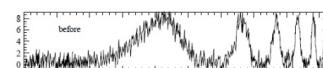
• c_n = filter coefficients (del polinomio di grado n)

usare $c_n = \frac{1}{n_L + n_R + 1}$
ottenere il rectangular filter!

L'idea è di muovere la finestra non di una costante ma con un polinomio di ordine più alto (tipicamente quadratico o quartico).

Fissato il grado del polinomio (M), n_L e n_R possiamo determinare il coefficiente offline.

Some example



Sliding-average smooth with window of 33 points



Savitzky-Golay of degree 4 with window of 33 points

Più è grande m, più si ottiene un riduzione del rumore, più il segnale viene distorto (Otteniamo dei picchi più bassi rispetto al segnale originale, perché la grandezza della finestra è più grande di quella del segnale). La scelta ottimale di smooth width dipende dalla grandezza e forma del segnale e l'intervallo digitalizzato.

Con il polinomio di grado 4 seguiamo di più il segnale, quindi approssimiamo meglio i picchi ma nella prima parte del segnale abbiamo una "modellizzazione" troppo adatta. (tipo overfitting).

smooth width / n° punti nella metà del picco

Se l'obiettivo è misurare il vero picco di altezza e grandezza, allora smooth ratios minori di 0.2 dovrebbero essere usati.

Se l'obiettivo è di misurare la posizione del picco (x-value) allora ratio più grandi possono essere usati perché smoothing non ha effetto sulla posizione del picco ma solo sulla sua altezza/grandezza.

When should you smooth a signal?

1. for cosmetic reasons, to prepare a nicer-looking graphic of a signal for visual inspection or publication;
2. if the signal will be subsequently processed by an algorithm that would be adversely affected by the presence of too much high-frequency noise in the signal, for example if the location of maxima, minima, or inflection points in the signal is to be automatically determined by detecting zero-crossings in derivatives of the signal.

When should NOT you smooth a signal?

1. Prior to statistical procedures such as least-squares curve fitting, because:
 - (a) smoothing will not significantly improve the accuracy of parameter measurement by least-squares measurements between separate independent signal samples;
 - (b) all smoothing algorithms are at least slightly "lossy", entailing some change in signal shape and amplitude,
 - (c) it is harder to evaluate the fit by inspecting the residuals if the data are smoothed, because smoothed noise may be mistaken for an actual signal, and
 - (d) smoothing the signal will seriously underestimate the parameters errors predicted by propagation-of-error calculations and the bootstrap method.

b) Binning = Smooth un valore ordinato consultando i suoi vicini

- a. Ordinare i dati e partizionarli in bins di ugual frequenza (ogni intervallo contiene lo stesso numero di valori)
- b. Scegliere il tipo di smooth che si preferisce tra bin means, bin median, bin boundaries etc.

Sorted data for price (in dollars): 4, 8, 15, 21, 21, 24, 25, 28, 34

```
Partition into (equal-frequency) bins:  
Bin 1: 4, 8, 15  
Bin 2: 21, 21, 24  
Bin 3: 25, 28, 34  
  
Smoothing by bin means:  
Bin 1: 9, 9  
Bin 2: 22, 22, 22  
Bin 3: 29, 29, 29  
  
Smoothing by bin boundaries:  
Bin 1: 4, 4, 15  
Bin 2: 21, 21, 24  
Bin 3: 25, 25, 34
```

c) Regressione = Smooth fissando i dati in una funzione di regressione

d) Clustering = Detectare e rimuovere gli outliers

e) Combinare ispezione combinata tra computer e umano

3) Rilevare la discrepanza nei dati

- 1) Usare metadati (dominio, range, dipendenza, distribuzione) -> Usati per rilevare in automatico possibili discrepanze nei dati.
- 2) Controllare le regole di:
 - a. Unicità = Ogni valore di un certo attributo deve essere differente da tutti gli altri valori
 - b. Consecutività = Non ci possono essere valori mancanti tra il valore più grande e il più piccolo
 - c. Null = Specifica l'uso del null value, question mark e altro
- 3) Usare tool commerciali
 - a. Data scrubbing = Usare la conoscenza del dominio per rilevare errori e fare correzioni
 - b. Data auditing = Analizzare i dati per scoprire regole e relazioni per rilevare violatori

4) Migrazione dei dati e integrazione

- 1) Tools per la migrazione dei dati = Permettono di specificare trasformazioni dei dati (come ad esempio rimpiazzamenti di parole)
- 2) ETL (Extraction/Transformation/Loading) tools: Permettono all'utente di specificare trasformazioni attraverso un'interfaccia grafica
 - a. Estrazione = I dati sono estratti dal sorgente in uno staging area.
 - b. Trasformazione = I dati estratti dal server sono grezzi e non utilabili. Devono essere puliti, mappati e trasformati. Possibili problemi:
 - a) Different spelling of the same person like Jon, John, etc.
 - b) There are multiple ways to denote company name like Google, Google Inc.
 - c) Use of different names like Cleveland, Cleveland.
 - d) There may be a case that different account numbers are generated by various applications for the same customer.
 - e) In some data required files remains blank
 - f) Invalid product collected at POS as manual entry can lead to mistakes.
 - c. Caricamento: Un grosso ammontare di dati deve essere caricato in poco tempo, il processo di caricamento deve essere ottimizzato. Tipi di caricamento:
 - i. Initial load = Popolare tutte le tabelle del data Warehouse
 - ii. Incremental load = Applicare i cambiamenti richiesti periodicamente man mano
 - iii. Full refresh = Eliminare il contesto di una o più tabelle e ricaricarlo con nuovi dati



Data integration

L'operazione di data integration consiste nell'andare a combinare i dati da sorgenti dati diverse in un unico store coerente, risolvendo i problemi di ridondanza, identificazione delle entità, conflitti sui valori delle entità.

Correlation Analysis

L'obiettivo della correlation analysis è andare a trovare le ridondanze e inconsistenze a livello di attributi ed andare a ridurre il loro numero per migliorare la qualità del mining (Course of Dimensionality).

Nominal Data (I valori che quel attributo può assumere sono limitati)

La correlation analysis per gli attributi nominali può essere fatta tramite il chi-square test (χ^2 test). I dati devono essere rappresentati in una Contingency Table:

	b_1	b_2	...	b_r	
a_1	$o_{1,1}$	$o_{1,2}$		$o_{1,c}$	$o_{1,r}$
a_2	$o_{2,1}$	$o_{2,2}$		$o_{2,c}$	$o_{2,r}$
...					
a_c	$o_{c,1}$	$o_{c,2}$		$o_{c,c}$	$o_{c,r}$
	$O_{1,1}$	$O_{1,2}$		$O_{1,r}$	Tot

Pertanto da questa tabella vogliamo verificare se A e B sono indipendenti.

Il chi-square test può essere calcolato nel seguente modo:

$$\chi^2 = \sum_{i=1}^c \sum_{j=1}^r \frac{(o_{ij} - e_{ij})^2}{e_{ij}} \quad \text{dove } e_{ij} = \frac{\text{count}(A=a_i) \times \text{count}(B=b_j)}{n}$$

e_{ij} rappresenta il numero di istanze che ci aspettiamo per quella combinazione i,j con l'assunzione che A e B siano indipendenti.

Quindi stiamo praticamente applicando la probabilità di A e B indipendenti.

$$A = \{a_1, a_2, a_3, a_4\}$$

$$B = \{b_1, b_2, b_3, b_4\}$$

Le $a_{i,j}$ rappresentano le volte che un'istante ha come valore a_i per l'attributo A, ha come valore b_j per l'attributo B.

$O_{1,1}$ e $O_{1,2}$ contengono il numero di istanze che contengono quello specifico valore di a_1 e b_1 rispettivamente.

Tot = Numero totale di istanze all'interno del dataset

$O_{1,1} = \text{numero di righe che hanno il valore } a_1 \text{ per l'attributo } A$

$O_{1,2} = \text{numero di righe che hanno il valore } b_1 \text{ per l'attributo } B$

Se $a_{i,j}$ e $e_{i,j}$ sono simili otteniamo che i due attr. sono indipendenti. Quindi più il chi-square è vicino a 0 e più i due attr. sono probabilmente indipendenti. Dobbiamo definire una soglia per definire che sotto quella soglia i due attr. sono indipendenti (una soglia di confidenza).

Come determiniamo la loro indipendenza? → chi-square test (Non parametrico, cioè non è necessario conoscere la forma della distribuzione della popolazione)

Null Hypothesis (H_0)	Afferma che non esiste associazione tra le due variabili nella popolazione e esse sono indipendenti tra loro
Alternative Hypothesis (H_1)	Propone che due variabili sono correlate nella popolazione
Degrees of Freedom	Indica il numero minimo di valori che devono essere fissati per completare la contingency table.

Vogliamo verificare la Null hypothesis tieni quindi le due variabili siano indipendenti. Se rejetiamo la null hypothesis abbiamo quindi che le due variabili sono dipendenti (alternative hypothesis).

Il Degree of Freedom indica quale chi-distribution andiamo a considerare. Le chi-distribution sono sempre skewed right.

Determinato il chi-square possiamo andare a determinare il valore dell'area, chiamato p-value, la quale indica la probabilità della null-hypothesis. Quindi maggiore è il chi-square value, minore è la probabilità che che la null-hypothesis tenga.



→ difficile calcolare l'area sotto la curva → utilizziamo la χ^2 -table

Numeric Data

Per determinare se due attributi sono correlati tra loro si calcola il correlation coefficient.

$$r_{A,B} = \frac{\sum_{i=1}^n (a_i - \bar{A})(b_i - \bar{B})}{n\sigma_A \sigma_B} = \frac{\sum_{i=1}^n (a_i b_i) - n\bar{A}\bar{B}}{n\sigma_A \sigma_B}$$

cosa dicono sono dipendenti!

Dove:

- n è il numero di tuple
- a_i e b_i sono rispettivamente i valori di A e B per la tupla i
- \bar{A} e \bar{B} sono rispettivamente la media di A e di B
- σ_A e σ_B sono rispettivamente le deviazioni standard di A e di B

Se:

- $r_{A,B} > 0$, A e B sono positivamente correlati, più grande è più forte la correlazione
- $r_{A,B} < 0$, A e B sono negativamente correlati, più grande è più forte la correlazione
- $r_{A,B} = 0$, A e B sono indipendenti

Attenzione: Non è detto che se $r_{A,B} = 0$ allora gli attributi sono indipendenti allora $r_{A,B} = 0$. Questo perché il correlation coefficient "rileva" solo le correlazioni lineari.

La correlazione tra gli attributi può essere vista visualmente usando lo scatter plot.

Per calcolare la correlazione facilmente: $d'_i = (a_i - \text{mean}(A)) / \text{std}(A)$
 $b'_i = (b_i - \text{mean}(B)) / \text{std}(B)$
 $\text{correlation}(A, B) = A' * B'$

Per calcolare la correlazione tra i due attributi si può usare anche la Covarianza

$$\text{Cov}(A, B) = E((A - \bar{A})(B - \bar{B})) = \frac{\sum_{i=1}^n (a_i - \bar{A})(b_i - \bar{B})}{n}$$

$$\text{Correlation coefficient: } r_{A,B} = \frac{\text{Cov}(A, B)}{\sigma_A \sigma_B}$$

Operazioni:

- Calcolo il grado di libertà
- Calcolo il chi-square
- Determino con che percentuale voglio rejecare la null hypothesis (con quale probabilità le due var. sono indipendenti tra loro)
- Se il valore che ottengo calcolando il chi-square è maggiore del valore nella tabella allora ho che le due variabili sono correlate tra loro, senno sono indipendenti. Questo perché se il valore del chi-square calcolato è maggiore, otteniamo che la prob. che tenga la null-hypothesis è minore.

Degrees of freedom	Probability of a larger value of χ^2
0	0.99
1	0.95
2	0.90
3	0.85
4	0.80
5	0.75

chi square valori spiegamento

Osservazione: Quando troviamo che due variabili sono correlate, non stiamo dicendo che una causa l'altra, ma stiamo dicendo che esse hanno lo "stesso trend", cioè se una sale l'altra sale, o se una sale l'altra scende ecc...

Data Reduction

1) Dimensionality Reduction

Ridurre la dimensionalià di un dataset ci permette di gestire il problema della curse of dimensionality, cioè il problema che con l'aumentare della dimensionalià i dati che si potevano reputare vicini, pian piano si allontano sempre di più. Inoltre ci vanno ad eliminare attributi irrilevanti, ridurre il rumore, avere la possibilità di visualizzare i dati.

Principal Component Analysis (PCA) - Approccio non-supervisionato

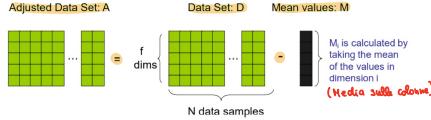
Trova una proiezione che cattura la più grande quantità di variazione nei dati. Questo perché un'alta variazione dei dati potrebbe significare una maggiore quantità di informazioni che possono essere ottenute.

I dati vengono allora proiettati in uno spazio molto più piccolo (anche se non è detto, il nuovo spazio potrebbe avere la stessa dimensionalià). Il nuovo spazio viene definito dagli autovettori della matrice di covarianza. L'errore della proiezione è minore del dataset originale. I nuovi punti proiettati sono più sparsi rispetto al dataset originale e quindi abbiamo più varianza.

Prima di iniziare il processo, tutti gli attributi devono essere normalizzati. Il PCA viene applicato solo su dati numerici.

Quindi, dati N vettori di dati composti da f dimensioni, trovare $k \leq f$ vettori ortogonali che possono essere usati per rappresentare i dati)

1) Calcolare il dataset "aggiustato" (Normalizzazione) → tutti gli attributi hanno lo stesso range



2) Calcolare la matrice di covarianza, C, dal dataset "aggiustato", A. Dato che la media delle dimensioni in A è 0, la matrice di covarianza può essere semplicemente scritta come:

$$C = (A A^T) / (n-1)$$

3) Calcolare gli autovettori e gli autovalori di C. Se qualche autovalore è 0 o molto piccolo, possiamo essenzialmente scarfarlo e anche gli autovettori associati, riducendo la dimensionalià della nuova base.



4) Trasformare il dataset nella nuova base

- where:
- F is the transformed data set
- E^T is the transpose of the E matrix containing the eigenvectors
- A is the adjusted data set

$$F = E^T A$$

Attribute subset Selection - Approccio supervisionato

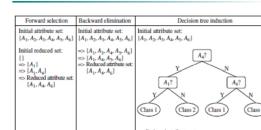
- Attributi ridondanti
- Attributi non rilevanti
- Ricerca euristica

Essendo che le combinazioni degli attributi sono 2^d dove d è il numero di attributi, devo usare dei greedy approaches che fanno la miglior scelta in un determinato momento, e non nel "totale" (Ottimo locale).

Quindi quando utilizziamo uno di questi metodi non possiamo assicurare la selezione migliore degli attributi, ma una selezione che si avvicina all'ottimo.

- Best single attribute under the attribute independence assumption: choose by significance tests
- Best step-wise feature selection:
 - The best single-attribute is picked first
 - Then next best attribute condition to the first, ...
- Step-wise attribute elimination:
 - Repeatedly eliminate the worst attribute
- Best combined attribute selection and elimination
- Optimal branch and bound:
 - Use attribute elimination and backtracking

Heuristic Search in Attribute Selection



Example of Heuristic Approach

Questo è un tipo di approccio supervisionato e lavora sullo spazio originale. Seleziona gli attributi sullo spazio originale e possiamo pensare di eliminare gli attributi non selezionati da esso perché probabilmente irrilevanti.

Intuizione:

- Vogliamo quindi selezionare gli attributi che sono molto correlati con l'attributo classe.
- Tra questi attributi che sono correlati con l'attributo classe, dobbiamo selezionare attributi che non siano tra loro correlati (perché potrebbero causare ridondanza).

Formula:

Per trasformare la nostra intuizione in formula, dati due attributi X e Y

Abbiamo bisogno in primis di definire la mutual information (1):

$$I(X, Y) = \sum_{x \in X} \sum_{y \in Y} p(x, y) \cdot \log \frac{p(x, y)}{p(x)p(y)}$$

Dove:

- $p(x, y)$ = Joint probability di x e y
- $p(x)$ = Probabilità di x
- $p(y)$ = Probabilità di y

Attenzione: Per recuperare A da F si può semplicemente eseguire queste operazioni

$$\begin{aligned} (\text{ET})^T \mathbf{F} &= (\text{ET})^T \text{ETA} \\ (\text{ET})^T \mathbf{F} &= \mathbf{A} \\ \mathbf{E}\mathbf{F} &= \mathbf{A} \end{aligned}$$

Osservazioni:

- 1) Essendo che il PCA è un metodo unsupervised, esso può essere utilizzato per un qualsiasi tipo di ML algorithm
- 2) Le nuovi dimensioni sono una combinazione lineare delle vecchie dimensioni, quindi non posso determinare tramite il PCA se un attributo è utile o meno nel mio dataset originale.

XX e YY contengono rispettivamente ogni valore di X e di Y rispettivamente.
Se X e Y sono indipendenti ottengo una mutual information vicina allo 0.

Successivamente dobbiamo definire la seconda parte dell'intuizione (2):

Dato un insieme iniziale F composto da n attributi, dobbiamo trovare un sottoinsieme S contenuto in F composto da k attributi che massimizzano la mutual information $I(C; S)$ dove C è la classe, e minimizzano la mutual information tra loro.

Quindi:

Normalized mutual information tra due attributi uno in F e uno già contenuto nel sottoinsieme S, f_i e f_s :

$$NI(f_i, f_s) = \frac{I(f_i; f_s)}{\min(H(f_i), H(f_s))}$$

Normalizziamo perché vogliamo compensare il mutual information bias che ci può essere tra i vari attributi e per restringere i valori nel range [0,1].

Viene usata l'entropia delle due variabili perché vogliamo ridurre l'effetto delle variabili con un alto numero di valori.

Dobbiamo selezionare le features da F che massimizzano la misura G, cioè feature che hanno un alta mutual information tra esse e la classe, ma una bassa mutual information tra essa e le restanti feature già selezionate (S):

$$G = I(C, f_i) - \frac{1}{S} \sum_{f_s \in S} NI(f_i; f_s)$$

Dove

$$I(C, f_i) = \sum_{c \in CC} \sum_{f_i \in FF} p(c, f_i) \cdot \log \frac{p(c, f_i)}{p(c)p(f_i)}$$

The algorithm

1. Initialization: Set $F = \{f_i | i = 1, \dots, N\}$, initial set of N features, and $S = \{\}$, empty set.
2. Calculate the MI with respect to the classes: calculate $I(f_i; C)$, for each pair $(f_i; f_j)$, with $f_i \in F$ and $f_j \in S$.
3. Select the first feature: Find $\hat{f}_1 = \max_{i=1, \dots, N} I(f_i; C)$. Set $F \leftarrow F \setminus \{\hat{f}_1\}$; set $S \leftarrow \{\hat{f}_1\}$.
4. Greedy Selection: Repeat until $|S| = k$
 - a. Calculate the MI between features: Calculate $I(f_i; f_j)$ for all pairs $(f_i; f_j)$, with $f_i \in F$ and $f_j \in S$.
 - b. Select next feature: Select feature $f_i \in F$ that maximizes G . Set $F \leftarrow F \setminus \{f_i\}$; set $S \leftarrow S \cup \{f_i\}$.
5. Output the set S containing the selected features.

Osservazione: Il numero di feature selezionate è impostato da noi prima che l'algoritmo di selezioni inizi.

Attribute Creation (Feature Generation)

Consiste nel creare nuovi attributi che catturano informazioni più importanti del dataset rispetto agli attributi originali:

Metodologie generali:

- 1) Estrazione degli attributi
- 2) Mappare i dati in un nuovo spazio
- 3) Costruire attributi combinando altri o facendo la Data discretization

2) Data Reduction

Ridurre il volume dei dati scegliendo forme alternative e più piccole per rappresentare i dati.

Questo ci permette di ridurre la complessità delle data analysis che potrebbero richiedere molto tempo per essere eseguite sull'intero dataset

- 1) Metodi parametrici = Assumere che i dati fittano un modello, stimare i parametri del modello, salvare i parametri e poi scaricare i dati
- 2) Regressione lineare = I dati sono modellati per fittare una linea retta, solitamente si usa l'errore least squares method per determinare quanto è accurata la nostra regressione lineare rispetto ai dati. Possiamo rimpiazzare i nostri dati con la linea retta trovata.

$$Y = w_1 X + w_0 \rightarrow \text{Retta}$$

I parametri sono w_0 e w_1 . Essi devono essere determinati con i dati.

$$\begin{aligned} \text{Size of the dataset} \\ \sum_{i=1}^{|D|} (x_i - \bar{x})(y_i - \bar{y}) \\ w_1 = \bar{y} - w_0 \cdot \bar{x} \\ \sum_{i=1}^{|D|} (x_i - \bar{x})^2 \end{aligned}$$

- 2) Regressione multipla lineare = Permette di modellare una var. di risposta Y come una funzione lineare di vettori di attributi multidimensionali

$$Y = w_0 + w_1 X_1 + w_2 X_2$$

- 3) Regressione polinomiale (non lineare):

$$Y = w_0 + w_1 X + w_2 X^2 + w_3 X^3$$

Per trasformarla in un'equazione lineare possiamo effettuare questa trasformazione:

E' ottenuta questa equazione:

$$Y = w_0 + w_1 X_1 + w_2 X_2 + w_3 X_3$$

- 4) Log-linear model = Approssima distribuzioni di probabilità multidimensionali discrete

- 2) Metodi non parametrici = Non assumono un modello

- 1) Histogrammi = Dividere i dati in buckets e salvare solo la media per ognuno di esso. I bucket possono essere divisi seguendo due regole:
 - a. Equal-Width = Il range del bucket è uguale per tutti
 - b. Equal-Frequency = Il numero di valori nei bucket è uguale per tutti

- 2) Clustering = Partizionare i dati in cluster basati sulla similarità e salvare solo il rappresentante del cluster (ad esempio: centroidi o diametro)

- 3) Sampling (Approccio più usato) = Ottenere un piccolo campionamento s che rappresenta l'intero dataset N. Questo permette di eseguire gli algoritmi in una complessità sub-linear nella grandezza dei dati. La chiave per fare ciò è trovare un sottoinsieme rappresentativo dei dati.

- Tipi di campionamento:
- a. Campionamento randomico = Scelgono dei valori a caso (tutti hanno la stessa prob.)
 - b. Campionamento senza rimpiazimento = Quando un valore viene scelto esso viene rimosso dalla pull dei valori disponibili
 - c. Campionamento con rimpiazimento (Non cambio la % per ogni estrazione) = Quando un valore viene scelto esso non viene rimosso dalla pull dei valori disponibili
 - d. Campionamento stratificato = Partizionare il dataset e pescare campionamenti da ogni partizione, viene usato molto nei task di classificazione perché voglio andare a preservare la distribuzione delle due classi, in modo che non venga "samplerata" solo la majority class.

- 4) Data Cube Aggregation = Il datacube è la rappresentazione che viene solitamente usata dalle data warehouse per rappresentare i dati. Andando a generalizzare o meno come andiamo a rappresentare i dati, possiamo ottenere una riduzione di essi.

Ad esempio andando a rappresentare la data invece che per mesi, in anni.

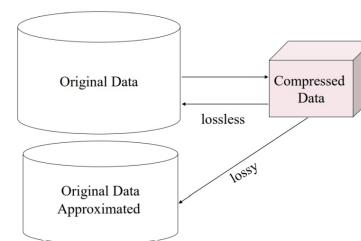


3) Data Compression

- String compression
 - There are extensive theories and well-tuned algorithms
 - Typically lossless, but only limited manipulation is possible without expansion
- Audio/video compression
 - Typically lossy compression, with progressive refinement
 - Sometimes small fragments of signal can be reconstructed without reconstructing the whole
- Time sequence is not audio
 - Typically short and vary slowly with time
- Dimensionality and numerosity reduction may also be considered as forms of data compression

Le tecniche di data compression si dividono in due tipologie:

- 1) Lossless: Comprimiamo i dati in modo che quando li decomprimiamo riotteniamo i dati originali, non abbiamo perdite di informazione. (esempio: JPEG).
 - 2) Lossy: Comprimiamo i dati però quando li decomprimiamo non riotteniamo i dati originali, abbiamo delle perdite di informazione. (esempio: JPEG).
- Quando usiamo i lossy otteniamo un'altra compressione e perdiamo dei dati che possono essere ignorati, questo è il motivo per cui a volte si utilizzano le Lossy compression.



Data Transformation and Discretization

E' una funzione che mappa l'intero insieme di valori di un dato attributo a un nuovo insieme di valori di rimpiazzamento tali che ogni vecchio valore può essere identificato con uno dei nuovi valori. Oltre alle tecniche di trasformazione già viste (Smoothing, Attribute construction, Aggregation), abbiamo anche la normalization e discretization.

Normalizzazione (Conviene provare entrambe le alternative, sia con normalization che senza quando si usano i modelli di ML)
 (Ottenerne gli attributi tutti in uno stesso range di valori per poter utilizzare tecniche specifiche, ad esempio la distanza euclidea perché i valori grandi vanno a "dominare" la computazione rispetto ai piccoli.)

- 1) **Min-Max normalization:** Scalare i valori in un certo intervallo [min,max]

$$v' = \frac{v - \min}{\max - \min} \cdot (\text{new_max} - \text{new_min}) + \text{new_min}$$

Problema: Se c'è un outlier nel dataset tutti i dati vengono compresi in una zona vicina all'outlier.

- 2) **Z-score normalization (meglio):** Scalare i valori utilizzando la media e la deviazione standard (non sono in un range). Gli attributi vengono comunque mappati in un range simile perché usiamo come denominatore la deviazione standard la quale dipende dalla sparsità dei dati. Se i dati sono molto sparsi la deviazione standard è grande, viceversa è piccola. Quindi otteniamo che i dati varieranno approssimativamente nello stesso range.

$$v' = \frac{v - \mu}{\sigma}$$

- 3) **Decimal scaling (non popolare)** $v' = \frac{v}{10^j}$ Dove j è il più piccolo intero tale che $\text{Max}(|v'|) < 1$

Discretization = Dividere il range di un attributo continuo in intervalli (Da numerici a categorici), per potere utilizzare algoritmi specifici che lavorano solo su attributi categorici. (Discretization può influire sui risultati dei modelli). Esistono tecniche supervised e unsupervised.

- 1) **Binning** = Dividere i dati in bins, problema: I dati di diverse classi vengono divisi negli stessi bins.

- a. **Equal-width partitioning:**

- i. Divide il range in N intervalli di ugual grandezza
- ii. Se A è il minimo e B il massimo, la grandezza degli intervalli è $W = \frac{B-A}{N}$
- iii. Gli outliers potrebbero "dominare" la presentazione.
- iv. I dati distorti non sono ben gestiti

- b. **Equal-depth partitioning:**

- i. Divide il range in N intervalli, ognuno contenente approssimativamente lo stesso numero di valori
- ii. È un buon data scaling
- iii. Gestire gli attributi categorici può essere difficile

- 2) **Histogram analysis:** Appuccio top-down, unsupervised

- 3) **Clustering Analysis:** Appuccio top-down o bottom-up, unsupervised

- 4) **Classificazione (Decision tree analysis):** Supervised, usa l'entropia per determinare i punti di splitting. L'esecuzione è top down con split ricorsivi

- 5) **Correlation Analysis (chi-square):** Supervised, trova i migliori intervalli vicini da unire. Il merge è bottom-up ricorsivo (da intervalli piccoli si ottengono intervalli più grandi)

Esempio

Sample	F	K
1	1	1
2	3	2
3	7	1
4	8	1
5	9	1
6	11	2
7	23	2
8	37	1
9	39	2
10	45	1
11	46	1
12	59	1

- Statistical approach to Data Discretization
- Applies the Chi Square method to determine the probability of similarity of data between two intervals.
- F-> feature
- K-> class

Sample	F	K	Intervals
1	1	1	{0,2}
2	3	2	{2,5}
3	7	1	{5,7,5}
4	8	1	{7,5,8,5}
5	9	1	{8,5,10}
6	11	2	{10,17}
7	23	2	{17,30}
8	37	1	{30,38}
9	39	2	{38,42}
10	45	1	{42,45,5}
11	46	1	{45,5,52}
12	59	1	{52,60}

Sample	F	K	Intervals
1	1	1	{0,2}
2	3	2	{2,5}
3	7	1	{5,7,5}
4	8	1	{7,5,8,5}
5	9	1	{8,5,10}
6	11	2	{10,17}
7	23	2	{17,30}
8	37	1	{30,38}
9	39	2	{38,42}
10	45	1	{42,45,5}
11	46	1	{45,5,52}
12	59	1	{52,60}

4

Sample	K=1	K=2
2	0	1
3	1	0
total	1	2

$$\begin{aligned} E_{11} &= (1/2)*1 = .5 \\ E_{12} &= (1/2)*1 = .5 \\ E_{21} &= (1/2)*1 = .5 \\ E_{22} &= (1/2)*1 = .5 \end{aligned}$$

$$X^2 = (0.5)^2/1.5 + (0.5)^2/1.5 + (0.5)^2/1.5 + (0.5)^2/1.5 = 2$$

Sample	K=1	K=2
3	1	0
4	1	0
total	2	0

$$E_{11} = (1/2)*2 = 1$$

$$E_{12} = (0/2)*2 = 0$$

$$E_{21} = (1/2)*2 = 1$$

$$E_{22} = (0/2)*2 = 0$$

$$X^2 = (1-1)^2/1 + (0-0)^2/0 + (1-1)^2/1 + (0-0)^2/0 = 0$$

Threshold .1 with df=1 from Chi square distribution chart
 merge if $X^2 < 2.7024$

5

Sample	F	K	Intervals	Chi ²
1	1	1	{0,2}	2
2	3	2	{2,5}	2
3	7	1	{5,7,5}	0
4	8	1	{7,5,8,5}	0
5	9	1	{8,5,10}	0
6	11	2	{10,17}	2
7	23	2	{17,30}	0
8	37	1	{30,38}	2
9	39	2	{38,42}	2
10	45	1	{42,45,5}	2
11	46	1	{45,5,52}	0
12	59	1	{52,60}	0

e il chi-square < 2.7024

6

Sample	F	K	Intervals	Chi ²
1	1	1	{0,2}	2
2	3	2	{2,5}	4
3	7	1	{5,7,5}	5
4	8	1	{7,5,8,5}	5
5	9	1	{8,5,10}	5
6	11	2	{10,17}	2
7	23	2	{17,30}	2
8	37	1	{30,38}	3
9	39	2	{38,42}	2
10	45	1	{42,45,5}	4
11	46	1	{45,5,52}	4
12	59	1	{52,60}	1

Repeat

7

Sample	F	K	Intervals	Chi ²
1	1	1	{0,5}	1.875
2	3	2	{5,10}	5
3	7	1	{10,30}	1.33
4	8	1	{30,42}	1.875
5	9	1	{42,60}	1.875

Sample	F	K	Intervals	Chi ²
1	1	1	{0,5}	1.875
2	3	2	{5,10}	1.875
3	7	1	{10,30}	3.93
4	8	1	{30,42}	3.93
5	9	1	{42,60}	3.93

Again

Until

9

Sample	F	K	Intervals	Chi ²
1	1	1	{0,10}	2.72
2	3	2	{10,30}	3.93
3	7	1	{30,42}	3.93
4	8	1	{42,60}	3.93
5	9	1	{0,10}	2.72

There are no more intervals that can satisfy the Chi Square test.

Che hanno il chi-square < di 2.7024

Frequent pattern Analysis

Cos'è il Frequent Pattern Analysis?

Esso consiste nel determinare dei pattern che occorrono frequentemente in un dataset e potrebbero determinare una correlazione tra data objects.

Terminologia

Itemset I	Un insieme di uno o più items, rappresenta il totale degli item
k-itemset X	insieme composto da k elementi di I
Absolute support	Occorrenza di un k-itemset X nelle transazioni
Relative support	Percentuale di transazioni che contiene un k-itemset X rispetto al totale di transizioni

→ NUMERO
→ percentuale

Un itemset X è frequente se il suo relative support non è minore di una certa soglia minsup

Transazione(T) Insieme di items tale che T è contenuto in I

Ogni transazione può essere rappresentata da un vettore booleano che rappresenta se contiene o meno un oggetto

Regola di associazione	E' un'implicazione $X \Rightarrow Y$, dove
	• X contenuto in I
	• Y contenuto in I
	• X intersezione $Y = \text{insieme vuoto}$

Indica che la presenza di X → es: $\{\text{Bread}\} \Rightarrow \{\text{coffee}\}$ indica che se in una transaction è presente "Bread" allora

è probabile che sia presente anche "coffee"

OSSERVAZIONE: Ogni transazione può essere rappresentata da un vettore booleano che identifica la presenza o meno di uno specifico item

Supporto e Confidenza di una regola di associazione

Supporto($X \Rightarrow Y$) = $P(X \cup Y)$

Probabilità che una transazione contiene sia X che Y = Riflette utilità

Riflette utilità perché ci dice il numero di transizioni che contengono entrambi gli item sul totale delle transizioni.

Confidence ($X \Rightarrow Y$) = $P(Y | X) = \text{supp}(X \cup Y) / \text{supp}(X)$

Probabilità condizionale che la transazione che ha anche Y = Riflette certezza

Riflette certezza perché ci dice il numero di transizioni che contengono Y sapendo che contengono X . Cioè dato un certo item X in quante di quelle transizioni che contengono X è stato comprato anche Y .

Attenzione: Se Y è sempre presente nelle transazioni e $\text{confidence}(X \Rightarrow Y) = 1$ non otteniamo che $X \Rightarrow Y$ perché non è possibile dedurlo.

Come trovare pattern frequenti e creare le regole di associazione?

- 1) Trovare tutti gli itemset frequenti cioè fissare un min_supp ed estrarre gli itemset che rispettano quel min_supp .
- 2) Generare forti regole di associazione dagli itemset frequenti, trovati nel punto 1, che rispettano il min_conf .

Problema: Un itemset grosso può contenere numero combinatorio di sub-itemset veramente grande

Soluzione: Minare closed itemsets e max-itemsets

Closed itemset Un itemset X è chiuso in un dataset D se X è frequente e non esiste un super-itemset Y ("più grande") che lo contiene, con lo stesso supporto di X .

Max-itemset Un itemset X è un max-itemset in un dataset D se X è frequente e non esiste un super-itemset Y ("più grande") che lo contiene.

Exercise.

DB = { a_1, \dots, a_{100} , $< a_1, \dots, a_{50} >$ }
 $\text{Min_sup} = 1$.

What is the set of closed itemset?

$< a_1, \dots, a_{100} >: 1$

$< a_1, \dots, a_{50} >: 2$

What is the set of max-itemset?

$< a_1, \dots, a_{100} >: 1$

Importante = I closed itemset sono una compressione senza perdita di informazione dei pattern frequenti riducendo il numero di pattern e regole

Complessità computazionale del minare gli itemset frequenti

Il numero di itemset frequenti generati dipende dalla soglia di minsup . Quando minsup è basso, esistono potenzialmente un numero esponenziale di itemset frequenti.

Nel caso peggiore vengono generati M^N itemset dove:

- $M = \# \text{ item distinti}$
- $N = \text{lunghezza massima delle transizioni}$

Quando il numero di item aumenta, la probabilità di un frequent pattern diminuisce

con il max-itemset perdiamo l'info che $\{a_2, \dots, a_{50}\}$ ha supporto 2, assumendo che abbia supporto 1!

APRIORI ALGORITHM

Proprietà di Apriori = Ogni sottoinsieme di item di un itemset frequente, è sicuramente frequente

Apriori pruning principle = Se c'è un qualsiasi itemset che non è frequente, tutti i suoi superset non dovrebbero essere generati e testati.

Algoritmo di Apriori

In ciclo generiamo itemset sempre più lunghi e controlliamo se sono frequenti. Terminiamo quando non abbiamo più itemset da generare.

- 1) Scannerizzare il DB per ottenere gli 1-itemset frequenti

Ciclo:

- 2) Generare gli itemset di lunghezza $(k+1)$ dagli itemset frequenti di lunghezza k
- 3) Testare gli itemset generati
- 4) Terminare quando non ci sono itemset frequenti o non possono essere generati altri itemset

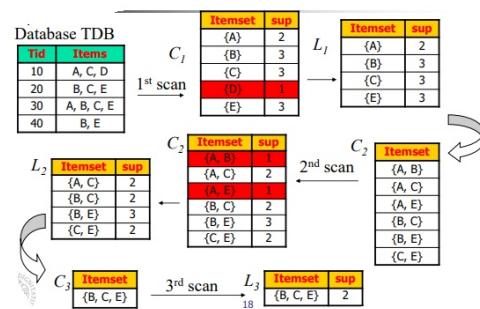
L'algoritmo termina sicuramente perché con l'aumentare della lunghezza diminuisce il supporto degli X_k . Bisogna scegliere un min_supp che sia interessante per il problema che vogliamo analizzare.

Esempio dell'algo con:

L = Insieme di itemset frequenti

C = Insieme di itemset generati da controllare

$\text{Min_supp} = 2$



Join step = Per trovare L_k , un insieme di k -itemsets candidati, è generato unendo L_{k-1} con se stesso.

- 1) Gli itemset sono ordinati in ordine lessicografico.
- 2) La join è fatta in modo che se sto generando un k -itemset (k di elementi), joino gli itemset che hanno i primi $k-1$ elementi uguali.

Esempio ($k=3$):

Se sto generando gli itemset di lunghezza 3 prendendo gli itemset di lunghezza 2, unisco quegli itemset che hanno il primo item uguale (perché $k - 2 = 1$)

Prune step = Dopo la generazione devo verificare:

- 1) Se ho degli itemset che devo rimuovere perché non rispettano il supporto.
- 2) Se gli itemset hanno dei sub-itemset che non rispettano il supporto, questo perché i sub-itemset non frequenti non possono generare itemset frequenti per la proprietà di Apriori.

Generazione Association Rules : Dagli itemset posso generare le association rules nel seguente modo.

- 1) Per ogni itemset frequente, genero tutti i subset non vuoti di esso.
- 2) Per ogni subset (S) non vuoto, se $\frac{\text{support(itemset)}}{\text{support}(S)} \geq \text{min-confidence}$

do in output la regola:

$$S \Rightarrow (Itemset - S)$$

OSSERVAZIONI:

- 1) La scelta del min_supp influenza i risultati dell'algoritmo:
 - Complessità
 - Numero di frequent pattern minimi
 - Basso: Molti frequent pattern però non riusciamo ad individuare l'importanza di essi -> Generazione di molti candidate set e la complessità aumenta.
 - Alto: Riduciamo il numero di frequent pattern ma rischiamo di perdere dei pattern importanti -> Generazione di pochi candidate set, la complessità diminuisce.
- 2) Il dataset necessita di essere scansionato ogni volta che generiamo i candidati e verifichiamo se sono frequenti

POSSIBILI MIGLIORAMENTI DELL'APRIORI ALGORITHM MA NON TROPPO EFFETTIVI

- 1) Dividere il DB in n partizioni, trovare i frequent pattern su essi in parallelo e infine unire i frequent pattern trovati
- 2) Usare un hash table per gli itemset, se scopriamo che il supporto di un certo bucket è minore del min_supp otteniamo che allora i pattern all'interno di quel bucket non sono frequenti
- 3) Usare un sample del database
- 4) Anticipare lo scanning del DB anche a pattern di lunghezza maggiore rispetto alla lunghezza che stiamo analizzando

Problema dell'algoritmo Apriori

Genera i candidati e poi li testa. Generare i candidati è computazionalmente costoso. Contare il supporto è costoso perché richiede più accessi al DB.

FPGrowth Approach

L'FPGrowth Approach si divide principalmente in due passi:

- 1) Generare una struttura compressa chiamata FP-tree
- 2) Estrarre gli itemset frequenti direttamente dall'FP-tree

Come funziona l'algoritmo:

- 1) Creare un FP-tree compatto in modo da riassumere il DB senza perdere info
- 2) Dividere l'FP-tree in un insieme di DB condizionali, ognuno associato con un item frequente o pattern fragment e minare ogni database separatamente.
- 3) Per ogni pattern fragment, solo i dataset associati necessitano di essere esaminati

Attenzione: Se il db è grande, si divide in partizioni, si costruisce una FP-tree per ogni partizione

Vantaggi del FP-Growth

- Solo due passaggi sul dataset (fatti per costruire l'FP-tree)
 - Trovare gli items frequenti
 - Generare l'FP-tree
- Comprime il dataset
- Nessuna generazione dei candidati
- Più veloce di Apriori

Disadvantages of FP-Growth

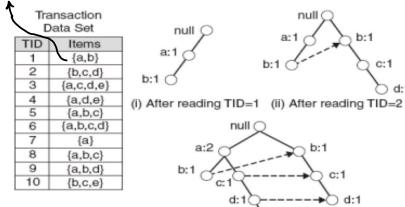
- FP-Tree potrebbe non fitare in memoria
- FP-Tree è costoso da costruire

Ma se fitta in memoria, è super efficiente

Passo 1 (Prima della generazione)

- 1) Scansionare i dati e trovare il supporto per ogni item
- 2) Scartare gli items non frequenti
- 3) Ordinare gli oggetti all'interno degli items frequenti, in ordine decrescente basato sul supporto (così i prefissi comuni possono essere condivisi)

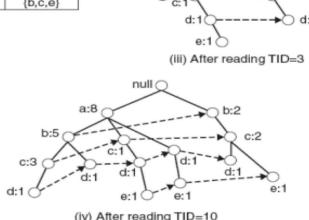
gli items più frequenti sono nel seguente ordine: a,b,c,d,e e gli items sono ordinati in base all'ordine basato sul loro supporto



Passo 2 (Generazione del FP-tree)

I nodi corrispondono agli item e hanno un contatore

- 1) FP-Growth legge una transazione alla volta e la mappa in un path del FP-tree
- 2) Un ordine fissato è usato, così i paths si possono sovrapporre quando le transazioni condividono gli items, cioè quando hanno lo stesso prefisso (in questo caso, i contatori vengono aumentati)
- 3) I puntatori sono mantenuti tra i nodi contenenti lo stesso item, creando delle linked list (dotted lines). Più path sono sovrapposti, più l'FP è compresso e potrebbe fittare in memoria (fondamentale).



Osservazioni del FP-tree generato:

- 1) L'FP-tree solitamente ha una dimensione più piccola perché molte transazioni condividono gli stessi items.
 - Miglior caso: Tutte le transazioni contengono lo stesso insieme di items
 - Peggior caso: Ogni transazione ha un insieme unico di items
- 2) La grandezza del FP-tree dipende da come gli items sono ordinati, ordinare gli item per supporto decrescente non sempre porta al FP-tree più piccolo (è euristico).
- 3) Come parametro usiamo il min_supp, è necessario fissarlo.
 - min_supp alto: Riduciamo il numero di item e ci aspettiamo un FP-tree più compresso
 - min_supp basso: Aumentiamo il numero di item e ci aspettiamo un FP-tree meno compresso

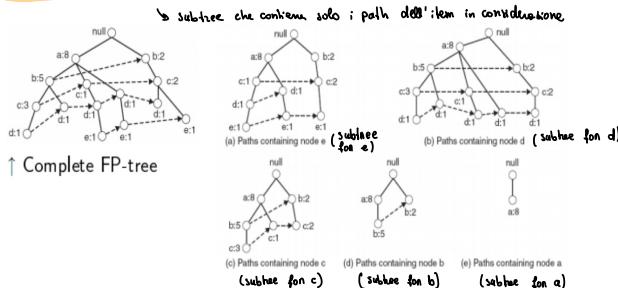
In ogni caso il min_supp non influenza così tanto come nel caso dell'Apriori sulla complessità.

Passo 3 (Generazione degli Itemset Frequenti)

FP-Growth estrae i frequent itemset dal FP-Tree generato nel passo 2 con un approccio divide and conquer (prima controlla tutti i frequent itemsets che finiscono in e, poi in de, ecc...).

↓
1-item 2-items 3-items ...

1) Estraie i Prefix Path Sub-Trees



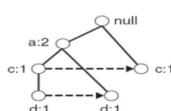
2) Dai Prefix Path subtrees costruisco il conditional fp-tree ed esraggo i pattern frequenti

a) Esempio con "e":

- Verifico se e è un frequent item contando lungo la linked list.
- Verifico i path che hanno come suffisso "e" nel transaction database e il loro supporto

$\{a,c,d : +\}, \{a,d : +\}, \{b,c : +\}$
 \nwarrow supporto < 1

- Costruisco il conditional FP-tree per "e" considerando gli item che compaiono con supp. ≥ 2

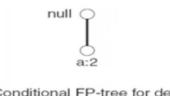


min-supp
↓

- Considero poi ricorsivamente i path con suffisso "de" nel conditional fp-tree di "e" e il loro supporto

$\{a,d : +\}, \{a : +\}$
 \nwarrow supp < 1

- Costruisco il conditional fp-tree per "de" considerando gli item che compaiono con supp. ≥ 2



$\{d, e\}$ e $\{a, d, e\}$ sono frequent-itemsets

- Considero poi ricorsivamente i path con suffisso "ce" nel conditional fp-tree di "e" e il loro supporto

$\{a : 4\}$

Not che non ho un conditional tree per "ce" \rightarrow solo $\{c, e\}$ frequent

- Considero poi ricorsivamente i path con suffisso "ae" nel conditional fp-tree di "e" e il loro supporto

Not che non ho un conditional tree per "ae" \rightarrow solo $\{a, e\}$ frequent

b) Ripeto per tutte le altre lettere (-item itemsets)

okeng

Transaction Data Set

TID	Items	Suffix	Frequent Itemsets
1	{a,b}	e	{e}, {d,e}, {a,d,e}, {c,e}, {a,c,e}
2	{b,c,d}	d	{d}, {c,d}, {b,c,d}, {a,c,d}, {b,d}, {a,b,d}, {a,d}
3	{a,c,d,e}	c	{c}, {b,c}, {a,b,c}, {a,c}
4	{a,d,e}	b	{b}, {a,b}
5	{a,b,c}	a	{a}
6	{a,b,c,d}		
7	{a}		
8	{a,b,c}		
9	{a,b,d}		
10	{b,c,e}		

Benefici della struttura FP-tree

- 1) Completezza
 - o Preserva la completezza dell'informazione (restituisce gli stessi risultati di APRIORI)
 - o Non spezza mai un lungo pattern di una qualsiasi transizione
- 2) Compattezza
 - o Riduce le informazioni irrilevanti (Gli item non frequenti vengono levati)
 - o Gli item sono ordinati in ordine decrescente per frequenza, più sono frequenti, più è probabile che vengano condivisi
 - o Non sarà mai più grande del DB originale

OSSERVAZIONI

- 1) L'FP-tree deve essere salvato interamente in memoria per velocizzare l'esecuzione dell'algoritmo.
- 2) Min-sup influenza sulla complessità del FP-tree, esso diventa più complesso con min-sup che diminuisce
- 3) FP-grow risulta più veloce in termini di run-time rispetto all'algoritmo Apriori.

ECLAT

ECLAT consiste nel andare a minare i dati nel formato "verticale".

Formato verticale: Restituisce la lista delle transazioni che contengono un determinato itemset

$t(AB) = \{T_{11}, T_{25}, \dots\}$ Le transazioni che contengono sia A che B sono la numero 11, 25 ...

A differenza dell'algoritmo Apriori, ECLAT non ha bisogno di fare lo scan del DB perché le liste delle transazioni portano con se l'informazione completa richiesta per calcolare ogni supporto.

Anche se, le liste delle transazioni possono essere lunghe, occupando molta memoria e tempo computazionale per fare l'intersezione tra insieme lunghi.

Terminologia:

$t(X) = t(Y)$	X e Y appaiono sempre insieme
$t(X) \text{ contiene } T(Y)$	Le transazioni che hanno X hanno sempre anche Y

Per accelerare il mining si utilizza il diffset il quale indica la differenza tra due liste di transazioni

$$t(X) = \{T_1, T_2, T_3\}, t(XY) = \{T_1, T_3\}$$

$$\text{Diffset}(XY, X) = \{T_2\}$$

Come valutare i pattern e capire se sono interessanti

Le misure usate (supporto e confidenza) hanno alcuni problemi:

- 1) Il supporto ha il problema del rare item problem: Gli item che occorrono molto infrequentemente in un dataset sono tagliati fuori anche se potrebbero produrre regole interessanti e potenzialmente valide (dovuto al fatto che hanno un supporto minore rispetto alla soglia).
- 2) Le regole di associazione possono essere ingannevoli. Anche se la confidenza di una regola di associazione è alta, non implica che i due itemset siano correlati positivamente. Infatti posso avere una confidenza alta, ma comunque minore della probabilità di comprare i due oggetti separatamente.

Possibili altre metriche che però soffrono delle misure null-transaction (Sono affette dal numero totale di transazioni)

- Lift - Lift misura quante volte in più X e Y occorrono insieme rispetto a quanto ci si aspetterebbe se fossero statisticamente indipendenti. Se vicino a 1 indica indipendenza, il valore del lift può essere maggiore di 1

$$\text{lift} = \frac{P(A \cup B)}{P(A)P(B)}$$

Se A e B sono indipendenti, otengo che $P(A \cup B) = P(A) + P(B)$ e di conseguenza il lift vale 1.
Se lift >> 1 ho una correlazione positiva tra A e B
Lift permette anche di investigare gli itemset rari perché producono valori di Lift molto alti

- Correlation (chi-square)

Queste misure possono dare un'idea della correlazione ma soffrono comunque di un problema generale: Sono influenzate dalle null-transaction, cioè transazioni le quali non contengono nessuno degli itemset che vengono esaminati. (Non contengono né A né B).

Vengono quindi usate queste misure (valore alto, vicino a 1, indica una correlazione forte), essendo misure null-invariant cioè il loro valore non è influenzato dalle null-transactions:

■ All_confidence [0,1]

$$all_conf(A, B) = \frac{sup(A \cup B)}{max\{sup(A), sup(B)\}} = min\{P(A|B), P(B|A)\}$$

■ Max_confidence [0,1]

$$max_conf(A, B) = max\{P(A|B), P(B|A)\}$$

■ Kulczynski [0,1]

$$Kulc(A, B) = \frac{1}{2} \cdot (P(A|B) + P(B|A))$$

■ Cosine [0,1]

$$\cosine(A, B) = \frac{P(A \cup B)}{\sqrt{P(A) \times P(B)}} = \frac{sup(A \cup B)}{\sqrt{sup(A) \times sup(B)}}$$

Null-invariance è una importante proprietà per misurare le associazioni in db grandi di transazioni.

Le null-transaction sono il numero di transazioni che non contengono né A né B.

Sono null-invariant perché non dipendono dal tot. Number delle transaction perché si elide al numeratore e al denominatore quando si vanno a scomporre le prob. Condizionali.

Attenzione: Se l'imbalance ratio, il quale misura lo sbilanciamento di due itemset A e B nelle regole di implicazione (Se c'è A non c'è B, e viceversa), è molto alto, non possiamo dedurre effettivamente se abbiamo una correlazione.

$$IR(A, B) = \frac{|sup(A) - sup(B)|}{sup(A) + sup(B) - sup(A \cup B)}$$

Null-Transaction

Data	mc	mc̄	m̄c	m̄c̄	all conf.	max conf.	Kulc.	cosine	IR
D ₁	10.000	1.000	1.000	100.000	0.91	0.91	0.91	0.91	0.0
D ₂	10.000	1.000	1.000	100	0.91	0.91	0.91	0.91	0.0
D ₃	100	1.000	1.000	100.000	0.09	0.09	0.09	0.09	0.0
D ₄	1.000	1.000	1.000	100.000	0.5	0.5	0.5	0.5	0.0
D ₅	1.000	100	100	100.000	0.09	0.91	0.5	0.29	0.89
D ₆	1.000	100	100	100.000	0.01	0.99	0.5	0.10	0.09

Non bilanciati



Kulczynski: il più preciso perché ci dice che non posso decidere se sono correlati o meno (giusto perché l'imbalance ratio è alto).

Classificazione

La classificazione consiste nel andare a determinare la classe di appartenenza di una certa istanza data in input.

Data Cleaning	Rimuovere o ridurre il rumore e trattare i valori mancanti
Relevance Analysis	Molti degli attributi potrebbero essere non rilevanti => Correlation analysis e Attribute subset selection
Data Transformation and Reduction	<ul style="list-style-type: none"> Normalizzare i dati (scalarli in un certo range) Generalizzarli con concetti di livello più alto (ad esempio dividere gli attr. Numerici in categorie, trasformandoli in attr. Nominali) Discretizzazione: Trasformare attributi continuoi in categorici cercando di discretizzarli in modo da migliorare il più possibile la classificazione. Questo procedimento è spesso "obbligato" perché alcune tecniche di machine learning non lavorano con valori continuoi ma richiedono che essi siano categorici. (Ad esempio per il decision tree). La discretizzazione influenza sul risultato della classificazione.

Discretizzazione di Fayyad e Irani (supervised): Discretizzare i dati in modo da trovare la partizione ottimale per ogni valore continuo dell'attributo A. Ordinare i valori dell'attributo A, gli autori provvedono che l'ultimo punto di taglio è sempre tra due valori di A di classi differenti.

Funzionamento:
Un possibile punto di taglio T è un punto di confine che nella sequenza di esempi ordinata dal valore A, esistono due esempi, e_1 e e_2 , aventi classi differenti, C_1 e C_2 , con $A(e_1) < T < A(e_2)$ e non esiste nessun altro esempio t.c. $A(e_3) < T < A(e_4)$. Una volta trovati tutti i possibili tagli T, l'algoritmo usa l'entropy per misurare la qualità dei punti di taglio trovati e decide quali sono i migliori.

Definizione: Si considera un insieme di esempi. Siano C_1, \dots, C_n le classi. Sia $P(C_i, S)$ la probabilità di C_i sugli esempi S. La classe entropy è definita come:

$$Ent(S) = - \sum_{i=1}^n P(C_i, S) \cdot \log(P(C_i, S))$$

La entropy è alta quando la probabilità di ogni classe è approssimativamente la stessa, viceversa se la probabilità di una classe è molto alta (ad esempio 1, quando ha una sola classe), l'entropia varrà 0.

Dato un insieme S, vogliamo trovare quel punto di taglio che divide S in due partitioni S_1, S_2 che hanno entropia minima.

La classe information entropy di una partizione indotta dal taglio T è nota come $EP(A, T; S)$ ed è definita come:

$$EP(A, T; S) = \frac{|S_1|}{N} Ent(S_1) + \frac{|S_2|}{N} Ent(S_2)$$

Dove $\frac{|S_1|}{N} = \frac{T}{S}$

T_A è un punto di taglio tale che EP è minimo rispetto a tutti gli altri punti di taglio trovati (dove l'entropia di S_1 e S_2 è minore).

Questo approccio ha una stopping condition definita. Divide in loop gli insiemini S $\rightarrow S_1, S_2 \rightarrow S_1, S_2, S_3, S_4$ ecc.. Fin quando la stopping condition non viene rispettata.

Decision Tree

Il decision tree è un modello di classificazione e rappresenta un albero di decisione generato dal training set.

Ogni nodo (non foglia) è un "test" su uno specifico attributo. Ogni ramo è un possibile risultato del test. Ogni nodo foglia è un class label.

La fase di generazionne consiste nel seguire il decision tree dalla radice alle foglie per andare a definire la classe data un'istanza.

Caratteristiche:

- Non richiede la conoscenza del dominio
- Non richiede il settaggio dei parametri (per la generazione dell'albero)
- Può gestire dati di grande dimensionalità
- La rappresentazione è intuitiva e facile da assimilare
- Possiamo fare attribute selection durante la creazione del decision tree (gli attributi che non vengono usati possono essere considerati non rilevanti per la classificazione che facciamo)
- Potremo rappresentare ogni path come un insieme di regole IF-ELSE
- Tutti gli attributi DEVONO ESSERE CATEGORICI (usare chi-merge o Fayyad e Irani per andare a categorizzare gli attributi continui).

Algoritmo di generazione dell'albero:

- Si parte dalla radice
- Viene scelto un attributo (sulla base di una misura statistica) e vengono creati i nodi per ogni valore di quel attributo categorico
- L'algoritmo si ferma quando:
 - Non ci sono attributi ulteriori da scegliere
 - Tutte le istanze rimaste appartengono alla stessa classe
 - Non ci sono più istanze da classificare

E' un algoritmo greedy (scelta dell'attributo per l'ottimo locale ad ogni stage) e l'albero è costruito top-down in modo ricorsivo con un approccio divide-and-conquer.

Gli attributi vengono scelti in modo da creare rami i più puri possibili (1 sola classe per ramo)

1 partizione per ogni elemento \rightarrow partizioni pure

2) Gain Ratio (C4.5 - J48)

L'information gain ha un problema nel gestire gli attributi che hanno un numero alto di valori (ad esempio gli ID, date ecc.).

Questo perché l'entropia risulta 0 per ogni valore dell'attributo e quindi l'information gain risulta molto alto. Di conseguenza viene scelto come attributo per il decision tree anche se non è di nessuna utilità per la classificazione.

Il C4.5 successore dell'ID3 utilizza il gain ratio per risolvere questo problema.

Adotta una normalizzazione all'information gain usando una "split information" definita come:

$$SplitInfo_A(D) = - \sum_{j=1}^r \frac{|D_j|}{|D|} \times \log_2 \left(\frac{|D_j|}{|D|} \right)$$

Quando ho molti valori distinti la cardinalità di D è bassa e di conseguenza $\log_2 \left(\frac{|D_j|}{|D|} \right)$ è un valore molto alto. Di conseguenza il valore di SplitInfo è molto alto.

Quindi se andiamo a dividere l'information gain per lo SplitInfo, otengo che gli attributi con molti valori distinti avranno un information gain molto basso!

$$GainRatio(A) = Gain(A)/SplitInfo(A)$$

Verrà quindi scelto l'attributo A con il valore di GainRatio(A) maggiore rispetto agli altri attributi.

Info_A(D) = $\sum_{j=1}^r |D_j| \times Info(D_j)$

es: A = bianco, nero, rosso

D₁ = 4 tuple che contengono bianco

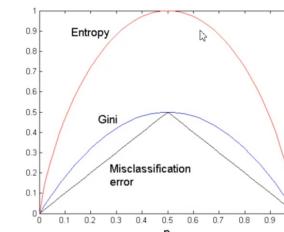
D₂ = 4 tuple che contengono nero

D₃ = 4 tuple che contengono rosso

Info_A(D) = $\sum_{j=1}^r |D_j| \times Info(D_j)$

quando queste 3 funzioni sono pure,

Così che divide il dataset in partizioni più pure possibili.



3) Gini Index (CART, IBM)

Sia D l'insieme delle istanze e m il numero di classi, il gini index è definito come:

$$gini(D) = 1 - \sum_{i=1}^m p_i^2$$

Dove: $p_i = \frac{|C_i|}{D}$

2 subsets

Il gini index considera un binary split per ogni attributo.

Se A è un attributo categorico con n valori, tutti i suoi subset possibili che possono essere formati sono esaminati per determinare qual è il binary split ottimale.

Se D l'insieme di istanze viene diviso su A in due subset D_1 e D_2 , allora il gini index è definito come

$$gini_A(D) = \frac{|D_1|}{|D|} gini(D_1) + \frac{|D_2|}{|D|} gini(D_2)$$

• Per ogni attributo categorico, ogni binary split possibile è considerato

• Per ogni attributo continuo, ogni split point deve essere considerato (molto costoso)

binary split con $gini_A(D)$ minore per l'altra. A

Riduzione in impurity:

$$\Delta gini(A) = gini(D) - gini_A(D)$$

L'attributo che massimizza la riduzione dell'impurity (o equivalentemente ha il minimo Gini index) viene scelto come splitting attribute.

The three measures, in general, return good results but

• Information gain: biased toward multivalued attributes

• Gain ratio: tends to prefer unbalanced splits in which one partition is much smaller than the others

• Gini index: suited to multi-valued attributes

• Gini index: not difficult when the number of classes is large

• Tends to favor tests that result in equal-sized partitions and purity in both partitions

Quale attribute selection misure è la migliore?

Tutte danno dei buoni risultati, nessuna è significativamente superiore alle altre.

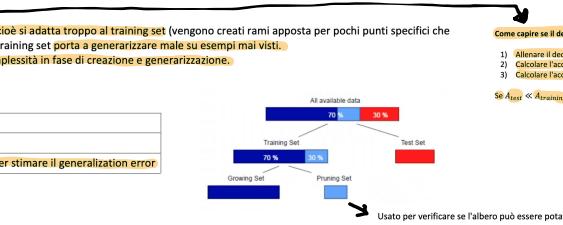
Dovremo scegliere quella che riduce il più possibile l'altezza dell'albero di decisione perché la complessità aumenta esponenzialmente con l'altezza dello stesso.

Problema del decision tree

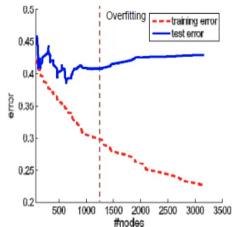
Il modello **Decision Tree** soffre del problema dell'**overfitting**, cioè si adatta troppo al training set (vengono creati rami apposta per pochi punti specifici che potrebbero essere noise points). L'adattamento eccessivo al training set porta a generizzare male su esempi mai visti. Inoltre maggiore è l'altezza dell'albero, maggiore è la sua complessità in fase di creazione e generalizzazione.

Definizioni:

Re-substitution errors (e_{TR})	Errore sul training set
Generalization errors (e_{TS})	Errore sul test set
Reduced error pruning (REP)	Usare un data set di pruning per stimare il generalization error



- Come capire se il decision tree è in overfitting?
- 1) Allenare il decision model tramite il training set
 - 2) Calcolare l'accuracy del modello classificando il training set
 - 3) Calcolare l'accuracy del modello classificando il test set



Occam's Razor: tra due modelli con generalization errors simili, dovremmo scegliere il modello più semplice rispetto a quello più complesso. Questo perché il modello complesso ha una chance maggiore che sia stato fittato accidentalmente dagli errori nei dati.

Per ridurre l'overfitting si effettua un **Tree Pruning** (sulle foglie dell'albero).

- 1) Prepruning: Fermare la costruzione dell'albero in anticipo. Si possono usare diverse condizioni, ad esempio:
 - a. Se tutte le istanze appartengono alla stessa classe
 - b. Se tutti i valori sono uguali
 - c. Se il numero di istanze rimaste è minore di una certa soglia specificata dall'utente
 - d. Se la distribuzione delle istanze è indipendente dagli attributi disponibili (usando il Chi-square test)
 - e. Se espandendo il nodo corrente non si migliora le misure di impurità (information gain o Gini index)

E' difficile scegliere delle soglie appropriate!

Tree	Number of terminal nodes	Cross-validated relative cost	Resubstitution relative cost	Complexity parameter
1	32	0.704 +/- 0.060	0.145	0.000
8	16	0.639 +/- 0.058	0.244	0.008
9	14	0.635 +/- 0.058	0.276	0.008
10	12	0.633 +/- 0.058	0.310	0.008
11*	11	0.603 +/- 0.057	0.332	0.011
12**	8	0.634 +/- 0.058	0.430	0.016
13	7	0.668 +/- 0.059	0.464	0.017
14	5	0.689 +/- 0.059	0.420	0.019
15	3	0.700 +/- 0.058	0.619	0.020
16	2	0.729 +/- 0.048	0.696	0.038
17	1	1.000 +/- 0.000	1.000	0.152

Initial misclassification cost = 5.00
Initial class assignment = 0

- 2) Postpruning: Rimuovere alcuni rami da un albero "plenamente cresciuto" e sostituirli con una foglia. Per fare ciò si usa un insieme di dati differenti (pruning set) dal training dataset per decidere quali è il miglior albero "potato".

- a. **Reduced Error Pruning (REP)**

Si usa il pruning set per stimare l'accuratezza dei sotto-alberi e l'accuratezza dei nodi individuali.

Sia T un sub-tree con il nodo V alla radice



Il gain dato dal potare su V può essere definito come: #misclassifications in T - #misclassifications in V → Le misclassifications vengono considerate sul pruning set!

Ciclo: Si pota il nodo con gain maggiore fin quando tutti i nodi rimasti non hanno gain negativo.

Restrizione: T può essere potato solo se non contiene un sub-tree con errore minore rispetto a T

- b. **Cost complexity Pruning (used in CART)**

Viene usata una funzione chiamata cost complexity.

$$\text{Cost complexity} = \frac{\text{Resubstitution error}}{\text{Resubstitution Error} + B \cdot \text{Number of leaf nodes}}$$

Resubstitution error	Pruning set errors
Number of leaf nodes	Numero di foglie nell'albero
Beta	E' un parametro che indica la penalità per nodi terminali addizionali

Si inizia dal bottom dell'albero e per ogni nodo interno, N, viene calcolata la cost complexity del subree in N e la cost complexity del subree in N se venisse potato. Vengono comparati i due valori e se potare il subree (N diventa una foglia) porta ad una cost complexity minore, allora il subree viene potato. Altrimenti viene lasciato com'è.

- c. **Pessimistic pruning**

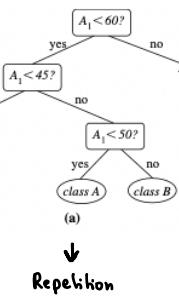
E' simile al cost complexity pruning ma non usa un pruning set. Aggiusta gli "error rates" ottenuti sul training set aggiungendo una penalità, computata adottando un approccio euristico basato sulla teoria statistica (la penalità serve per counterare il bias introdotto dal considerare l'error rate sul training set). Se l'error rate in un nodo è minore dell'error rate in un sottoalbero con origine quel nodo, allora quel sottoalbero viene potato.

Problema del postpruning

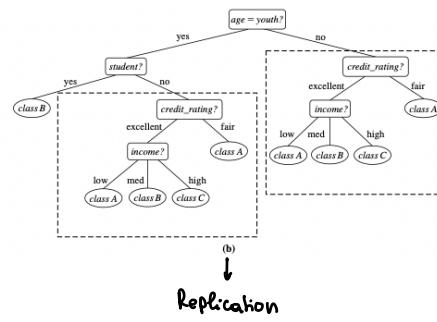
Il postpruning tende ad essere più compatto della sua controparte non potata, ma continua a soffrire di:

Ripetizione	Un attributo è ripetutamente testato lungo un ramo
Replicazione	Esistono sottoalberi duplicati dentro un albero

Questi due fattori possono impedire l'accuratezza e la comprensibilità di un decision tree.
Per risolvere si può: Usare splits (divisioni) basati su più attributi (nella scelta degli attributi se ne sceglio più insieme) o anche usare una differenza forma di rappresentazione, come le regole IF-THEN al posto degli alberi decisionali.



↓
Repetition



↓
Replication

Bayesian Classification

E' un classificatore statistico, cioè performa predizioni basate sulla statistica. Esso è basato sul **teorema di bayes**.

Questo classificatore è **incrementale**, cioè ogni training example può incrementare/diminuire la probabilità che un'ipotesi sia corretta.

Terminologia:

Sia X una tupla dove la class label non è conosciuta. Sia H un'ipotesi che X appartenga ad una classe C.

$P(H|X)$ – **Posteriori probability**: E' la probabilità che l'ipotesi H tenga data l'istanza X.

$P(H)$ – **Prior probability of H**: Probabilità iniziale

$P(X)$ – **Prior probability of X**: Probabilità che una certa istanza venga osservata

$P(X|H)$ – **Posteriori probability of X conditioned on H**: Probabilità di osservare il sample X sapendo che l'ipotesi tiene.

Osservazione: $P(H), P(X)$ e $P(X|H)$ possono essere stimata tramite i dati. $P(H|X)$ può essere calcolata tramite il Bayes' theorem.

Attenzione: Calcolare la probabilità $P(H|X)$ non è computazionalmente facile perché X è un vettore di valori e bisogna calcolare la probabilità condizionale tra H e ogni valore di X.

Dato un training data X:

$$P(H|X) = \frac{(P(X|H) * P(H))}{P(X)}$$

Funzionamento

Sia D un training set di tuple che hanno associata la loro classe, e ogni tupla è rappresentata da un vettore $X = (x_1, x_2, \dots, x_n)$.

Supponiamo che ci sono m classi C_1, C_2, \dots, C_m

Vogliamo determinare qual è la classe che ha la più alta posteriori probability, cioè che massimizza $P(C_i|X)$.

$$P(C_i | X) = P(X | C_i) * P(C_i)$$

Osservazione: $P(X)$ al denominatore scompare perché è costante

Dove:

$$1) P(C_i) = \frac{|C_i|}{|D|}$$

Calcolare $P(C_i | X)$ è molto costoso, andiamo a velocizzare questa computazione con l'assunzione che gli attributi siano indipendenti tra loro.

$$2) P(X|C_i) = \prod_{k=1}^n P(x_k | C_i) = P(x_1 | C_i) \times P(x_2 | C_i) \times \dots \times P(x_n | C_i)$$

Attenzione: $P(X|C_i)$ richiede che ogni probabilità condizionale sia diversa da 0, altrimenti la computazione diventa 0!

Per risolvere questo problema usiamo la Laplacian Correction. Consiste nell'andare ad aggiungere 1 al numero di istanze per ogni valore dell'attributo.

Ex. Suppose a dataset with 1000 tuples, income=low (0), income=medium (1), income=high (2)

Use Laplacian correction (or Laplacian estimator)

* Adding 1 to each case

$\text{Prob}(\text{income} = \text{low}) = 1/1000 = 0.001$

$\text{Prob}(\text{income} = \text{medium}) = 991/1003 = 0.988$

$\text{Prob}(\text{income} = \text{high}) = 11/1003 = 0.011$

If A_i is categorical, $P(x_i|C)$ is the number of tuples in C having value x_i for A_i , divided by $|C|$, D (number of tuples of C in D)

If A_i is continuous-valued, $P(x_i|C)$ is usually computed based on Gaussian distribution with a mean μ and standard deviation σ

$$g(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

and $P(x_i | C_j) = g(x_i, \mu_j, \sigma_j)$

where μ_j and σ_j are the mean value and the standard deviation, respectively, of the values of attribute A_i for training tuples of class C_j .

Vantaggi e Svantaggi

Vantaggi

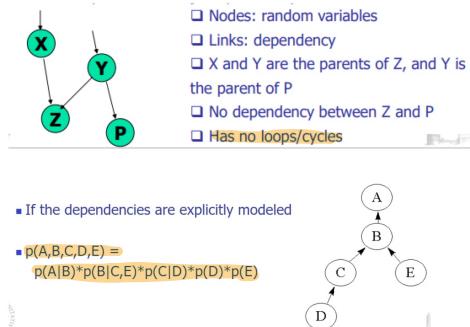
- Facile da implementare
- Buoni risultati nella maggior parte dei casi

Svantaggi

- Assunzione che tutti gli attributi siano indipendenti tra loro, le dipendenze tra gli attributi non possono essere modellati tramite il Naive Bayesian Classifier.

Bayesian Belief Networks

E' un classificatore basato sul Bayesian che risolve il problema dell'assunzione di indipendenza tra gli attributi, infatti permette la rappresentazione delle dipendenze tra sottoinsiemi di attributi.



Dato $X = (x_1, \dots, x_n)$ il quale è una tupla descritta dagli attributi Y_1, \dots, Y_n rispettivamente. Ogni Y_i è condizionalmente indipendente dai suoi non discendenti nel grafo.

$$P(x_1, \dots, x_n) = \prod_{i=1}^n P(x_i | \text{Parents}(Y_i))$$

→ dipende solo dai parents

Per allenare una Bayesian Network si usa il gradient descent:

- Compute the gradients (for each training tuple X_i)

$$\frac{\partial \ln P_w(D)}{\partial w_{ijk}} = \sum_{d=1}^{|D|} \frac{P(Y_i = y_{ij}, U_i = u_{ik} | X_d)}{w_{ijk}}$$

- Take a small step in the direction of the gradient (where λ is the learning rate)

$$w_{ijk} \leftarrow w_{ijk} + \lambda \frac{\partial \ln P_w(D)}{\partial w_{ijk}}$$

- Renormalize the weights (all the weights have to be between 0 and 1 being probabilities and $\sum_j w_{ijk}$ must equal 1 for all i,k)

Rule-based classification

Si rappresenta la conoscenza tramite la forma delle regole IF-THEN. → es: $(\text{age} = \text{youth}) \wedge (\text{student} = \text{yes}) \Rightarrow (\text{buys_computer} = \text{yes})$

Terminologia

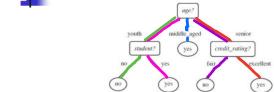
n_{covers}	Numero di tuple coperte della regola R	Le condizioni di una regola sono vere per una tupla se è risultato di una regola e' uguale a quello della tupla
$n_{correct}$	Numero di tuple correttamente classificate dalla regola R	
$coverge(R)$	$\frac{n_{covers}}{ D }$ dove D = training dataset	= % tuple coperte da R
$accuracy(R)$	$\frac{n_{correct}}{n_{covers}}$	= Numero tuple class corrette / su coperte

Se più di una regola viene "attivata" abbiamo bisogno di una conflict resolution.

Se nessuna regola viene attivata, abbiamo bisogno di una default rule.

Come estrarre le regole?

- 1) Le regole vengono estratte da un decision tree. Ogni path crea una regola.



Example: Rule extraction from our buys_computer decision-tree

```

IF age = youth AND student = no THEN buys_computer = no
IF age = youth AND student = yes THEN buys_computer = yes
IF age = middle-aged THEN buys_computer = yes
IF age = senior AND credit_rating = excellent THEN buys_computer = no
IF age = senior AND credit_rating = fair THEN buys_computer = yes
  
```

Come potere l'insieme di regole?

- 1) Ogni condizione che non migliora l'accuratezza di una regola può essere potata
- 2) Ogni regola che non contribuisce nell'accuratezza generale può essere potata

Dopo la potatura le regole non saranno più mutualmente esclusive.

CAIS adotta un class-based ordering scheme: raggruppa le regole per classe e determina un rank per questi insiemi di regole in modo da minimizzare il numero di falsi-positivi. La classe con il numero minimo di falsi-positivi viene esaminata per prima. Esiste una classe di default la quale contiene il più grande numero di tuple che non sono coperte da nessuna regola.

• le regole sono mutualmente esclusive, cioè non abbiano sulle conflicts

• le regole sono esauritive, cioè esiste una regola per ogni combinazione attributo-value. (non abbiamo bisogno della default rule)

Conflict Resolution = Una tupla è coperta da più regole

Le possibili strategie di conflict resolution sono:

- 1) Size ordering: Assegnare la priorità più alta alle regole che testano più attributi
- 2) Class-based ordering: Le regole per le classi più frequenti vengono prima (decreasing order of prevalence) oppure le regole per la classe con il costo più alto vengono prima (misclassification cost per class)
- 3) Rule-based ordering: Le regole sono organizzate in una lunga priority list, ordinate secondo una certa misura (rule quality) o basata su un consiglio dei domain experts.

as: Accuracy, coverage, size ...

2) Approchi euristici

Per l'approccio euristico viene usato il Sequential covering algorithm il quale estrae le regole direttamente dal training dataset.

Passo:

- 1) Le regole vengono generate una alla volta (una per ogni classe, si cicla sulla classe)
- 2) Ogni volta che una regola viene imparata, le tuple che vengono coperte dalla regola vengono rimosse
- 3) Il processo si ripete sulle tuple rimanenti fino a una condizione di terminazione, cioè quando non ci sono più esempi di training o quando la qualità di una regola è sotto una certa soglia specificata dall'utente.

Per imparare una regola si utilizza una strategia Greedy depth-first, si parte dalla regola più generale possibile e man mano si generano tutte le altre regole.

Method:

- 1) Rule_Set = [] // initial set of rules learned is empty
- 2) for each class c do
 - report
 - Rule = Learn One Rule(D, All attrs, c)
 - Rule_Set = Rule_Set ∪ Rules covered by Rule from D
 - 6) Rule_Set = Rule_Set ∪ Rule / add new rule to rule set
 - 7) until terminating condition
 - 8) endfor
 - 9) return Rule_Set

Per misurare la qualità di una regola si usa la misura FOIL-gain, la quale favorisce le regole che hanno un'alta accuratezza e copre tante tuple. (usa solo l'accuracy)

Do:

$$\text{FOIL_Gain} = pos \times (\log_2 \frac{pos}{pos+neg}) - \log_2 \frac{pos}{pos+neg}$$

Come potere l'insieme di regole? (è necessario un crit. test)

FOIL utilizza un sistema molto semplice ed effettivo:

$$\text{FOIL_Prune}(R) = \frac{pos - neg}{pos + neg} \rightarrow pos \text{ e } neg \text{ calcolati su un pruning set!}$$

Se $\text{FOIL_Prune}(R) > \text{FOIL_Prune}(R_{pruned}) \Rightarrow$ pota R

OSSERVAZIONE: Più corte sono le condizioni nelle regole, meglio è (se possibile).

Classificazione usando i pattern frequenti

Consiste nell'andare ad usare i pattern frequenti per fare la classificazione. Andare a minare i dati per trovare associazioni forti tra pattern frequenti e class labels.

Le regole generate hanno questa forma: $p_1 \text{ and } p_2 \text{ and } \dots \text{ and } p_l \Rightarrow C$ ($conf, supp$)

E' effettivo perché supera alcuni limiti del decision tree (il fatto di considerare un attr. per volta) ed inoltre è stato provato che spesso è più accurato dei metodi di classificazione tradizionali.

Classification based on associations (CBA)

Consiste nell'andare a minare possibili association rules nella forma

Cond-set (a set of attribute-value pairs) \Rightarrow class label

Costruzione del classificatore

Si usa un metodo euristico dove ogni regola è organizzata secondo una precedenza decrescente basata sulla confidenza e poi sul supporto. Se un insieme di regola hanno lo stesso antecedente, allora le regole con la confidenza più alta vengono selezionate per rappresentare l'insieme.

Classificare le nuove tuple

La prima regola che soddisfa la tupla è usata per classificare. Il classificatore ha anche una default rule che ha la precedenza più bassa.

Classification based on Multiple Association Rules (CMAR)

Costruzione del classificatore

Adotta una variante dell'algoritmo FP-growth per trovare un insieme completo di regole che soddisfano le soglie di min_{supp} e di min_{conf} .

Rule pruning ogni volta che una regola viene inserita nella rule base
Date due regole R_1 e R_2 , se l'antecedente di R_1 è più generale di quello di R_2 e $\text{conf}(R_1) \geq \text{conf}(R_2)$, allora R_2 viene potata.

Pota le regole per cui le regole antecedenti e la classe non sono positivamente correlate, basate su un test statistico chi-square.

Classificare le nuove tuple

Se le regole compatibili con la nuova tupla non sono consistenti con il class label si fa un'analisi statistica:

- 1) Le regole vengono divise in gruppi secondo i class labels
- 2) Viene usato una misura chi-square pesata per trovare il gruppo più forte di regole, basato su una correlazione statistica delle regole
- 3) Le nuove tuple sono assegnate al class label del gruppo più forte

CMAR ha un'accuratezza media leggermente più alta ed è più efficiente in memoria rispetto a CBA

Classification based on predictive association (CPAR)

Costruzione del classificatore

La generazione delle regole predittive è basata su un algoritmo di generazione delle regole (FOLI-like analysis) piuttosto che al frequent itemset mining.

Le regole vengono generate a partire dal training set come in FOLI, ma le tuple non vengono rimosse quando sono coperte da una regola, ma gli viene abbassato il loro peso.

Classificare le nuove tuple

- 1) Le regole vengono divise in gruppi secondo i class labels
- 2) Le migliori K regole, basate sull'accuratezza prevista, di ogni gruppo sono usate per predirre il class label

CPAR è molto efficiente e l'accuratezza è simile a CMAR

Lazy vs Eager Learning

Fin'ora abbiamo visto dei metodi Eager Learning, cioè dato un insieme di tuple di addestramento, viene costruito un modello prima di ricevere i nuovi dati da classificare.

Ora andremo ad analizzare i metodi di tipo Lazy Learning, i quali semplicemente immagazzinano i dati di addestramento e aspettano fin quando non gli viene data una tupla da testare.

I metodi Lazy richiedono meno tempo nell'addestramento ma più tempo nella generalizzazione!

I metodi Lazy usano uno spazio di ipotesi più ampio essendo che usa un insieme di funzioni lineari locali per formare una approssimazione globale implicita della funzione di target. Invece gli eager devono far affidamento ad una singola ipotesi che copre l'intero spazio delle istanze.

K-Nearest Neighbor Algorithm

Tutte le istanze corrispondono in punti in uno spazio di n dimensioni.

I punti più vicini tra loro sono determinati tramite la Euclidean Distance = $\text{dist}(X_1, X_2)$

K-NN restituisce il valore più comune tra i K esempi di training vicini a X_q



Come calcolare la distanza per gli attributi categorici?

Si comparano i valori degli attributi, se sono identici la differenza è 0, senno è 1

Come gestire i valori mancanti?

Se un valore di un dato attributo è mancante in uno delle due tuple, allora si assume la massima differenza possibile. Questa differenza è 1 per gli attr. Nominali e per gli attributi numeri quando il valore è mancante per entrambe le tuple.

Come è possibile determinare un buon valore di K?

Sperimentalmente, si usa un valore incrementale di K e si sceglie il K che da la massima accuratezza sperimentalmente.

Come scegliere la distanza?

Diverse misure possono essere usate incorportando il peso degli attributi e tagliando il rumore delle tuple.

Efficienza computazionale = $O(n \times |D|)$ – Devo comparare la nuova istanza con tutte le istanze nel training set

Ancora da ordinare e arrangiare le tuple in alberi di ricerca, il numero di comparazioni può essere ridotto a $O(\log(|D|))$. Inoltre si possono usare la **partial distance**, la quale usa un sottoinsieme degli n attributi e una soglia per determinare se la comparazione della distanza termina, e gli **editing methods** i quali rimuovono le tuple di training che sono provate essere inutili.

Essendo che n (numero di attributi) sarà piccolo, quando dobbiamo andare ad usare KNN con un numero veramente grande di istanze, dovremo andare a ridurle con gli editing methods (ridurre la dimensionalità non è cruciale allo stesso modo).

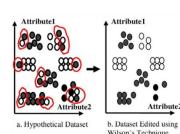
Editing methods

Wilson Editing

Pulisce le regioni di sovrapposizione tra classi, portando a confini più uniformi tra esse. Non ci aspettiamo di ridurre troppo il numero di istanze, ma di migliorare l'accuracy.

```
PREPROCESSING
A: For each example  $c_i$  in the dataset  $\mathcal{O}_i$ 
1: Take the k-nearest neighbors of  $c_i$  (excluding  $c_i$ )
2: Label  $c_i$  with the class associated with the largest number of examples among the k nearest neighbors (breaking ties randomly)
B: Retain  $\mathcal{O}_i$  by deleting all examples that were misclassified in step A.1.
```

CLASSIFICATION RULE:
Classify new example q using K-NN rule with the edited subset \mathcal{O}_i

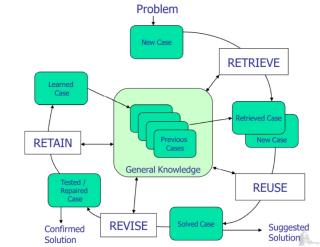


Case-Based Reasoning (CBR)

E' il processo di risoluzione dei nuovi problemi basati sulla soluzione di problemi passati uguali o simili.

Usa un db composto da risoluzioni a problemi per risolvere i nuovi problemi.

Immaginiamo descrizioni simboliche (tuple o casi), non punti nello spazio. -> Difficile da implementare CBR dovuto alla difficoltà nel comparare i casi non essendo valori numerici.



Retrieve

Uno o alcuni casi vengono selezionati, basati sulla similarità. Questo è un task K-NN basato su una specifica funzione di similarità. Quando la base dei casi cresce, l'efficienza con cui vengono trovati i casi simili diminuisce.

Reuse

Consiste nel riutilizzare la soluzione trovata per un caso se è adatta per il problema così com'è, oppure adattarla prima di utilizzarla. Di solito si cerca di evitare adattamenti costosi.

Revise

In questa fase viene restituito un feedback relativo alla soluzione costruita, il caso revised entra nel CBR system per il suo uso in una successiva retain phase.

Retain

Questo è la fase di addestramento del CBR system, cioè vengono aggiunti i casi revisionati alla case base.

Multi-edit

Applica il Wilson editing ripetutamente su N random sottosampli del dataset originale fin quando nessun altro esempio viene rimosso.

```
A: DIFFUSION: Divide the dataset O into N=3 random subsets  $S_1, \dots, S_N$ 
B: CLUSTERING: Classify  $S_1$  using the K-NN rule with  $S_{(1) \text{rest}}$  as the training set ( $i=1, \dots, N$ )
C: EDITING: Remove all incorrectly classified examples
D: COMBINATION: Replace  $\mathcal{O}$  by subset  $\mathcal{O}_i$  consisting of the union of all remaining examples in  $S_i$ 
E: TEST: If  $\mathcal{O}_i$  is the last iteration produced no editing then terminate; otherwise go to step A.
```

Citation Editing

Citation Editing

Analogia: if a paper cites another article, the paper is related to that article. Similarly, if a paper is cited by an article, the paper is also related to that article. Thus both the papers and references are considered to be related to a given paper.

```
a: For each example  $c_i$  in dataset  $\mathcal{O}$ :
1: Find the K nearest neighbors of  $c_i$  in  $\mathcal{O}$  (excluding  $c_i$ )
2: Find the C nearest citations of  $c_i$  among their K nearest neighbors
3: Classify  $c_i$  with the majority class of the examples in a group containing the C nearest neighbors and C nearest citations
b: Discard example  $c_i$  from  $\mathcal{O}$  that was identified in step A.1, obtaining  $\mathcal{O}'$ 
```

Supervised Clustering

\mathcal{O} è rimpiazzato da un subset \mathcal{O}_i che consiste in cluster rappresentanti che sono selezionati dall'algoritmo di clustering supervisionato.

```
A: Apply a supervised clustering algorithm to dataset  $\mathcal{O}$  to produce a set clusters (each having a single representative).
B: Test  $\mathcal{O}_i$  on  $\mathcal{O}$  by deleting all non-representative examples to produce subset  $\mathcal{O}'$ .
```

	I-NN	Wilson	Multi-edit	Cluster	SC Editing
Classification accuracy	93.11	93.11	72.10	87.71	82.57
TS Compression Rate	0	13.49	0	13.10	13.50
b. 2D Dots					
Classification accuracy	90.61	91.00	75.49	92.21	89.93
TS Compression rate	0	9.39	0	10.20	8.68

Number of instances after editing in the training set
 $\text{Training set compression rate} = (1 - \frac{\text{TS compression rate}}{100}) \cdot 100$
Number of instances in the original training set

Valutazione dei modelli e selezione

Metriche di valutazione dei modelli

Soltanente quando si valuta un modello di classificazione si costruisce la **confusion matrix**.

Actual class\Predicted class	C_1	$\neg C_1$
C_1	True Positives (TP)	False Negatives (FN)
$\neg C_1$	False Positives (FP)	True Negatives (TN)

Dalla matrice di confusione possiamo andare a ricavare diverse metriche di valutazione, tutti i valori sono compresi [0,1].

Accuracy	Percentuale di tuple che sono state classificate correttamente	$Accuracy = \frac{(TP + TN)}{All}$
Error Rate	Percentuale di tuple che sono state classificate erroneamente	$Error rate = \frac{(FP + FN)}{All}$
Sensitivity	Quanti positivi sono stati riconosciuti sul numero totale di positivi	$Sensitivity = \frac{TP}{P}$
Specificity	Quanti negativi sono stati riconosciuti sul numero totale di negativi	$Specificity = \frac{TN}{N}$
Precisione	Qual è la percentuale di tuple che il classificatore ha classificato positivamente che erano effettivamente positive?	$Precision = \frac{TP}{TP + FP}$
Recall	Qual è la percentuale di tuple positive che sono state classificate positivamente	$recall = \frac{TP}{TP + FN} = \frac{P}{P}$
F measure (F o P - sopra)	Media armonica tra precisione e recall, permette di combinarle insieme.	$F = \frac{2 \times precision \times recall}{precision + recall}$
F_{beta}	Misura pesata tra precisione e recall, solitamente viene usata con beta = 2 (pesa più i recall che la precisione) e beta = 0.5 (pesa di più la precisione che il recall)	$F_\beta = \frac{(1 + \beta^2) \times precision \times recall}{\beta^2 \times precision + recall}$

OSS: • Se il dataset è bilanciato uso l'accuracy come misura di qualità

• Se il dataset non è bilanciato uso precision e recall come misure di qualità

Selezione dei modelli

Per valutare quale modello scegliere, bisogna provare diversi modelli su più prove. Dobbiamo ottenere una distribuzione di misure di qualità dei modelli perché dobbiamo comparare che quelle distribuzioni non siano le stesse, che siano indipendenti tra loro.

Holdout method

Consiste nell'andare a dividere il dataset in due parti, uno di training e uno di test. Per ogni iterazione si va a dividere il dataset in diverso modo prendendo casualmente i valori. Si fanno k iterazioni con sampling diverso e si fa la media delle misure ottenute a ogni iterazione.



K-fold Cross-validation (solitamente k=10)

Il dataset viene partitionato in k subset mutualmente esclusivi, ognuno di ugual misura. All'iterazione i-th, si usa il subset D_i come test set mentre i rimanenti come training set.

Stratified cross-validation: i subset sono stratificati in modo che la distribuzione delle classi è approssimativamente la stessa rispetto al dataset iniziale.



Bootstrap

Il campionamento su un dataset di tuple viene fatto con replacement. Vengono selezionate d tuple e quelle non selezionate vengono usate come test set. Le tuple, estratte con replacement, possono essere selezionate più volte.

Bootstrap (632)

A data set with d tuples is sampled d times, with replacement, resulting in a training set of d samples. The data tuples that did not make it into the training set end up forming the test set. About 63.2% of the data tuples end up being included in the bootstrap, and the remaining 36.8% form the test set (since the probability of not being chosen is $(1 - \frac{1}{d})^d$, when d is large results to be $\approx e^{-1} = 0.368$)

• Repeat the sampling procedure k times. The overall accuracy of the model is:

$$Acc(M) = \frac{1}{k} \sum_{i=1}^k (0.632 \times Acc(M_i)_{test, set} + 0.368 \times Acc(M_i)_{train, set})$$

Come scegliere tra due classificatori?

- Usare la 10-fold cross validation per ottenere gli errori di M_1 e M_2
- Usare un test statistico significativo. Necessita di un test statistico per tenere in considerazione la distribuzione delle misure di entrambi i modelli. Se esse sono molto simili (overlapping), il valore medio dell'accuracy non basta per determinare se un classificatore è meglio di un altro a livello statistico.

OSS: Maggiore è K per la K-fold, migliore è la statistica che possiamo usare. 10 è un ottimo compromesso tra i tentativi e la qualità.

Test parametrico (T-test)

- Usare la 10-fold cross validation
- Assunzione per usare il T-test: I campioni (accuracy o f-measure) sono normalmente distribuiti all'interno di ciascun gruppo (output di ogni classificatore) e le varianze delle due popolazioni non sono assolutamente differenti.
- Scegliere il confidence limit = Margine di errore che un modello è meglio di un altro
- Usare il t-test (test con significato statistico) per valutare la differenza in media tra i due gruppi a. Null Hypothesis = Le due distribuzioni di accuratezza per M_1 e M_2 sono uguali
- Se possiamo rigettare la null hypothesis allora possiamo concludere che la differenza tra M_1 e M_2 è statisticamente significativa.
- Scegliere in questo caso il modello con l'error rate minore

Se il test set usato per M_1 e M_2 è uguale: pairwise comparison

- Per il i -esimo round del 10-fold cross-validation, usiamo la stessa partizione per ottenere $\bar{err}(M_1)$ e $\bar{err}(M_2)$.
- Si calcola la media su 10 round degli errori
- T-test: calcolare la t-statistic con k-1 degrees of freedom.

$$t = \frac{\bar{err}(M_1) - \bar{err}(M_2)}{\sqrt{\text{var}(M_1 - M_2)/k}}, \quad \text{dove} \quad \text{var}(M_1 - M_2) = \frac{1}{k} \sum_{i=1}^k [\bar{err}(M_1)_i - \bar{err}(M_2)_i - (\bar{err}(M_1) - \bar{err}(M_2))]^2$$

0.05 0.01
↑ ↑

4) Scegliere un significant level (tipicamente $5\% = 1\%$)

5) Essendo la t-distribution simmetrica, guardiamo la tabella per un livello di confidenza $z = \frac{\text{significant level}}{2}$ e un degree of freedom = $k-1$ dove k = numero di fold. $k-1$ perché se abbiamo 10 valori e le medie, abbiamo bisogno di 9 valori per conoscere gli ultimi sapendo le medie.

6) Guardando la tabella trovo t_α .

OSS: Se abbiamo molti classifier e non due, si usano dei test statistici che lavorano su più classifiers

OSS_2: Solitamente si utilizza un confidence lvl pari a 0.05

Test non parametrico statistico (Wilcoxon signed rank sum test)

Viene solitamente usato come alternativa al t-test. Esso, essendo non parametrico, non richiede nessuna assunzione sulla distribuzione degli errori dei modelli.

Null-hypothesis = La mediana di due samples (accuracy o f-measure) è uguale, cioè le due popolazioni hanno la stessa distribuzione con la stessa mediana.

Assunzione: i due samples sono indipendenti tra loro e le popolazioni hanno la stessa varianza/spread.

- Si calcola il numero di pair, per ogni fold calcoliamo la differenza di accuracy tra i due modelli in quella specifica fold.

1) $d_i = x_i - y_i$ Dove: x_i e y_i sono la coppia dell'accuracy dei due modelli al passo i -esimo del K-fold

2) Si calcolano un rank ad ogni d_i (IGNORANDO IL SENSO di d_i) → Rank 1 al $|d_i|$ più piccolo, Rank 2 al secondo $|d_i|$ più piccolo e così via...

3) Mettiamo un label ad ogni rank in cui segno, in accordo con il segno di d_i .

4) Calcoliamo

- $W^+ = \text{Somma dei ranks positivi}$
- $W^- = \text{Somma dei ranks negativi}$

La somma tra i due W dovrebbe essere uguale a $\frac{n(n+1)}{2}$ dove n è il numero di pairs

5) Si sceglie $W = \min(W^+, W^-)$

6) Si usa una tabella dei valori critici per Wilcoxon per trovare la probabilità di osservare quel valore di W o uno più "estremo".

Normal approximation

If the number of observations/pairs is such that $\frac{n(n+1)}{2}$ is large enough (> 20), a normal approximation can be used with

$$\mu_W = \frac{n(n+1)}{4}, \quad \sigma_W = \sqrt{\frac{n(n+1)(2n+1)}{24}}$$

$$8) \text{Calcola } z = \frac{W - \mu_W}{\sigma_W}$$

9) Guarda la z-table per trovare il p-value che indica con quale confidenza posso dire che le mediane sono significativamente differenti

Esempio:

Adapted to classification.
12-fold cross-validation.

	1	2	3	4	5	6	7	8	9	10	11	12
M1	2.0	3.6	2.6	2.6	7.3	3.4	4.9	6.6	2.3	2.0	6.8	8.5
M2	3.5	3.7	2.9	2.4	9.9	3.5	16.7	8.0	1.8	4.0	9.1	20.9
Diff	+1.5	+1.1	+0.3	-0.2	-6.3	-0.1	-1.8	-0.6	+1.5	+2.0	+3.2	+12.4

Ranking the differences

Diff	1	2	3	4	5	6	7	8	9	10	11	12
Rank	1	2	3	4	5	6	7	8	9	10	11	12
Sign	+	+	+	+	+	+	+	+	+	+	+	+

• Calculating W^+ and W^- :

$$W^+ = 1 + 2 + 4 = 7$$

$$W^- = 3 + 5.5 + 7 + 8 + 9 + 10 + 11 + 12 = 71$$

• Therefore

$$\frac{n(n+1)}{2} = \frac{12 \cdot 13}{2} = 78 > 20$$

$$W = \min(W^+, W^-) = 7$$

We get:

$$z = \frac{7 - 12 - 13}{\sqrt{\frac{12 \cdot 13 \cdot 25}{24}}} = \frac{7 - 39}{\sqrt{162.5 - 0.125}} = 2.51$$

• We can compute the z-score defined as:

$$\frac{\text{actual}}{w} \leftarrow \frac{z - \mu_w}{\sigma_w} \rightarrow \text{expected } w \quad (\text{w: } w^+ = w^-)$$

w → standard deviation

• A z-score is a measure of how many standard deviations below or above the population mean a raw score is.

• Looking up this score in the z-table*, we get an area of 0.9880, equal to a two-tailed p-value of 0.0112. This is a tiny p-value, a strong indication that the medians are significantly different.

*The z-table here is to the right of the mean, so I had to double the actual result I found (-4.940).

Se $t > t_\alpha$ o $t < -t_\alpha$ posso negare la null hypothesis e concludere che c'è una differenza statistica fra H_1 e H_2 . Altrimenti, H_1 e H_2 sono statisticamente uguali.

Se il test se lo uso per H_1 e H_2 è diverso: non-paired test

- $t = \frac{\overline{err}(M_1) - \overline{err}(M_2)}{\sqrt{var(M_1 - M_2)/k}}$: dove $var(M_1 - M_2) = \sqrt{\frac{var(M_1)}{k_1} + \frac{var(M_2)}{k_2}}$

- degree of freedom = $\min(k_1 - 1, k_2 - 1)$

Tabella t-test

df	α = 0.1	0.05	0.025	0.01	0.005	0.001	0.0005
0	$t_0 = 1.232$	1.645	1.960	2.226	2.576	3.091	3.291
1	3.078	3.314	3.706	3.821	6.656	318.289	636.578
2	1.886	2.920	4.303	6.965	9.925	22.328	31.600
3	1.638	2.393	3.182	4.541	5.841	10.214	12.924
4	1.533	2.132	2.776	3.747	4.604	7.173	8.610
5	1.476	2.015	2.571	3.365	4.032	5.894	6.869
6	1.440	1.943	2.447	3.143	3.707	5.208	5.959
7	1.415	1.895	2.365	2.998	3.499	4.785	5.408
8	1.397	1.860	2.306	2.896	3.355	4.501	5.041
9	1.383	1.833	2.262	2.821	3.250	4.297	4.781
10	1.372	1.812	2.228	2.764	3.169	4.144	4.587

Attenzione: Quando applichiamo il T-test assumiamo che le due distribuzioni siano normali. Dovremmo applicare un test prima di procedere per capire se le distribuzioni sono effettivamente normali.

Se abbiamo degli imbalanced dataset, dovremmo usare la coppia (Recall-Precision) o la coppia (Sensitivity-Specificity). Il problema è che i test statistici visti finora non funziona per coppie di misure, ma solo per misure singole come l'accuracy e l'F-misura. Quindi o usiamo l'F-measure oppure andiamo ad utilizzare la ROC CURVE.

ROC (Receiver Operating Characteristics) Curves

La ROC curve è composta dal FPR sulle x e TPR sulle y. Essa è la visualizzazione grafica della comparazione tra due modelli di classificazione.

Un classificatore ottimo ha un basso FPR e un TPR alto.

La ROC curve ranka i modelli in un decreasing order, il modello incima alla lista è il migliore. Più si avvicinano alla linea diagonale, meno i modelli sono accurati.

L'area sotto la curva è chiamata AUC ed è la misura dell'accuratezza dei due modelli. Maggiore è l'area sotto la curva, maggiore è l'accuratezza del modello.

Come andare a confrontare i classificatori con AUC?

- Per ogni K-fold trovo, per ogni modello, (FPR, TPR)
- Collego:
 - Il punto (0,0) con (FPR_{M_1}, TPR_{M_1})
 - Il punto (FPR_{M_1}, TPR_{M_1}) con (1,1)
- E trovo l'AUC

- Uso la distribuzione delle AUC per il test statistico.

Il miglior classificatore è quello con AUC media maggiore rispetto a tutti gli altri a livello statistico.

Costo (della missclassification) del classificatore rappresentato dai punti (FPR, TPR):

$$\text{cost} = FPR \cdot P(n) \cdot c(Y, n) + FNR \cdot P(p) \cdot c(N, p)$$

$$TPR = TP/P = TP/(TP + FN) \quad FNR = FN/(FN + TP) = 1 - TPR$$

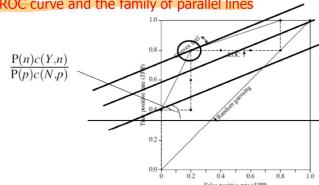
- $P(n)$ and $P(p)$: a-priori probabilities of a negative example and a positive example
- $c(Y, n)$ and $c(N, p)$: false positive cost and false negative cost

- Once fixed the values of $P(n)$, $c(Y, n)$, $P(p)$ and $c(N, p)$, we can obtain a family of parallel lines (called iso-cost lines) with slope

$$\frac{P(n)c(Y, n)}{P(p)c(N, p)}$$

The points belonging to the same line have the same cost, and the cost decreases as we move to parallel lines closer to the point (0,1), i.e., more north-west.

Point that minimizes the classification cost: tangent point between the ROC curve and the family of parallel lines



Il p-value indica la confidenza usata per dire che le due mediane sono significativamente differenti



Tecniche per migliorare la classification accuracy

Per migliorare l'accuracy si usano gli **Ensemble Methods** i quali combinano un insieme K di modelli addestrati (M_1, M_2, \dots, M_k) con l'obiettivo di creare un modello migliorato M^* .

Bagging method

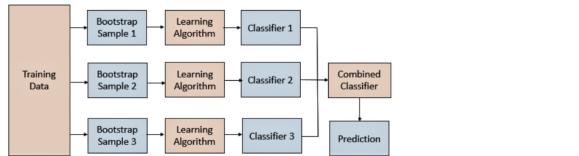
Fare la media tra le predizioni dell'insieme di modelli.

Addestramento

Dato un insieme D di d tuple, ad ogni iterazione i, un training set D_i di d tuple viene sampled con rimpiazzamento da D. Un classificatore M_i è addestrato dal training set D_i .

Classificazione

Ogni classificatore M_i ritorna una class prediction. Vengono contati i "voti" dei vari modelli e viene assegnata a X la classe con il maggior voto.



Può essere usato anche per predirre un valore continuo prendendo la media dei valori predetti dai vari classificatori.

Boosting

Vengono presi in considerazione le predizioni di tutti i modelli, ma hanno un peso. Il peso è determinato dall'accuracy di quel modello.

- 1) I pesi sono assegnati ad ogni tupla.
- 2) Una serie di k classificatori viene addestrata iterativamente, delle tuple.
- 3) Dopo che un classificatore M_i è stato addestrato, i pesi vengono aggiornati in modo da permettere al classificatore seguente, M_{i+1} , di fare più attenzione alla tupla che era stata classificata male da M_i .
- 4) Il modello finale M^* combina il voto di ogni classificatore individuale, dove il peso del voto di ogni classificatore è in funzione della sua accuratezza.

Il metodo Boosting tende ad avere una maggiore accuratezza, ma ha il rischio di overfittare il modello portando a un sbagliata classificazione dei dati.

AdaBoost

Dato un insieme di d tuple $(x_1, y_1) \dots (x_d, y_d)$

- 1) tutti i pesi delle tuple vengono settati inizialmente a $1/d$
- 2) y_n K rounds vengono generati K classificatori.

Ad ogni round (round i):

- a) D_i tuples vengono sampled da D (con rimpiazzamento) dove la probabilità di una tupla di essere presa è in base al suo peso.
- b) Viene costruito un classification model H_i (derivato da D_i) e viene calcolato il suo error rate usando D_i come test set.

L'error rate è uguale alla somma dei pesi delle tuple missclassified:

$$\text{error}(M_i) = \sum_{j=1}^d w_j \times \text{err}(X_j) \quad \text{dove: } \text{err}(X_j) = \text{missclassification error per } X_j$$

- c) Se una tupla (x_j, y_j) viene correttamente classificata, il suo peso viene aggiornato come segue:

$$w_j = w_j \cdot \text{error}(M_i) / (1 - \text{error}(M_i))$$

- d) Successivamente, il peso di TUTTE le tuple viene normalizzato moltiplicando il peso per la somma dei vecchi pesi e dividendo per la somma dei nuovi pesi.

In questo modo:

- Peso delle tuple classificate correttamente : diminuito
- Peso delle tuple missclassified : aumentato

- 3) Viene successivamente assegnato un peso al voto dei classificatori basato sull'error rate del classificatore ($< \text{error rate} \Rightarrow > \text{weight}$)

Il peso al voto viene assegnato come segue:

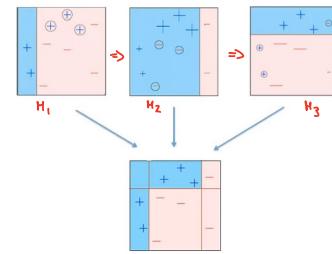
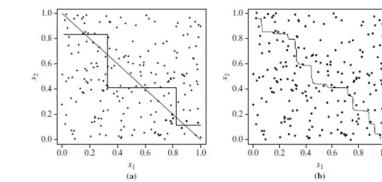
$$\log \frac{1 - \text{error}(M_i)}{\text{error}(M_i)}$$

- 4) Per ogni classe c sommiamo i pesi di ogni classificatore che ha assegnato la classe c a X. La classe con la somma maggiore è la vincente ed è assegnata come "class prediction" di X.

Accuracy

- Often significantly better than a single classifier derived from D
- For noise data: not considerably worse, more robust
- Proved improved accuracy in prediction

Figure (a) shows the decision boundary of a decision tree classifier on the problem. Figure (b) shows the decision boundary of an ensemble of decision tree classifiers on the same problem.



BAGGING	ADABOOST
Resample dataset	Resample or reweight dataset
Builds base models in parallel	Builds base models sequentially
Reduces variance (doesn't work well with e.g., decision stumps)	Also reduces bias (works well with stumps)

Random Forest (Bagging method)

Ogni classificatore è un decision tree classifier ed è generato usando una selezione random degli attributi ad ogni nodo per determinare lo split.

Ogni albero vota e la classe più popolare viene restituita.

Classical Algorithms

Let N and M be the number of training instances and attributes, respectively. Let m be the number of attributes to be used for choosing the decision attribute at any node

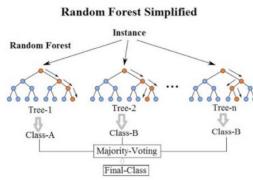
- Choose a training set by randomly extracting N samples with replacement from all available training instances (i.e. take a bootstrap sample). Use the rest of the instances to estimate the ensemble error.
- For each node of the tree, randomly choose m attributes on which to base the decision at that node. Calculate the best split based on these m variables in the training set.
- Each tree is fully grown and not pruned (as may be done in constructing a normal tree classifier).

m è un sottoinsieme degli attributi totali.

la scelta dell'albero lo split viene fatta su questi attributi.

Advantages:

- RF is one of the most accurate learning algorithms available. For many data sets, it produces a highly accurate classifier.
- RF runs efficiently on large databases.
- RF can handle thousands of input variables without variable deletion.
- RF gives estimates of what variables are important in the classification.
- RF generates an internal unbiased estimate of the generalization error as the forest building progresses.
- RF has an effective method for estimating missing data and maintains accuracy when a large proportion of the data are missing.



Classificazione di un Class-imbalanced Dataset

I modelli che usano la differenza dell'errore tra il label e la classe predetta durante il training soffrono molto il class-imbalanced dataset perché vanno a pesare maggiormente l'errore sulla numerosità class e in modo minore quello sulla minority class.

Questa situazione si ha quando molte istanze appartengono ad una classe poche ad un'altra.

I metodi tipici per una classificazione a 2-classi sono:

- Oversampling:** re-sampling of data from positive class
- Under-sampling:** randomly eliminate tuples from negative class
- Threshold-moving:** moves the decision threshold, t, so that the rare class becomes easier to classify, and hence, less chance of costly false-negative errors
- Ensemble techniques:** Ensemble multiple classifiers introduced above

Rimane difficile per un class-imbalance su un problema a classi multiple

ATTENZIONE: Il bilanciamento DEVE essere applicato SOLO sul training set. Il test set non deve essere bilanciato!

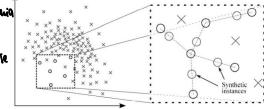
Questo perché lavoriamo sotto l'assunzione che conosciamo SOLO il training set.

SMOTE (Synthetic Minority Over-Sampling Technique)

SMOTE è una oversampling technique la quale prende in considerazione la classe con meno istante e considera un parametro K che indica il numero di vicini da prendere in considerazione.

Collega con un segmento un'istanza della classe minoritaria con K altre istanze della stessa classe.

Sui segmenti, crea randomicamente nuove istanze della classe minoritaria. ("crea istanze artificiali")



Clustering

L'operazione di clustering si concentra nell'andare a dividere le istanze in gruppi (cluster) che sono simili tra loro e diverse dalle istanze degli altri cluster (questa relazione di similarità e dissimilarità quantifica anche la qualità di un clustering).

Per fare ciò si usano delle tecniche di addestramento non supervisionato.

Questa tecnica viene principalmente usata come: Strumento stand-alone per avere informazioni sulla distribuzione dei dati, oppure, come passo di preprocessing per un altro algoritmo.

Gli approcci (metodi) di clustering più usati sono:

Method	General Characteristics
Partitioning methods	<ul style="list-style-type: none"> - Find mutually exclusive clusters of spherical shape - Distance-based - May use mean or medoid (etc.) to represent cluster center - Effective for small- to medium-size data sets
Hierarchical methods	<ul style="list-style-type: none"> - Agglomerative (bottom-up) or divisive (top-down) (i.e., multiple levels) - Cannot correct erroneous merges or splits - May incorporate other techniques like microclustering or consider object "linkages"
Density-based methods	<ul style="list-style-type: none"> - Can find arbitrarily shaped clusters - Clusters are dense regions of objects in space that are separated by low-density regions - Cluster centers: Each point must have a minimum number of points within its neighborhood - May filter out outliers
Grid-based methods	<ul style="list-style-type: none"> - Use a multi-resolution grid-data structure - Fast processing time (typically independent of the number of data objects, yet dependent on grid size)

Prima di applicare uno dei metodi, bisogna valutare se il dataset contiene cluster (strutture non random) o no.

Per fare ciò si usa il "Clustering Tendency Assessment" = Hopkins statistic

Hopkins statistic

Statistica spaziale utilizzata per valutare la Clustering Tendency di un insieme di dati, misurando la probabilità che un dato dataset è generato da un distribuzione uniforme di dati. In altre parole, testa la randomità dello spazio dei dati. Se la distribuzione è uniforme, vuol dire che non ho cluster significativi.

- Sia D il dataset di N valori, siano n < N punti p_1, \dots, p_n sampled uniformemente da D. Per ogni punto p_i trova il più vicino in D. Sia x_i la distanza tra p_i e il suo vicino in D.

$$x_i = \min_{v \in D} \{dist(p_i, v)\}$$

3) Calcola la Hopkins Statistic

$$H = \frac{\sum_{i=1}^n x_i}{\sum_{i=1}^n x_i + \sum_{i=1}^n y_i}$$

- If D were uniformly distributed, then the two terms at the denominator would be close to each other and therefore H would be about 0.5.
- In presence of clustering tendency, then the first term at the denominator would be smaller than the second and then H will increase.

Random uniform distribution

- Genera un dataset simulato ($random_d$) preso da una distribuzione uniforme randomica con n punti q_1, \dots, q_n e la stessa variazione di D, cioè genero punti fittizi.

Per ogni punto q_i trova il più vicino in D. Sia y_i la distanza tra q_i e il suo vicino in D.

$$y_i = \min_{v \in D \setminus \{q_i\}} \{dist(q_i, v)\}$$

- L'ipotesi nulla e alternativa sono definite come segue:

- Null hypothesis = Il dataset D è uniformemente distribuito (non ci sono cluster significativi)
- Alternative hypothesis = Il dataset D non è uniformemente distribuito (contiene cluster significativi)

- La Hopkins Statistic è calcolata per diverse selezioni randomi di punti e la media di tutti i risultati (H) è usata per decidere. Se H è maggiore di 0.75, allora indica una clustering tendency al 90% confidence level.

Partitioning Methods

Trova un punto e il centroide relativo

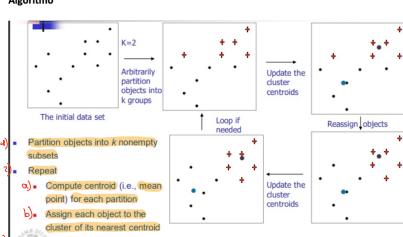
Partitionano il dataset D di N oggetti in un insieme di k cluster, tale che la somma quadratica della distanza è minimizzata (dove c_i è il centroide o mediodio del cluster C_i).

Quando usiamo un partitioning methods dobbiamo fissare il parametro K, il quale indica il numero di cluster che vogliamo trovare.

Naturalmente, andare a fissare il parametro K va ad imporre diverse limitazioni sull'algoritmo.

- 1) **K-means clustering** (Ogni cluster è rappresentato dal centro del cluster) - Essendo che usiamo il centro, dobbiamo determinarlo e di conseguenza usare solo numeric features (semplici, in alternativa, usare il k-modes).

Algoritmo



OSS: L'algoritmo è influenzato dall'inizializzazione dei cluster al passo 1.
OSS_2: Troviamo sferical clusters perché usiamo l'euclidean distance

Vantaggi e svantaggi

- Strength**
 - Efficient: $O(ktn)$, where n is # objects, k is # clusters, and t is # iterations. Normally, $k, t \ll n$.
 - Comparing: PAM: $O(k(n-k)^2)$, CLARA: $O(ks^2 + k(n-k))$, where s is the size of the data sample
 - Comment: Often terminates at a local optimal
- Weakness**
 - Applicable only to objects in a continuous n-dimensional space
 - Using the k-modes method for categorical data
 - Comparison, k-medoids can be applied to a wide range of data
 - Need to specify k , the number of clusters, in advance (there are ways to automatically determine the best k)
 - Sensitive to noisy data and outliers
 - Not suitable for discovering clusters with non-convex shapes

Essendo che l'outlier deve essere aggiunto in un cluster necessariamente, va ad "attrarre" il centroide verso lui, essendo lontano dagli altri punti.

Gestire i categorical data: k-modes

- Replacing means of clusters with modes (values that occur most frequently in a dataset)
- Using new dissimilarity measures to deal with categorical objects
- Using a frequency-based method to update modes of clusters
- A mixture of categorical and numerical data: k-prototype method

Non posso usare la cost function per determinare K essendo che all'aumentare di K, la cost function diminuisce.

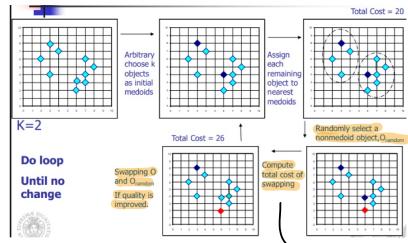
Con $K = n$ otteniamo un singolo punto per cluster e di conseguenza un errore pari a 0.

2) K-Medoids method - Versione migliorata di K-Means (Ogni cluster è rappresentato da un oggetto di esso) - Può essere usato anche con diversi tipi di features essendo che non dobbiamo determinare il centro dovuto dagli outliers.

Il problema del k-means è la sensibilità agli outliers. K-medoids sceglie random il prossimo punto (effettivo, non inferenza) e calcola il nuovo costo. Va a risolvere il problema di "attrazione"

Potrebbe non essere una riferenza il medoid ma un oggetto effettivo

PAM algorithm:



Vantaggi:

- Più robusto agli outliers e noise

Svantaggi:

- Molto costoso a livello computazionale, può lavorare solo su piccoli dataset.

Costoso perché esamina tutti i "vicini" del current node in cerca del minimo costo.

$O(k(n - k)^2)$ per ogni iterazione dove

n = numero di dati

k = numero di clusters

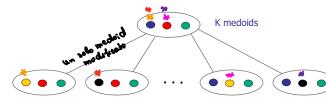
$$\text{Cost function} = \text{Absolute error calculation} = \sum_{i=1}^k |p - o_i|$$

OSS:

Quando usiamo i Partitioning Methods abbiamo due principali caratteristiche:

1) Dobbiamo fissare K prima di eseguire l'algoritmo.

2) Ottieniamo cluster di ferma convessa (sferici o ellittici).



CLARA (Clustering Large Applications) - Versione migliorata di PAM

Prende un sample del dataset originale e gli applica PAM sul sample per trovare i medoids. Se il sample è rappresentativo, i medoids del sample dovrebbero approssimare i medoids dell'intero dataset. Essendo che i medoidi vengono trovati sul sample, l'algoritmo non può trovare la migliore soluzione se uno dei k medoids non è incluso dentro il sample selezionato.

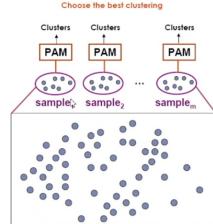
Per migliorare l'approssimazione, l'algoritmo viene ripetuto più volte selezionando sample diversi e restituendo il miglior clustering trovato tra le varie ripetizioni dell'algoritmo.

L'accuratezza del clustering è misurata dalla media della dissimilarità di tutti gli oggetti nell'intero dataset.

Algoritmo:

Vantaggi:

- Può lavorare su dataset più grandi di PAM



$$O(ks^2 + k(n - k)) \text{ dove: } S = \text{size del sample}, K = \text{numero di clusters}, N = \text{numero di oggetti}$$

Svantaggi:

- L'efficienza dipende dalla sample size
- Un buon clustering basato sui sample non è detto che rappresenti necessariamente un buon clustering sull'intero dataset se il sample è biased

CLARANS (Randomized Clara) - Versione migliorata di CLARA

Il processo di clustering può essere rappresentato come un grafo di ricerca, dove ogni nodo è potenzialmente una soluzione composta da K medoids. Due nodi sono "vicini" (neighbours) se il loro insieme di medoidi differisce SOLO DI UN medoid.

Funzionamento:

Partiamo da un nodo random del grafo composto da K medoids e analizziamo se cambiando un solo medoid dei K medoids otteniamo un miglioramento nella cost function (se i vicini di quel nodo migliorano la cost function). Se abbiamo un miglioramento, ci muoviamo verso quel nodo e facciamo lo stesso procedimento.

Analizzare l'intero numero di vicini è costoso, quindi CLARANS estrapola un sample di vicini da analizzare.

Una volta che un ottimo locale viene trovato, CLARANS può essere eseguito nuovamente, iniziando da un nuovo nodo del grafo random per trovare un nuovo ottimo locale. Il numero di ottimi locali da cercare è un parametro dell'algoritmo. In fine si trova il migliore fra gli ottimi locali trovati.

Costo: E' molto più efficiente di PAM e CLARA, la sua complessità è: $O(n)$.

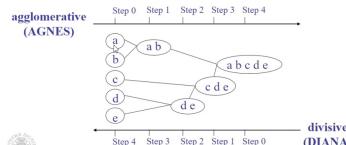
Differenza tra CLARA e CLARANS:

CLARA = Prende un sample di nodi all'inizio della ricerca, quindi restringe la ricerca a una specifica area del dataset originale. I "vicini" vengono presi dal sample.

CLARANS = Non confina la ricerca in una specifica area, i "vicini" vengono presi dall'intero dataset

Hierarchical Methods

Gli hierarchical methods creano una gerarchia di cluster. Posso avere due tipi di approcci principali:



Agglomerative: Parto da cluster composti da un solo oggetto e man mano vado a fare l'unione dei cluster per creare cluster di cardinalità più alta.

Ad ogni livello della gerarchia ho che il cluster è composto dai cluster di livello inferiore.

Idea: Faccio l'unione di due cluster se gli oggetti di due cluster diversi sono molto vicini.

Divisive: Parto da un cluster composto da tutti gli oggetti e man mano vado a fare la divisione in cluster di cardinalità minore.

Ad ogni livello della gerarchia ho che il cluster è una parte del cluster a livello inferiore.

Idea: Faccio lo split in due cluster se gli oggetti di due cluster diversi sono molto lontani.

Usano come clustering criterio la distance matrix (anche chiamata connectivity matrix). Questi metodi non richiedono in input il parametro k ma necessitano di una condizione di terminazione.

Per unire o separare sottoinsieme di punti invece che punti individuali, la distanza tra i punti individuali deve essere generalizzata alla distanza tra subset. Questa misura è chiamata Linkage metric.

Il tipo di Linkage metric influenza significativamente sul Hierarchical algorithm, dovuto al fatto che riflette il concetto particolare di chiusura e connettività.

Linkage Metrics

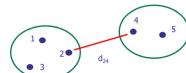
Nelle Linkage Metrics tra cluster sono incluse quelle: Single Link, Average Link e Complete Link.

La misura di dissimilità (solitamente la distanza) è misurata per ogni paio di punti, uno nel primo insieme e gli altri punti del secondo insieme.

$$d(C_1, C_2) = \text{operation}\{d(x, y) \mid x \in C_1, y \in C_2\}$$

Single Link (nearest neighbor)

La distanza tra due cluster è determinata dalla distanza dei due oggetti più vicini dei due differenti cluster.

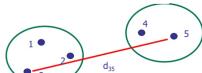


$$\text{Minimum distance: } dist_{min}(C_1, C_2) = \min_{p \in C_1, p' \in C_2} \{|p - p'||$$

Nearest-neighbor clustering algorithm:
L'algoritmo di clustering termina quando la minima distanza tra i cluster più vicini supera una certa soglia definita dall'utente.

Complete Link (furthest neighbor)

La distanza tra due cluster è determinata dalla distanza dei due oggetti più lontani dei due differenti cluster.

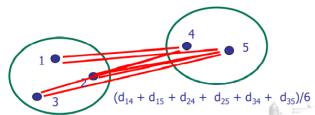


$$\text{Maximum distance: } dist_{max}(C_1, C_2) = \max_{p \in C_1, p' \in C_2} \{|p - p'||$$

Farthest-neighbor clustering algorithm:
L'algoritmo di clustering termina quando la massima distanza tra i cluster più vicini supera una certa soglia definita dall'utente.
Cerca di minimizzare la crescita in diametro dei cluster ad ogni iterazione (sceglie il minimo dei punti più lontani che superano una certa soglia).

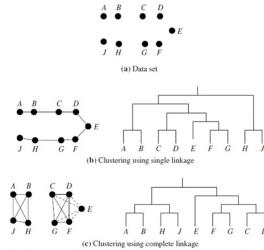
Pair-group average

La distanza tra due cluster è calcolata come la distanza media tra tutte le coppie di oggetti dei due differenti cluster.



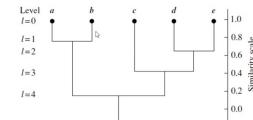
$$\text{Average distance: } \text{dist}_{avg}(C_i, C_j) = \frac{1}{n_i n_j} \sum_{p \in C_i, p' \in C_j} |p - p'|$$

Esempio:



Rappresentazione dell'esecuzione di un Hierarchical clustering.

Per rappresentare l'esecuzione di un hierarchical clustering si utilizza una struttura ad albero chiamata dendrogramma.



Vantaggi e svantaggi dell'Hierarchical Clustering

- Avere una gerarchia invece che una collezione amorfa di clusters
- Non necessita di specificare K
- In generale, restituisce cluster di qualità migliore rispetto ai metodi k-means

Svantaggi:

- Non può mai fare un "undo" di quello che ha fatto in precedenza
- Non scala bene, la complessità in tempo è di almeno $O(n^2)$ dove n è il numero di oggetti totali.
- I cluster sono di forma convessa (Sferici o ellittici)

AGNES (Agglomerative)

dai oggetti più vicini dei due clusters

Questo algoritmo usa come metodo il **single-link** e usa la **dissimilarity matrix** (posso avere solo la dissimilarity matrix invece che il dataset).

Inizialmente ogni oggetto è inserito in un cluster.

I cluster vengono uniti secondo i criteri del single-link.

Questo processo viene ripetuto fin quando tutti gli oggetti non sono uniti in un unico cluster.

	A	B	C	D	E
A	0	1	2	2	3
B	1	0	2	4	3
C	2	2	0	1	5
D	2	4	1	0	3
E	3	3	5	3	0

$$d = 1 \\ \begin{matrix} A \\ \cup \\ B \\ \cup \\ C \\ \cup \\ D \end{matrix}$$



Recompute the distance matrix

	AB	CD	E
AB	0	2	3
CD	2	0	3
E	3	3	0

$$d = 2 \\ (A \cup B) \cup (C \cup D)$$

	ABCD	E
ABCD	0	3
E	3	0

$$d = 3 \\ (ABCD) \cup (E)$$

DIANA (Divisive)

Questo algoritmo costruisce una gerarchia di cluster, iniziando da uno più grande contenente n esempi. I cluster vengono divisi fin quando ogni cluster non contiene un singolo esempio.

- 1) Ad ogni passo, il cluster con la dissimilarità maggiore tra due esempi al suo interno, viene selezionato. (DIAMETRO)
- 2) Per dividere il cluster selezionato, l'algoritmo prima cerca quale esempio ha la dissimilarità più grande rispetto agli altri esempi all'interno del cluster selezionato. L'esempio trovato inizializza lo "splinter group".
- 3) Nei passi successivi, l'algoritmo riassegna gli altri esempi allo "splinter group" se sono vicini all'esempio trovato precedentemente.
- 4) Il risultato è la divisione del cluster in due nuovi cluster.
- 5) Ripeti dal punto 1 scegliendo il cluster con diametro maggiore.

We need to find the most dissimilar element, thus we calculate the mean for each row:

Ex: $d(A_0) = (0.129 + 0.518 + 0.729 + 1.532 + 1.887 + 1.989 + 2.025) / 7 = 1.258$

A ₀	X ₁	X ₂	X ₃	X ₄	X ₅	X ₆	X ₇	X ₈
d(A ₀)	1.258	1.147	0.925	0.865	0.865	0.966	1.025	1.056

$A_1 = \{2, 3, 4, 5, 6, 7, 8\}$	X ₆ X ₇ X ₈ X ₅ X ₄ X ₃ X ₂	X ₁
$B_1 = \{1\}$		

We need to find the most dissimilar element for the cluster A₁, thus we calculate the mean for each row (the X₁-column and X₁-row are not considered):

Ex: $d(1, A_1) = (0.389 + 0.6 + 1.403 + 1.758 + 1.86 + 1.896) / 6 = 1.318$

A ₁	X ₂	X ₃	X ₄	X ₅	X ₆	X ₇	X ₈
d(1, A ₁)	1.318	0.993	0.889	0.754	0.813	0.864	0.894

In this case: $d(X_2, A_1) = 1.318$ and $d(X_2, B_1) = 0.129$

verifico se la dissimilarità è maggiore per A o B

$A_2 = \{3, 4, 5, 6, 7, 8\}$	X ₆ X ₇ X ₈ X ₅ X ₄ X ₃	X ₁ X ₂
$B_2 = \{1, 2\}$		

assegno al cluster con dissim. minore

stefi Df...

And so on:

A ₂	X ₃	X ₄	X ₅	X ₆	X ₇	X ₈
d(i,A ₂)	1.114	0.946	0.624	0.624	0.665	0.694

B ₂	X ₁	X ₂
d(i,B ₂)	0.129	0.129

In this case: d(X₃,A₂) = 1.114 and d(X₃,B₂) = 0.453, thus the element X₃ is included in B₂.

$A_3 = \{4,5,6,7,8\}$	$X_6 \quad X_7 \quad X_8 \quad X_5 \quad X_4$	$X_1 \quad X_2 \quad X_3$
$B_3 = \{1,2,3\}$		

⋮

In order to choose which cluster to divide,

$$\text{diam. } A_4 = \max d(i,j) = d(5,8) = 0.493 \quad i,j \in A_4$$

$$\text{diam. } B_4 = \max d(i,j) = d(1,4) = 0.729 \quad i,j \in B_4$$

Diam B₄ > diam A₄, thus we are going to divide the cluster B₄

B ₄	X ₁	X ₂	X ₃	X ₄
d(i,B ₄)	0.459	0.373	0.373	0.513

$A_5 = \{5,6,7,8\}$	$X_6 \quad X_7 \quad X_8 \quad X_5$	X_4	$X_1 \quad X_2 \quad X_3$
$B_5 = \{1,2,3\}$			
$C_5 = \{4\}$			

Estensioni dell'Hierarchical Clustering

BIRCH (Balanced Iterative Reducing and Clustering Using Hierarchies)

Consiste in un agglomerativo clustering disegnato per clustering con un grosso ammontare di dati numerici.

Il costo in complessità di BIRCH è $O(n)$ dove n è il numero di oggetti che devono essere raggruppati. Prima di eseguire l'algoritmo, fissiamo il diametro massimo che può avere un cluster e il limite di entry che può avere un nodo.

A ₃	X ₁	X ₅	X ₆	X ₇	X ₈
d(i,A ₃)	1.129	0.527	0.438	0.464	0.491

B ₃	X ₁	X ₂	X ₃
d(i,B ₃)	0.323	0.259	0.453

In this case: d(X₁,A₃) = 1.129 and d(X₃,B₃) = 0.513, thus the element X₄ is included in B₃.

$A_4 = \{5,6,7,8\}$	$X_6 \quad X_7 \quad X_8 \quad X_5 \quad X_4$	$X_1 \quad X_2 \quad X_3 \quad X_4$
$B_4 = \{1,2,3,4\}$		

B ₆	X ₁	X ₂
d(i,B ₆)	0.129	0.129

C ₆	X ₃	X ₄
d(i,C ₆)	0.211	0.211

The maximum distance is d(X₅,A₆) = 0.435 and d(X₅,B₆) = 1.467 and d(X₅,C₆) = 1.817; thus the element X₅ can not be included in B or C, and the process stops.

$$\text{diam. } A_6 = \max d(i,j) = d(5,8) = 0.493 \quad i,j \in A_6$$

$$\text{diam. } B_6 = \max d(i,j) = d(1,2) = 0.129 \quad i,j \in B_6$$

$$\text{diam. } C_6 = \max d(i,j) = d(3,4) = 0.211 \quad i,j \in C_6$$

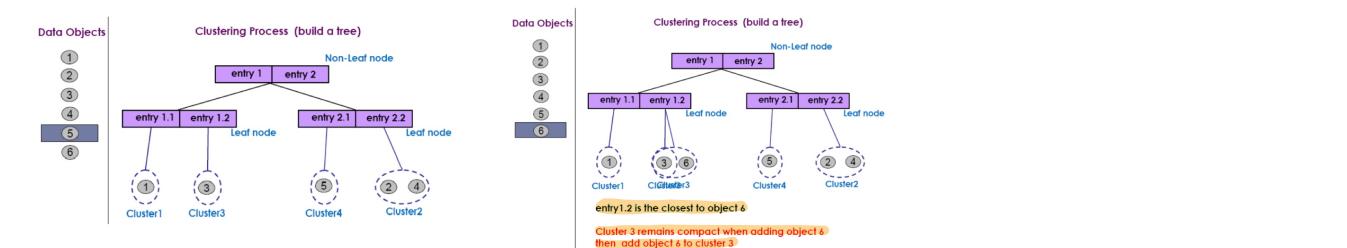
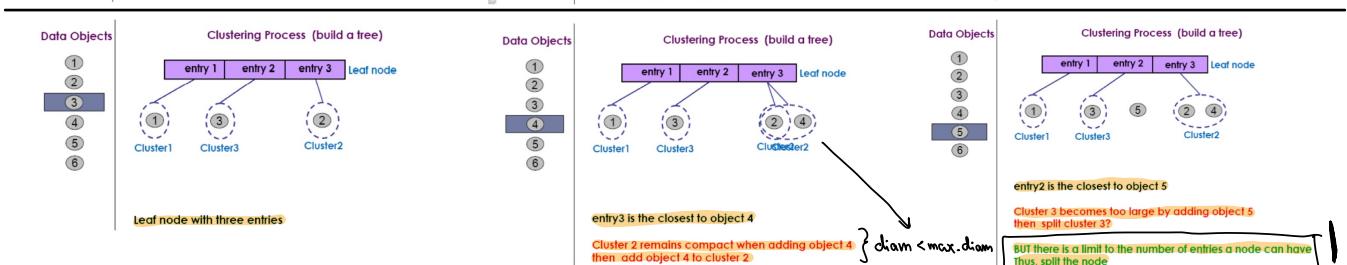
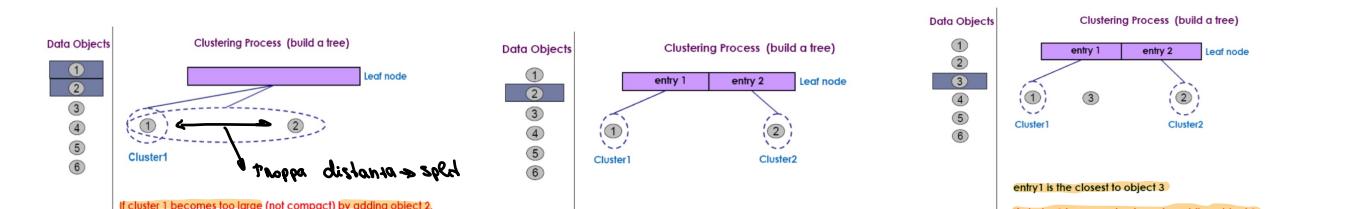
diam. A₆ > diam. C₆ > diam. B₆, thus we are going to divide the cluster A₆ creating the cluster[5]

The final result is:

$A_7 = \{7\} = 1.681$	X_8	X_7	X_6	X_5	X_4	X_3	X_2	X_1
$B_{11} = \{1\} = -0.308$								
$C_{11} = \{3\} = 0.210$								
$D_{11} = \{4\} = 0.421$								
$E_{11} = \{2\} = -0.179$								
$F_{11} = \{5\} = 1.224$								
$G_{11} = \{6\} = 1.579$								
$H_{11} = \{8\} = 1.717$								

Cerca di risolvere i seguenti problemi:

- Considerare che il dataset può essere troppo grande per fittare nella memoria principale
- Minimizzare il numero di scansioni del dataset
- Ridurre il costo delle operazioni di I/O



BIRCH, componenti principali

1) Clustering Feature

Per calcolare i centroidi, e misurare la compatezza e la distanza dei cluster, BIRCH utilizza la "Clustering Feature".

La Clustering Feature contiene il riassunto delle statistiche per un dato cluster e ha informazioni sufficienti per calcolare: Centroide, Raggio, Diametro e tante altre misure.

Centroid: the "middle" of a cluster

$$x_0 = \frac{\sum_{i=1}^n x_i}{n} \equiv \frac{LS}{n}$$

Clustering Feature (CF) = (N, LS, SS) dove:

N = Numero di data points

LS = Somma lineare di N punti

SS = Somma quadratica di N punti

Esempio:

$$\text{CF}_3 = (3+3, (9+35, 10+36), (29+417, 38+440)) = (6, (44,46), (446,478))$$

Cluster 1: (2.5, 3.2, 4.3)
Cluster 2: (3, 3.2, 4.3)
Cluster 3: (9, 10, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100)
$$\text{CF}_1 = (3, (2+3+4, 5+2+3), (25+32+45, 54+52+32)) = (3, (9,10), (29,38))$$

Radius: square root of average distance from any point of the cluster to its centroid

$$R = \sqrt{\frac{\sum_{i=1}^n (x_i - x_0)^2}{n}} = \sqrt{\frac{nSS - 2LS^2 + nL^2S^2}{n^2}}$$

Diameter: square root of average mean squared distance between all pairs of points in the cluster

$$D = \sqrt{\frac{\sum_{i=1}^n \sum_{j=1}^n (x_i - x_j)^2}{n(n-1)}} = \sqrt{\frac{2nSS - 2LS^2}{n(n-1)}}$$

Osservazione: Il CF può essere calcolato man mano che arrivano i dati in streaming, teoricamente possiamo lavorare con milioni di dati andando ad aggiornare il CF man mano che arrivano i dati, senza salvarli in memoria.

2) CF-Tree

Il CF-tree è un albero bilanciato in altezza.

Ha due parametri:

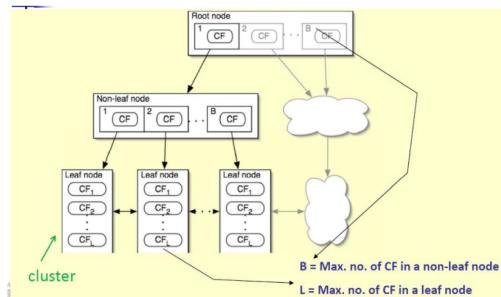
- Numero di entries in ogni nodo = Max_entries dei nodi
 - Diametro di tutte le entries in un nodo foglia = Max_diametro dei cluster
- Le foglie sono connesse tramite i puntatori "precedente" e "successivo".

Conservano implicitamente la dimensione del CF tree.

Esso serve le clustering features per un hierarchical clustering.

Parametri:

B = Branching factor	Specifica il numero massimo di figli per un nodo non foglia
T = Parametro soglia	Specifica il diametro massimo dei subcluster salvati ai nodi foglia dell'albero
L	Numero massimo di entries in una foglia
CF in un nodo padre	Somma delle CF di un nodo figlio



Osservazioni BIRCH

- Un leaf node rappresenta un cluster
- Un sub-cluster in un leaf node deve avere il diametro non più grande della soglia T .
- Un punto è inserito in un leaf node (cluster) con il quale esso è vicino
- Quando un item viene inserito in un cluster (leaf node), la restrizione T deve essere soddisfatta. Il corrispondente CF e l'CF di tutta la catena (nodi padre) devono essere aggiornati.
- Se non c'è spazio nel nodo, il nodo viene splitato.
- Possiamo usare i cluster trovati così come sono, oppure, usare un algoritmo di clustering arbitrario su quei cluster trovati.

Vantaggi e Svantaggi

Vantaggi

- Trova un buon clustering con una singola scan del Dataset e migliora la qualità del clustering con poche scan aggiuntive
- Complessità $\mathcal{O}(n)$

Svantaggi

- Gestisce solo dati numerici (perché deve calcolare sum e mean) ed è sensibile all'ordinamento dei dati (perché la costruzione dell'albero dipende dall'ordine degli oggetti).
- La grandezza dei nodi foglia viene fissata, quindi i cluster potrebbero non essere naturali.
- I cluster tendono a essere sferici date le misure di raggio e diametro (si trovano cluster convessi).

BIRCH Algorithm

1) Scansiona il DB per costruire un CF-tree iniziale

a) Sceglie una soglia (T e B) iniziale ed inizia a inserire i data points uno per volta, splitando se necessario.

b) Se durante il punto a la dimensione del CF tree eccede la dimensione della main memory, aumenta il valore della soglia e ricostuisce un nuovo albero partendo dalle foglie del vecchio albero (senza dover rileggere gli obj dal db)

c) Gestire gli outliers: dovrebbero componere cluster da un singolo elemento, quindi facili da rilevare

2) Costruire un albero più piccolo aumentando la soglia del vecchio albero

3) Applicare un clustering algorithm ai subclusters nelle foglie del CF-tree

4) Scansionare i cluster per fare labeling dei data points e outlier handling

OSS: 9 punti: 2 e 4 sono optional.

CHARMELEON

È un hierarchical clustering algorithm nel quale la cluster similarity viene misurata basandosi su:

- 1) Quanto bene sono connessi gli oggetti all'interno di un cluster (interconnectivity)
- 2) La vicinanza dei cluster

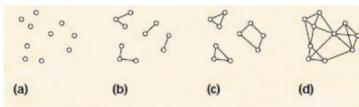
Questo vuol dire che due cluster vengono uniti se hanno un'alta interconnettività e se sono vicini.

Questa caratteristica fa sì che CHARMELEON possa automaticamente adattarsi alle caratteristiche interne dei cluster da unire.

CHARMELEON PHASES

PASSO 1) Costruisce un KNN graph (connette ogni data obj con i k obj più vicini), il quale è un grafo sparsò (la sparsità dipende dalla scelta di k)

- b -> $k=1$
- c -> $k=2$
- d -> $k=3$



OSS: Gli archi sono pesati per riflettere la similarità fra gli oggetti.

PASSO 2) Viene usato un graph partition algorithm per partizionare il KNN graph in tanti cluster (relativamente piccoli) che minimizzano l'edge cut.

Cioè:

- a) Inizialmente tutti i nodi del KNN graph appartengono a un unico cluster
- b) Iterativamente viene selezionato il più grande subcluster (quello con più elementi) e suddiviso con il graph partition algorithm

Graph-Partitioning algorithm

Un cluster C è partizionato in due subcluster C_i e C_j in modo da minimizzare la somma dei pesi degli archi che sarebbero tagliati (edge cut) se C dovesse essere diviso in C_i e C_j .

Vincolo del taglio: Ognuno di questi sub-cluster contiene almeno il 25% dei nodi contenuti in C .

Il taglio degli archi (edge cut) $EC_{\{C_i, C_j\}}$ misura l'assoluta interconnettività tra i subcluster C_i e C_j , cioè la somma del peso degli archi che connettono i vertici in C_i con i vertici in C_j .

- c) Il processo termina quando il più grande subcluster contiene meno di

uno specifico numero di vertici (4-5%)

PASSO 3) Viene usato un agglomerative hierarchical clustering algorithm che iterativamente unisce i subcluster generati al passo 2, basandosi sulla similarità.

La similarità viene misurata tra ogni paio di cluster (C_i, C_j) basandosi su due metà:

- Relative Interconnectivity (RI) tra due cluster (C_i, C_j) è definita come l'absolute interconnectivity ($EC_{\{C_i, C_j\}}$) tra (C_i, C_j) nonnormalizzata rispetto all'interconnectivity dei due cluster (EC_{C_i} e EC_{C_j})

$$RI(C_i, C_j) = \frac{|EC_{\{C_i, C_j\}}|}{\frac{1}{2}(|EC_{C_i}| + |EC_{C_j}|)},$$

OSS: l'interconnectivity di un cluster è la minima somma dei pesi degli archi che vengono tagliati per partitionare il cluster in due parti uguali.

- Relative Closeness (RC) tra due cluster (C_i, C_j) è definita come l'absolute closeness ($\bar{S}_{EC_{\{C_i, C_j\}}}$) tra (C_i, C_j) nonnormalizzata rispetto all'internal closeness dei due cluster ($\bar{S}_{EC_{C_i}}$ e $\bar{S}_{EC_{C_j}}$)

$$RC(C_i, C_j) = \frac{\bar{S}_{EC_{\{C_i, C_j\}}}}{\frac{|C_i|}{|C_i|+|C_j|}\bar{S}_{EC_{C_i}} + \frac{|C_j|}{|C_i|+|C_j|}\bar{S}_{EC_{C_j}}}$$

OSS:

- $\bar{S}_{EC_{\{C_i, C_j\}}}$ è l'avg weight degli archi che connettono C_i e C_j
- $\bar{S}_{EC_{C_i}}$ ($\circ \bar{S}_{EC_{C_j}}$) è l'avg weight degli archi che appartengono al min-cut bisection del cluster C_i ($\circ C_j$)

- a) L'algoritmo visita ogni cluster C_i e verifica se c'è un cluster C_j che soddisfa le seguenti condizioni:

$$RI(C_i, C_j) \geq T_{RI} \quad \text{e} \quad RC(C_i, C_j) \geq T_{RC}$$

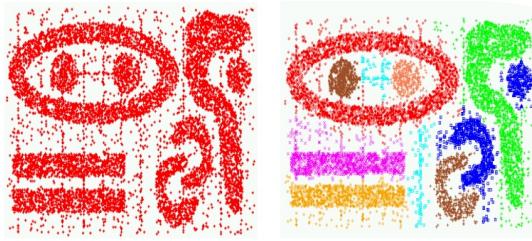
dove T_{RI} e T_{RC} sono soglie specificate dall'utente.

- b) Se più di un cluster soddisfa le condizioni, CHAMELEON sceglie di unire

C_i con il cluster con il quale ha l'absolute interconnectivity maggiore.

Vantaggi

- È più bravo nel scoprire cluster che hanno diverse forme rispetto a BIRCH e DBSCAN, non solo forme convesse.
- La complessità è $O(n^2)$ nel peggior dei casi, dove n è il numero di oggetti.
- Dipende da tre parametri fissati: K , T_{RI} e T_{RC}



Original dataset

Chameleon

Density-Based Clustering Methods

Sono algoritmi di clustering basati sulla densità.

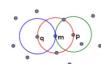
Parametri usati per definire la densità:

I parametri usati sono due:

EPS - neighborhood	Massimo raggio del vicinato
MinPts	Minimo numero di punti in un EPS-neighborhood di quel punto

Se l'EPS-neighborhood (N_{eps}) di un oggetto contiene almeno un numero minimo di oggetti, MinPts, allora quell'oggetto è chiamato **core object**.

Example: $eps = 1 \text{ cm}$, $MinPts = 3$



• Directly Density - Reachable

Un oggetto è directly density-reachable da O se:

1) O è un core object ($|N_{eps}(O)| \geq MinPts$)

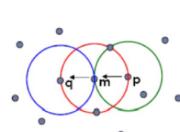
2) L'oggetto è contenuto dentro il raggio del vicinato di O (EPS-neighborhood)

Example:
• q is directly density-reachable from m
• m is directly density-reachable from p and vice versa

• Density - Reachable

Un oggetto p è density-reachable da un oggetto q se esiste una catena di oggetti:

p_1, \dots, p_n (dove $p_1 = q$ e $p_n = p$) t.c. p_{i+1} è directly density-reachable da p_i .



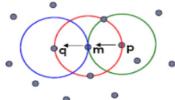
Example

- q is density-reachable from p because q is directly density-reachable from m and m is directly density-reachable from p
- p is not density-reachable from q because q is not a core object

• Density connected

Un oggetto p è density-connected a un oggetto q se esiste un oggetto o t.c. p e

q sono entrambi density-reachable da o



Example:
• p, q and m are all density-connected

Da queste definizioni otteniamo:

- 1) La densità di un oggetto viene misurata sulla base di quanti oggetti ha vicino (cioè sono nel suo vicinato)
- 2) Il clustering task consiste nell'usare i core objs e i loro vicinati per formare dense regions, cioè clusters.

Un cluster è quindi definito come un maximal set di density connected points.

- 3) Esistono tre tipi di oggetti:

- Core objects
- Border objects: Determinano il bordo di un cluster
- Outliers: Oggetti che non appartengono a nessun cluster

DB SCAN

È un algoritmo di clustering density-based.

- 1) Marca tutti gli oggetti in D come "unvisited"
- 2) Seleziona randomicamente un oggetto "unvisited" p e:
 - a) lo marca come "visited"
 - b) Controlla se il vicinato contiene almeno MinPts objects
 - No \Rightarrow Il punto viene marcato come "noise point"
 - Sì \Rightarrow Passo 3
- 3) a) Un nuovo cluster C viene creato per p e Tutti gli oggetti nel vicinato di p vengono aggiunti a un candidate set N
 - b) Iterativamente, gli obj vengono estinti da N e:
 - Aggiungi in C se non appartengono a nessun cluster
 - Marcati come "visited"

- Viene controllato se il loro vicinato ha almeno MinPts objects e in corso potranno quegli oggetti aggiunti al candidato set N .
- c) Quando N è vuoto, il cluster C è completato e viene restituito in output
 - 4) Per trovare gli altri cluster riparte dal punto (2).
 - 5) L'algoritmo termina quando tutti gli obj sono stati visitati.

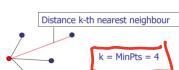
OSSERVAZIONI:

- DBSCAN adotta la closure of density-connectedness, cioè tutti i punti density connected appartengono allo stesso cluster e non esistono punti al di fuori del cluster che siano density connected ai punti all'interno del cluster.
- Complessità: $O(n^2)$
- DBSCAN è molto sensibile ai parametri EPS e minPTS, inoltre, possiamo identificare ottimamente i cluster solo se hanno tutti densità simili (perché sotto EPS e minPTS globalmente e non per parti specifiche dello spazio dove la densità può cambiare notevolmente).

Come determinare EPS e Minpts

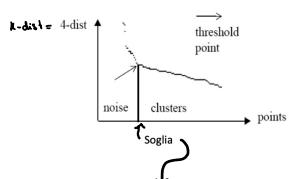
Per determinare i due parametri si usa un approccio euristico.

Dato un certo k , si definisce una funzione $k\text{-dist}$, la quale mappa ogni punto con la distanza con il suo k -th nearest neighbor. Minore è la distanza, più alta è la probabilità che i punti appartengano a una regione densa.



$k\text{-dist}(\text{punto})$ = Distanza tra il punto e il suo k -th punto vicino

Ordiniamo i punti in ordine decrescente rispetto ai loro valori $k\text{-dist}$, il grafico di questa funzione restituisce qualche suggerimento riguardo alla density distribution.
Se sceglieremo un punto arbitrario p , impostiamo il parametro Eps uguale a $k\text{-dist}(p)$ e settiamo il parametro MinPts uguale a k , tutti i punti con un uguale o minore valore di $k\text{-dist}$ sarà un core point.
I punti a sinistra della soglia possono essere border o outliers.



La soglia è data dalla prima "valley" del grafico.
tutti i punti a destra sono core points.
i punti a sinistra sono noise points

DBSCAN assume densità costante dei clusters nello spazio

Problema: Pur usando l'approccio euristico, l'intrinsic cluster structure non può essere caratterizzata da parametri di densità globali.

Soluzione: Eseguire un algoritmo (OPTICS) il quale produce un ordine speciale del dataset che rispetta la sua density based clustering structure. Con questo ordine, gli oggetti in un cluster più denso sono listati vicini agli altri nel clustering order.
Oltre a questo, il clustering ordering può essere usato anche per estrarre informazioni di base sul clustering (cluster centers) e fornire una visualizzazione del clustering.



OPTICS (Ordering Point To Identify the Clustering Structure)

E' un cluster-ordering method, il quale produce un ordine speciale del dataset riguardo alla sua density-based clustering structure.

Dopo l'esecuzione di OPTICS non abbiamo dei veri e propri clusters ma abbiamo un ordine dei punti che ci permette di determinare i valori per ottenere un buon clustering.

Salvo l'ordine nel quale gli oggetti vengono processati e le informazioni le quali verranno utilizzate da un algoritmo DBSCAN esteso per assegnare la cluster membership.

Questa informazione consiste in solo due valori per ogni oggetto: la core-distance e la reachability-distance.

Core-distance di un oggetto p

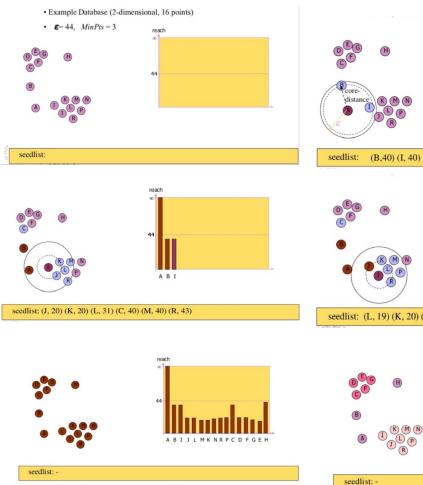
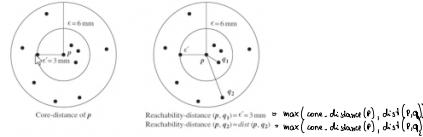
E' il più piccolo valore ϵ che fa sì che p sia un core object. Se p non è un core object, la core-distance è indefinita.

Reachability-distance dell'oggetto q da p

E' il minimo raggio che fa sì che q sia density-reachable da p. p deve essere un core object e q deve essere nel vicinato di p. La reachability-distance tra p e q è: $\max(\text{core_distance}(p), \text{dist}(p, q))$.

Se p non è un core object, la reachability-distance è indefinita.

L'oggetto q può essere directly reachable da più core objects (quindi avere più reachability-distances rispetto ai differenti core obj). Viene scelta la reachability-distance di p più piccola (perché ci da il path più corto per il quale p è connesso ai dense cluster).



Algoritmo:

Per costruire le differenti partizioni simultaneamente, gli oggetti devono essere processati in uno specifico ordine:

- 1) OPTICS inizia con un oggetto arbitrario p dal dataset in input. Restituisce gli N_{eps} di p , determina la core-distance, e setta la reachability-distance a indefinito.
- 2) Se p è un core object, allora per ogni oggetto q contenuto in N_{eps} di p , OPTICS aggiorna la reachability-distance da p e inserisce q nella OrderSeeds list se q non è stato ancora processato.
- 3) Gli oggetti contenuti nella OrderSeeds sono ordinati secondo la loro reachability-distance rispetto al core object più vicino dal quale loro sono directly reachable. In ogni step del while-loop, un oggetto currentObject il quale ha la più piccola reachability-distance nella seed-list viene selezionato dal metodo OrderSeeds.next().
- 4) La N_{eps} di questo oggetto e la sua core-distance sono calcolate. Successivamente, l'oggetto viene semplicemente scritto nel file OrderedFile con la sua core-distance e la sua reachability-distance corrente.
- 5) L'iterazione continua fin quando l'input non viene consumato completamente e la OrderSeeds è vuota.

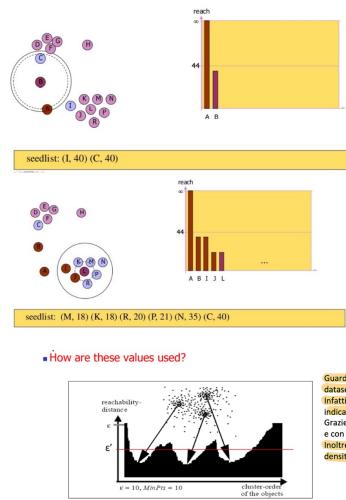
Alla fine dell'esecuzione di OPTICS, abbiamo la lista degli oggetti nell'ordine nel quale vengono processati cioè secondo la più piccola reachability distance.

Attenzione: L'epsilon che fissiamo per l'esecuzione del algoritmo OPTICS NON è lo stesso del DBSCAN.

Complessità: Nel caso peggiore $O(n^2)$, dove n è il numero di oggetti che devono essere raggruppati.

La combinazione OPTICS e DBSCAN ci permette di trovare i parametri "corretti" e trovare i clusters.

ordinamento crescente



DENCLUE (Density-based CLUSTering)

Il problema di DBSCAN o OPTICS è che la densità è calcolata contando il numero di oggetti in un "quartiere" definito da ϵ . Questa densità può essere molto sensibile al valore del raggio utilizzato.

DENCLUE è basato su un insieme di density distribution function

Ogni oggetto osservato è trattato come un indicatore di high-probably density nella regione circostante. La probability density in un punto dipende dalla distanza dal punto e l'oggetto osservato.

IDEA: Per ogni oggetto (punto), piazzare una density distribution function in quel punto (solitamente una Gaussian function).

Faccendo la somma delle funzioni che si sovrappongono, ottengo dei valori (nel caso delle Gaussian ottengo dei "picchi") più altri che mi vanno a descrivere la densità in quella zona. Per determinare i cluster dobbiamo decidere:

- Come "tagliare" queste montagne trovate (punto di soglia).
- Quanto possono essere grandi le montagne (alla base)

Siano x_1, \dots, x_n esempi indipendenti e ugualmente distribuiti di una variabile random f. La kernel density approximation della probability density function equivale a:

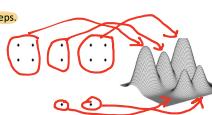
$$\hat{f}_h(x) = \frac{1}{nh} \sum_{i=1}^n K\left(\frac{x-x_i}{h}\right) \rightarrow \text{somma delle funzioni (density in the overall space)}$$

Il Kernel $K()$ è una funzione integrabile, con valori reali, non negativi che dovrebbe soddisfare due requisiti per ogni valore di h :

$$\int_{-\infty}^{+\infty} K(u) du = 1 \text{ and } K(-u) = K(u)$$

Esempio di Kernel function (Gaussian function): $K_{\text{Gauss}}\left(\frac{x-x_i}{h}\right) = \frac{1}{\sqrt{2\pi}} e^{-\frac{(x-x_i)^2}{2h^2}}$

$K_{\text{Square}}\left(\frac{|x-x_i|}{h}\right) = \begin{cases} 0 & \text{if } \frac{|x-x_i|}{h} > 1 \\ 1 & \text{otherwise} \end{cases}$



Algoritmo di assegnazione dei density attractors per ogni punto:

Fin quando esiste un samples nei dataset:

- 1) Seleziona il sample x. Il density attractor per x è calcolato usando la procedura di hill climbing. (Metàforicamente cerchiamo di scalare la montagna detta del density attractor).

$$x^{j+1} = x^j + \frac{\nabla \hat{f}(x^j)}{|\nabla \hat{f}(x^j)|}$$

Dove: Δ indica la velocità di convergenza dell'algoritmo. Se Δ è troppo grande rischio di "saltare" il picco della density function!

- 2) La procedura di hill climbing termina allo step $K > 0$: $\hat{f}(x^{j+1}) < \hat{f}(x^j)$

Cioè se troviamo un valore al passo $K+1$ maggiore del valore al passo K \rightarrow Vedi dove altrove saliamo il picco

- 3) Assegna x al density attractor x^*

Per migliorare l'efficienza dell'algoritmo, si va a considerare tra i punti nel dataset essendo che probabilmente punti vicini verranno attratti dallo stesso density attractor, usando questa euristiche, possiamo ridurre il numero di volte che la procedura di hill climbing viene fatta, perché andiamo a determinare il density attractor per quei punti senza dover computare l'algoritmo :

L'algoritmo salva tutti i punti x^* con $d(x_j, x^*) \leq \Delta$ per ogni step $0 < j < k$ durante la procedura di hill-climbing e assegna questi punti al cluster di x^* .

ATTENZIONE: Un oggetto x è un outlier/noise se l'algoritmo di hill-climbing su quell'oggetto converge in un local maximum x^* .

$$\hat{f}(x^*) \geq \xi$$

di shape arbitraria!
i cluster vengono creati andando ad unire i density attractors che sono connessi, tra loro, da pattern con alta densità (maggiore soglia).

Un cluster di forma arbitraria per un insieme di density attractors x , è un subset $C \subseteq D$, dove:

$$\forall x \in C \exists x' \in X : \hat{f}(x') \geq \xi$$

Cioè, per ogni density attractor x_1^*, x_2^* esiste un path ρ che connette x_1^*, x_2^* dove $\forall p \in \rho: f(p) \geq \xi$

PARAMETRI DA CONSIDERARE:

- H = Grandezza dei picchi - Impatta il numero di punti che vengono inclusi durante l'hill climbing per ogni attractor



ξ - Fissa il cut point per i nostri picchi, indica il numero di punti che vengono inclusi nel cluster. Se è troppo basso, rischiamo che tutti i punti vengano inclusi nello stesso cluster. Viceversa, aumentando ξ otteniamo un maggior numero di clusters.



Maggiori caratteristiche di DENCLUE

- Solide fondamenta matematiche
- Buono per dataset con grandi quantità di rumore
- Permette una descrizione matematica compatta di un cluster con forma arbitraria in un dataset multidimensionale
- Significativamente più veloce degli algoritmi esistenti
- Complessità (se ottimizzato): $O(N \log(N))$

Necessità però di un'accurata scelta dei parametri h e ξ :

- h determina l'influenza di un punto nei suoi vicini
- ξ Descrive se un density-attractor è significante, permettendo la riduzione del numero di density-attractors e aiutando nel migliorare come i parametri dovrebbero essere scelti per ottenere un buon risultato

Ottimizzazione dell'algoritmo

PASSO 1

- 1) Creiamo una "griglia" all'interno del dataset per dividerlo in d-dimensional hypercubes. ($sd \gg \approx 2h$)
 - 2) Solo gli hypercubes che contengono dei data points sono determinati.
 - 3) Gli hypercubes vengono numerati dipendentemente dalla loro posizione relativa rispetto ad un'origine fissa. Le chiavi dei cubi possono essere salvate efficientemente in un albero di ricerca o in un B+ tree.
 - 4) Per ogni cubo c_i , in aggiunta alla chiave, vengono salvati il numero di punti che appartengono a c_i , i puntatori a questi punti e la loro somma lineare. Queste informazioni sono usate durante il passo di clustering per una veloce computazione della media di un cubo. Essendo che i cluster possono essere sparsi su più di un cubo, i cubi vicini devono essere accoppiati.
 - 5) Per velocizzare il processo, connettiamo tra loro i cubi vicini.
- Formalmente connettiamo due cubi se $d(\text{mean}(c_i), \text{mean}(c_j)) < 4\sigma$

PASSO 2

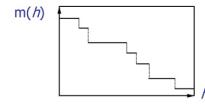
- 1) Solo i cubi che sono molto popolati (C_{ip}) e i cubi che sono collegati ai cubi molto popolati vengono considerati per determinare i clusters. Questo ci permette di velocizzare l'esecuzione concentrandosi solo su alcuni cubi.
- 2) Per ogni punto nei cubi popolati, calcoliamo la local density-function come segue:

$$\hat{f}_{Gauss}^D(x) = \sum_{x_1 \in \text{near}(x)} e^{-\frac{d(x, x_1)^2}{2\sigma^2}}$$

3) Applichiamo l'hill climbing procedure

Con questa implementazione possiamo velocizzare la computazione della density function e della local density function.

- Suggested choice for h : consider different h and determine the largest interval between h_{\max} and h_{\min} where the number of density attractors $m(h)$ remains constant



- Suggested choice for ξ : if the database is noise free, all density attractors of D are significant and ξ should be chosen in

$$0 \leq \xi \leq \min_{x \in X} \{f^{(n)}(x)\}$$

Considerazioni finali dei Density-Based clustering algorithms

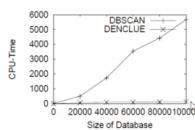
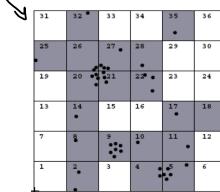
Tutti i density-based clustering methods non sono molto effettivi quando facciamo clustering di high dimensional data perché incarna nel problema la curse of dimensionality, cioè all'aumentare della dimensionialità, si allontano i punti nello spazio.

Quindi quando abbiamo spazi con grande dimensionale è INAPPROPRIATO utilizzare algoritmi di clustering basati sulla densità.

L'importante: Utilizzare altri algoritmi più adatti.

IMPORTANTE: Sono molto interessanti perché permettono di trovare clustering di diverse forme, però è molto complicato dovuto al settaggio dei parametri. Anche un piccolo cambiamento nei parametri può portare a grosse differenze.

Abbiamo però il vantaggio di non dover determinare il numero di cluster che vogliamo ottenere, anche se andando a settare i parametri degli algoritmi, indirettamente andiamo anche a settare il numero di cluster che andremo ad ottenere.



Grid-Based Clustering Method

Usano una multi-resolution grid data structure.

Tipicamente questi tipi di algoritmi non sono molto costosi a livello computazionale.

I risultati che possiamo ottenere sono molto inferiori rispetto ai risultati che possiamo ottenere con gli altri algoritmi.

Si utilizzano solitamente quando dobbiamo gestire grandi quantità di dati e velocizzare la computazione per arrivare al risultato.

STING(STatistical INformation Grid approach) clustering method

STING è un algoritmo grid-based e gerarchico.

Lo spazio viene diviso in celle rettangolari. Ci sono diversi livelli di celle corrispondenti a differenti livelli di risoluzione.

Ogni cella ad un alto livello è partitionata in un maggiore numero di celle più piccole in un livello successivo più basso.

Per ogni cella vengono calcolate, in anticipo, delle informazioni statistiche e vengono salvate (verranno utilizzate per rispondere alle queries).

I parametri per le celle più basse (i-th layer) sono calcolati direttamente dai valori presenti nelle tabelle, quando i dati vengono caricati nel dataset.

I parametri delle celle a livello più alto saranno facilmente calcolabili dai parametri delle celle più basse.

I parametri includono:

- Count, mean, standard deviation, min, max
- Tipo di distribuzione: normale, uniforme, esponenziale o nulla

Venne usato un approccio bottom-up per costruire la piramide.

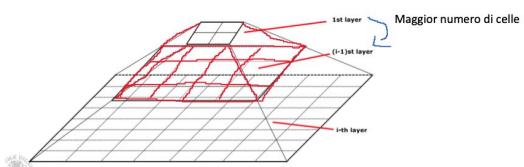
Venne usato un approccio top-down per rispondere alle queries (SQL) sui dati.

Le regioni restituite da STING sono un'approssimazione dei risultati di DBSCAN. Quando la granularità si avvicina a zero, le regioni restituite da STING si avvicinano al risultato di DBSCAN.

Query Processing

Usiamo un approccio top-down.

- 1) Iniziamo da un pre-selected layer (solitamente quello con il minor numero di celle) - Non deve essere obbligatoriamente il livello più alto della piramide.
- 2) Per ogni cella nel current layer computiamo il confidence interval (riflette la pertinenza della cella in base alla query). Il confidence interval è calcolato usando i parametri statistici di ogni cella.
- 3) Dal confidence interval calcolato andiamo a dare un label alle celle (rilevante o non rilevante per la query)
- 4) Rimuoviamo le celle irrilevanti, vogliamo seguire solo le celle rilevanti per la query.
- 5) Dato il layer corrente, procediamo con il livello inferiore andando a esaminare solo le celle che derivano da celle "rilevanti"
- 6) Ripetere questo processo fin quando il bottom-layer non viene raggiunto.
- 7) Se le condizioni della query sono rispettate, le regioni che sono rilevanti per la query, raggiunto il bottom-layer, vengono restituite.
- Altrimenti, i dati che sono contenuti nelle celle rilevanti vengono restituiti e processati fin quando non rispettano le condizioni della query.

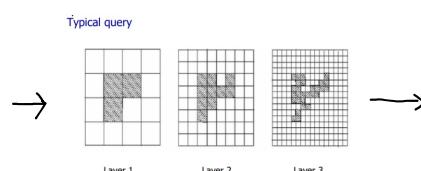


Vantaggi

- Indipendente dalle query, facile da parallelizzare, update incrementale
- La generazione dei cluster ha costo $O(n)$
- Il processo di Query Processing costa $O(g)$, dove g è il numero di celle all'ultimo livello

Svantaggi

- Tutti i boundaries dei cluster sono orizzontali o verticali, non vengono trovati boundary diagonali.



Mettendo insieme le celle che otteniamo ai vari livelli, possiamo considerare questo algoritmo un algoritmo di clustering.

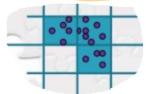
CLIQUE (Clustering In QUEst) clustering method

CLIQUE identifica automaticamente i subspaces di un high dimensional data space che permettono un clustering migliore rispetto allo spazio originale. CLIQUE può essere considerato sia un density-based che grid-based algorithm perché lavora sull'idea di densità e divide lo spazio in celle.

- Partiziona ogni dimensione nello stesso numero di intervalli di uguale lunghezza (rettangoli non sovrapposti)
- Un'unità è densa se la frazione dei punti totali contenuti in una cella supera il **parametro in input del modello**.



- Un cluster è insieme massimo di celle dense connesse all'interno di un sottospazio.



Se una cella k dim. è densa, anche le sue **vicine** in uno spazio $k-1$ dim lo sono!

! IMPORTANTE

Passi principali:

- Partizionare il data space e trovare il numero di punti che appartengono a ogni cella
- Identificare i subspaces che contengono clusters usando l'**APRIORI PRINCIPE**: Se una cella k -dimensionale è densa, allora lo sono anche le sue proiezioni di $(k-1)$ dimensione. Perciò, le celle dense nello spazio k -dimensionale sono generate dalle celle dense nello spazio $(k-1)$ -dimensionale.
- Identificare i cluster:
 - Determinare le celle dense in tutti i subspaces di interesse
 - Determinare le celle dense connesse in tutti i subspaces di interesse
- Generare una minima descrizione per i clusters:
 - Determinare la regione massima che copre un cluster di celle dense connesse per ogni cluster
 - Determinare la regione minima coperta da ogni cluster

Vantaggi:

- Trova automaticamente subspaces con la dimensionalità maggiore t.c cluster con alta densità esistono in questi subspaces
- Non è sensibile all'ordine dei records in input e non assume nessuna data distribution
- Scala linearmente con la dimensione dell'input e ha una buona scalabilità quando il numero di dimensionalità dei dati aumenta.

Svantaggi:

- Ottenerne un cluster che ha significato dipende da un giusto perfezionamento dei parametri (grid size e density threshold). Se usiamo una piccola grid size otteniamo una buona precisione nel definire il cluster ma ovviamente è più costoso computazionalmente.
- L'accuratezza del clustering potrebbe essere degradata a discapito della semplicità dell'algoritmo.

Algoritmo

PRIMO PASSO - Partizionare il data space d -dimensionale in celle rettangolari che non sono sovrapposte, identificando quali sono celle dense.

- CLIQUE partiziona ogni dimensione in intervalli e identifica gli intervalli che contengono almeno l punti, dove l è un density threshold. Queste sono le celle candidate (density cells).
- CLIQUE iterativamente unisce due k -dimensional celle dense (c_1 e c_2) se:
 - $D_{i_1} = D_{j_1}, \dots, D_{i_{k-1}} = D_{j_{k-1}}$ **J** hanno le stesse $k-1$ dimensioni
 - c_1 e c_2 condividono lo stesso intervallo in queste dimensioni. L'operazione di join genera una nuova cella $(k+1)$ -dimensional space. La cella generata viene aggiunta alle celle candidate.
- CLIQUE verifica se il numero di punti nella cella generata supera il density threshold. L'iterazione termina quando nessun candidato può essere generato o non ci sono celle candidate che sono dense.

SECONDO PASSO - CLIQUE usa le celle dense in ogni subspace per rappresentare i cluster, i quali possono essere di forma arbitraria ma vengono rappresentati da dei boundary rettangolari.

Attenzione:

- Minimum Description Length (MDL) Principle:** Tutte le celle che vengono unite in un cluster devono essere dense e il cluster non può essere esteso in qualsiasi dimensione del subspace. Il cluster è una maximal region (hyperrectangle) che copre tutte le connected dense cell.
- Trovare la miglior descrizione di un cluster è un problema NP-HARD. Per questo motivo CLIQUE adotta un greedy approach. Inizia con un numero arbitrario di celle, trova la maximal region che copre le celle e successivamente lavora sulle celle rimanenti che non sono state ancora coperte. Il metodo greedy termina quando tutte le dense cells sono state coperte.

Quindi la main idea è:

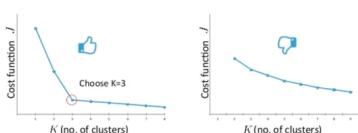
Si usano dense cells in un subspace di dimensionalità minore per trovare le dense cells in subspace con dimensionalità maggiore.

Determinare il numero di clusters

1) Elbow Method

IDEA: Se vado a suddividere un "natural cluster" in due cluster non ottengo un grosso miglioramento nella cost function perché la distribuzione dei punti nel cluster singolo è la stessa che nei due cluster separati. Uso il **turning point** nella curva della somma cioè cerco quel punto nel grafico della cost function all'aumentare di K che mi determina un cambio brusco.

Questo metodo funziona se naturalmente ho dei cluster che sono ben separati, nel caso in cui non ho cluster che sono ben separati e identificabili questo metodo non funziona perché non posso determinare un turning point.



Turning point -> Cluster ben separati

Non posso identificare il turning point -> Cluster non identificabili / ben separati

2) Cross-Validation Method

- Dividiamo il dataset in m parti
- Usiamo $m-1$ parti per ottenere un clustering model
- Usiamo la restante parte per testare la qualità del cluster

Per ogni $K > 0$, ripetere il procedimento e comparare la quality measure tra i differenti K e trovare il numero di cluster (K) che fitta al meglio i dati

Valutazione di un clustering

Una volta effettuato il clustering, bisogna valutarlo. Per valutarlo si misura quanto bene i cluster fittano il dataset e grazie a questa misura si confronta con altri tipi di clustering sempre sul solito dataset.

Per misurare la qualità di un clustering ci sono due metodi:

- Extrinsic (supervised):** Abbiamo disponibile il ground truth. Di solito siamo in questa casistica quando dobbiamo comparare la clustering quality measure di alcuni clustering algorithmi, cioè verificare se il clustering che applichiamo è di buona qualità rispetto a quello che già abbiamo come ground truth.
- Intrinsic (unsupervised):** Non abbiamo disponibile il ground truth. Valuta quanto è buono un clustering considerando quando bene sono separati i clusters e quanto essi sono compatti.

Extrinsic Methods

La Clustering Quality Measure per un clustering C e un ground truth C_g è rappresentata come: $Q(C, C_g)$

Una Clustering Quality Measure è buona se soddisfa questi 4 criteri essenziali:

- Cluster homogeneity: Più un cluster è puro (contiene solo una classe), meglio è
- Cluster completeness: Dovrebbe assegnare oggetti che appartengono alla stessa categoria, nel ground truth, allo stesso cluster.
- Rag bag: Inserire un oggetto eterogeneo all'interno di un cluster puro dovrebbe penalizzare di più che mettere un oggetto in un cluster composto da più categorie (Rag Bag)
- Small cluster preservation: Splittere una piccola categoria in più pezzi è più dannoso che splittare una grande categoria in più pezzi (dovremmo preservare i cluster piccoli).

Bcubed

E' uno tra gli extrinsic methods più famosi e tiene in considerazione la precisione e il recall.

Calcola la precisione e il recall per ogni oggetto in un clustering dato un certo dataset, in accordo con il ground truth.

- Precision di un oggetto: Indica quanti altri oggetti nello stesso cluster appartengono alla stessa categoria dell'oggetto.
- Recall di un oggetto: Riflette quanti oggetti della stessa categoria dell'oggetto vengono assegnati allo stesso cluster.

Formalmente

Sia Ω un insieme di oggetti $o_1 \dots o_n$

Sia C un clustering di Ω

Sia $L(o_i)$ la categoria assegnata all'oggetto o_i secondo il ground truth

Sia $C(o_i)$ l'ID del cluster a cui appartiene l'oggetto o_i

Intrinsic Methods

Non abbiamo disponibile il ground truth. Valuta quanto è buono un clustering considerando quando bene sono separati i clusters e quanto essi sono compatti.

Silhouette coefficient

Metrica utilizzata per valutare la qualità di un modello di clustering.

- Si calcola il silhouette coefficient per ogni oggetto o all'interno del dataset.
- Si calcola la media dei valori ottenuti

Calcolo del Silhouette coefficient

- Calcolare $a(o)$, cioè la distanza media tra o e tutti gli altri oggetti del cluster in cui o è contenuto. (Misura la compattatezza del cluster in cui o è contenuto, vogliamo che il valore di $a(o)$ sia basso perché vogliamo che il cluster sia compatto).

$$a(o) = \frac{\sum_{o' \in C_o, o \neq o'} \text{dist}(o, o')}{|C_o| - 1}$$

- Calcolare $b(o)$, cioè la minima distanza media tra o e tutti i clusters in cui o non è contenuto. (Misura la separatezza tra i cluster, vogliamo che il valore di $b(o)$ sia alto).

Dati due oggetti o_i, o_j , la correttezza della relazione tra o_i, o_j nel clustering C è data nel seguente modo:

$$\text{Correctness}(o_i, o_j) = \begin{cases} 1 & \text{if } L(o_i) = L(o_j) \Leftrightarrow C(o_i) = C(o_j) \\ 0 & \text{otherwise.} \end{cases}$$

Vale 1 se entrambi gli oggetti sono assegnati allo stesso cluster, e hanno la stessa classe nel ground truth.
Vale 0 altrimenti.

Da questa definizione ci ricaviamo la **precision** e il **recall**:

$$\text{Precision BCubed} = \frac{\sum_{i=1}^n \sum_{j \neq i, C(o_i)=C(o_j)} \text{Correctness}(o_i, o_j)}{\sum_{i=1}^n \| \{o_j | i \neq j, C(o_i)=C(o_j) \| \}}$$

$$\text{Recall BCubed} = \frac{\sum_{i=1}^n \sum_{j \neq i, L(o_i)=L(o_j)} \text{Correctness}(o_i, o_j)}{n}$$

Confrontando due modelli tra loro: il **modello con Precision e Recall più alti** risulta il modello migliore su quel dataset.

$$b(o) = \min_{C_j: 1 \leq j \leq k, j \neq i} \left\{ \frac{\sum_{o' \in C_j} \text{dist}(o, o')}{|C_j|} \right\}$$

3) Calcolare il silhouette coefficient $s(o)$:

$$s(o) = \frac{b(o) - a(o)}{\max\{a(o), b(o)\}}$$

- Quando $s(o)$ è vicino a 1, il cluster che contiene o è compatto e o è lontano dagli altri clusters. (**CASO MIGLIORE**)
- Quando $s(o)$ è vicino a -1, otteniamo che o è più vicino agli oggetti di un altro cluster rispetto agli oggetti dello stesso cluster -> cluster non compatto. (**CASO PEGGIORE**).

Per misurare la qualità di un clustering, possiamo usare la media, calcolata su tutti gli oggetti all'interno del dataset, del silhouette coefficient value.

ATTENZIONE:

Il silhouette coefficient venne definito per algoritmi di clustering del tipo Partition Clustering. Esso lavora molto bene se la forma dei cluster è convessa (cluster sferici o ovali). Se lavoriamo con cluster concavi, il silhouette coefficient non funziona perché $a(o)$ è grande e $b(o)$ potrebbe risultato molto piccolo. Per clustering concavi è difficile determinare la qualità del clustering perché è difficile determinare la compattezza e separatezza dei clusters.



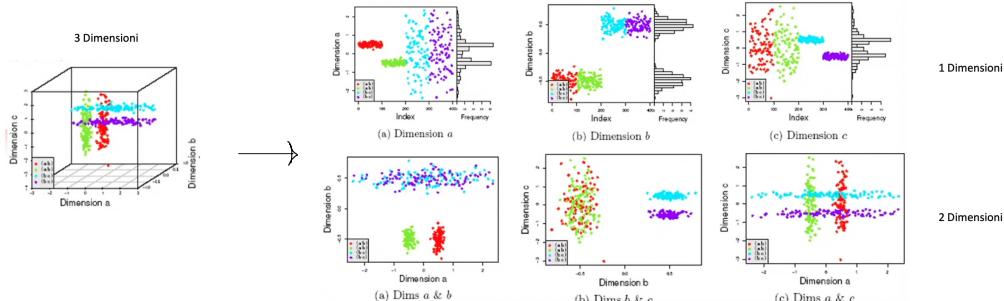
High-Dimensional Clustering

Per andare a fare l'operazione di clustering su **high-dimensional dataset** ci sono principalmente due metodi:

- Subspace-clustering:** Cercare i cluster in subspaces dello spazio originale (CLIQUE, Proclus e bi-clustering approaches)
- Dimensionality reduction approaches:** Costruire uno spazio a dimensioni minori e cercare i cluster in questo spazio (Dimensionality reduction methods e spectral clustering)

Subspace-Clustering Methods

I cluster potrebbero esistere solo in alcuni sottospazi. Con il **subspace-clustering** vogliamo trovare i cluster in tutti i sottospazi.



Subspace Search Methods

Cerchiamo nei vari subspazi per trovare i clusters. La similarità è basata sulla distanza o la densità.

- Approccio bottom-up**
 - Iniziamo da un subspace a basse dimensioni e cerchiamo in un subspace con più dimensioni solo quando ci potrebbero essere dei cluster in quel sottospazio.
 - Si usano varie tecniche di pruning per ridurre il numero di subspace con più dimensioni in cui cercare.

Esempio di algoritmo: CLIQUE
- Approccio top-down**
 - Iniziamo da un "full space" e cerchiamo in subspace più piccoli **ricorsivamente**.
 - Questo approccio è effettivo solo quando la locality assumption holds: il subspace di un cluster può essere determinato dal vicinato (scoprire local similarity in alte dimensioni).

Esempio algoritmo: PROCLUS (k-medoid like method)

Correlation-Based Methods

La similarità è basata su modelli avanzati di correlazione.

Esempio di algoritmo: PCA, Hough transform, Fractal dimension

Lavorando con il PCA possiamo:

- Ridurre la dimensionalità dello spazio
- Trovare i cluster nello spazio ridotto
- Ritornare allo spazio originale e riportare i cluster su esso (possiamo farlo perché possiamo ricondursi allo spazio originale da quello ridotto e di conseguenza sapere quali punti dello spazio ridotto corrispondono a quali punti nello spazio "completo")

Bi-Clustering Methods

L'idea è di fare il cluster degli oggetti e degli attributi simultaneamente, durante l'esecuzione dell'algoritmo.

- Requirements:**
- Solo un piccolo insieme di oggetti partecipa in un cluster
 - Un cluster comprende solo un piccolo numero di attributi
 - Ogni oggetto può partecipare in più clusters, o non partecipare i nessun cluster (questo perché un oggetto potrebbe partecipare a cluster diversi in subspazi diversi).
 - Un attributo potrebbe partecipare in più cluster, o non partecipare i nessun cluster.

Tipi di bi-clusters

Siano:

- $A = \{a_1, \dots, a_n\}$ un insieme di oggetti
- $B = \{b_1, \dots, b_n\}$ un insieme di attributi
- $E = [e_{ij}]$ il valore dell'attributo j per l'oggetto i

Un **bi-cluster** è definito come una submatrix dove oggetti e attributi seguono alcuni pattern consistenti.

	...	b_6	...	b_{12}	...	b_{36}
a_1	...	60	...	60	...	60
...
a_{33}	...	60	...	60	...	60
...
a_{86}	...	60	...	60	...	60

$$E = [e_{86,36}] = 60$$

Possible applications:

- Gene expression
- Clustering customers and products

1) Bi-clusters con valore costante

Abbiamo un subset di oggetti e attributi tali che ogni valore E è costante.

Vogliamo trovare nella matrice generale una sottomatrice dove i suoi valori hanno valore costante.

for any j in I and $j \in J$, $e_{ij} = c$

	...	b_6	...	b_{12}	...	b_{36}
a_1	...	60	...	60	...	60
...
a_{33}	...	60	...	60	...	60
...
a_{86}	...	60	...	60	...	60

Submatrice composta da

$$\{a_1, a_{33}, a_{86}\} \times \{b_6, b_{12}, b_{36}\}$$

3) Bi-clusters con valori coerenti (basati su un pattern)

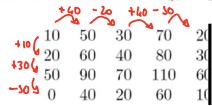
Abbiamo un subset di oggetti e attributi tali che ogni valore E sulle righe cambia in un modo sincronizzato in rispetto delle colonne e viceversa.

Vogliamo trovare nella matrice generale una sottomatrice dove i suoi valori hanno un pattern basato sulle righe.

NB: Naturalmente invece di avere i valori con pattern per riga, potremmo averli per colonna.

$e_{ij} = c + a_i + \beta_j$
A $I \times J$ is a bicluster with coherent values if and only if for any

$i_1, i_2 \in I$ and $j_1, j_2 \in J$, then $e_{i_1 j_1} - e_{i_2 j_1} = e_{i_1 j_2} - e_{i_2 j_2}$



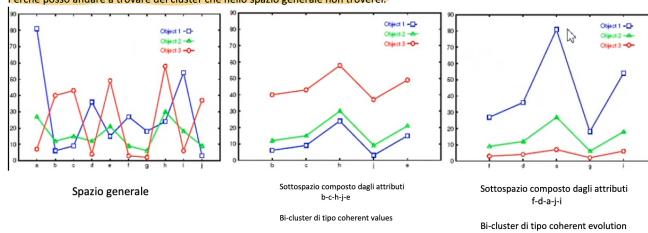
Bi-Clusters Methods

I metodi Bi-Clusters si dividono in: Optimization-based Methods e Enumeration Methods.

Questi metodi vanno a trovare un'approssimazione dei tipi di bi-cluster presentati sopra (questo perché i dati nel "mondo reale" sono rumorosi), solitamente e quindi non avremo esattamente i bi-cluster presentati sopra.

Perché è interessante applicare i metodi di bi-cluster?

Perché posso andare a trovare dei cluster che nello spazio generale non trovavo.



Optimization-based Methods

E' una ricerca iterativa, nella quale ad ogni iterazione la submatrix con il più alto significance score è definito come bicluster. La ricerca è greedy per trovare il local optimal bi-cluster (dovuto al costo della computazione).

ALGORITMO: δ – Cluster Algorithm

δ – Cluster Algorithm

Definizioni:

Data una submatrix $I \times J$:

$$e_{iJ} = \frac{1}{|J|} \sum_{j \in J} e_{ij} \quad \text{Media della } i\text{-th riga}$$

$$e_{IJ} = \frac{1}{|I||J|} \sum_{i \in I, j \in J} e_{ij} \quad \text{Media della } j\text{-th colonna}$$

$$e_{IJ} = \frac{1}{|I||J|} \sum_{i \in I, j \in J} e_{ij} = \frac{1}{|I|} \sum_{i \in I} e_{iJ} = \frac{1}{|J|} \sum_{j \in J} e_{ij} \quad \text{Media di tutti gli elementi nella submatrix}$$

$$H(I \times J) = \frac{1}{|I||J|} \sum_{i \in I, j \in J} (e_{ij} - e_{iJ} - e_{Ji} + e_{IJ})^2 \quad \text{Qualità della submatrix come bi-cluster}$$

tranne il mean squared residue value

1) **δ – bi – cluster:** Una submatrix $I \times J$ è una δ – bi – cluster se $H(I \times J) \leq \delta$, dove $\delta \geq 0$ è una soglia.

OSS: $\delta = 0$, $I \times J$ è un bi-cluster perfetto con valori coerenti.

c) Quando $\delta = 0$, $I \times J$ è un bi-cluster perfetto con valori coerenti.

c) Settando $\delta > 0$, l'utente può specificare la tolleranza sul rumore medio per ogni elemento rispetto al bi-cluster perfetto.

2) **Maximal δ – bi – cluster:** Un maximal δ – bi – cluster è un δ – bi – cluster $I \times J$ t.c. non esiste un altro δ – bi – cluster $I' \times J'$ che contiene $I \times J$.

La computazione per trovare il maximal δ – bi – cluster è costosa quindi utilizziamo una ricerca heuristica greedy per ottenere l'ottimo locale.

La computazione si divide in due parti:

- 1) **Deletion Phase:** Si inizia dall'intera matrice, iterativamente si rimuovono le righe e le colonne per le quali il mean squared residue della matrice è maggiore di δ.
- a. Ad ogni iterazione, per ogni riga/colonna, calcoliamo il mean squared residue

$$d(i) = \frac{1}{|J|} \sum_{j \in J} (e_{ij} - e_{iJ} - e_{Ji} + e_{IJ})^2 \quad d(j) = \frac{1}{|I|} \sum_{i \in I} (e_{ij} - e_{iJ} - e_{Ji} + e_{IJ})^2$$

b. Rimuoviamo le righe o colonne con il più grande mean squared residue.

OSS: Al termine di questa fase otteniamo una sottomatrice $I \times J$ che è un δ – bi – cluster. La sottomatrice però non è maximal.

2) **Addition Phase:**

- a. Si estende iterativamente il δ – bi – cluster $I \times J$ ottenuto nella deletion phase fin quando i requisiti del δ – bi – cluster sono soddisfatti.
- b. Considerare tutte le righe/colonne non incluse nel bi-cluster $I \times J$ corrente calcolando i loro mean squared residues.

c. Una riga/colonna con il più piccolo mean squared residue viene aggiunta nel δ – bi – cluster corrente.

ATTENZIONE: Questo procedimento trova solo UN δ – bi – cluster, quindi richiede di essere eseguito molte volte, per trovare altri bicluster, andando a rimpiazzare gli elementi del bi-cluster trovato in output con numeri random.

2) Bi-clusters con valore costante sulle righe

Abbiamo un subset di oggetti e attributi tali che ogni valore E sulle righe è costante.

Vogliamo trovare nella matrice generale una sottomatrice dove i suoi valori hanno valore costante basato sulle righe.

NB: Naturalmente invece di avere i valori costanti per riga, potremmo averli per colonna.

$$e_{ij} = c + a_i$$

where a_i is the adjustment for row i .

10	10	10	10	10
20	20	20	20	20
50	50	50	50	50
0	0	0	0	0

4) Bi-cluster con evoluzione coerente sulle righe

Si considera la tendenza delle righe/colonne, cioè se i valori tendono a salire per una riga, allora salgono per tutte le righe. Uguale per le colonne.

$i_1, i_2 \in I$ and $j_1, j_2 \in J$, then $(e_{i_1 j_1} - e_{i_2 j_1})(e_{i_2 j_1} - e_{i_2 j_2}) \geq 0$

10	50	30	70	20
20	100	50	1000	30
50	100	90	120	80
0	80	20	100	10

Enumeration Methods

Usa una soglia di tolleranza per specificare il grado di rumore permesso nel bi-cluster per essere minato.

Successivamente cerca di enumerare tutte le sottomatrici (bi-clusters).

ALGORITMO: δ – pCluster Algorithm

δ – pCluster Algorithm

Dato che una submatrix $I \times J$ è un bi-cluster con coherent values se e solo se

$$e_{1,1} - e_{2,1} = e_{1,2} - e_{2,2}$$

Per ogni 2×2 submatrix di $I \times J$, definiamo il p-score come:

$$p\text{-score} \left(\begin{array}{cc} e_{1,1} & e_{1,2} \\ e_{2,1} & e_{2,2} \end{array} \right) = |(e_{1,1} - e_{1,2}) - (e_{2,1} - e_{2,2})|$$

Una submatrix $I \times J$ è un δ – pcluster se il p-score di ogni 2×2 submatrix di $I \times J$ vale al più δ , dove $\delta \geq 0$ è una soglia specificata dall'utente come tolleranza al rumore rispetto ad un perfect bi-cluster.

Il p-score controlla il rumore di ogni elemento nel bi-cluster
Il mean squared residue cattura la media del rumore

Monotonicità: Se $I \times J$ è un δ – pCluster, ogni $x \times y$ ($x, y \geq 2$) submatrix di $I \times J$ è un δ – pCluster.

Un δ – pCluster è massimo se nessun'altra riga o colonna può essere aggiunta al cluster e conserva δ – pCluster.

Obiettivo: Dobbiamo solo trovare tutti i massimi δ – pCluster.

MaPle algorithm - Efficient enumeration of δ – pClusters
Per ogni attributo J , trovare il massimo subset di oggetti I c $I \times J$ è un δ – pCluster, se $I \times J$ non è una submatrix di un altro δ – pCluster allora $I \times J$ è un maximal δ – pCluster.

Spectral Clustering

Combina feature extraction e clustering.

Usa lo spettro della similarity matrix per eseguire una dimensionality reduction per fare il clustering in dimensionalità ridotta.

Algoritmo (Ng-Jordan-Weiss Algorithm)

L'algoritmo consiste nel, dato un insieme di oggetti o_1, \dots, o_n e la distanza tra ogni paio di oggetti $dist(o_i, o_j)$, trovare K clusters:

- Calcolare l'affinity matrix W , dove: σ = scaling parameter che controlla quanto velocemente la affinity W_{ij} decresce all'aumentare della distanza $dist(o_i, o_j)$.

$$W_{ij} = e^{-\frac{dist(o_i, o_j)}{\sigma^2}},$$

Attenzione: $W_{ii} = 0$

- Deriviamo una matrice $A = f(W)$ in questo modo:

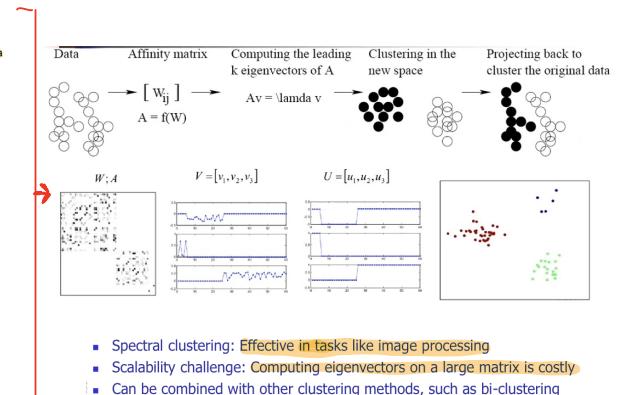
a. Definiamo una matrice D come una matrice diagonale t.c. D_{ii} è la somma dell' i -esima riga di W

$$D_{ii} = \sum_{j=1}^n W_{ij}$$

b. Calcoliamo A come:

$$A = D^{-\frac{1}{2}} W D^{-\frac{1}{2}}$$

- Troviamo i k primi autovettori di A (cioè i k autovettori che hanno gli autovalori più grandi).
- Usando i k autovettori, proiettiamo i dati originali in un nuovo spazio definito dai k autovettori.
- Eseguiamo un algoritmo di clustering sul nuovo spazio (ad esempio un k-means) per trovare i k clusters.
- Assegniamo i punti del dataset originale ai cluster in base a come i punti trasformati sono assegnati nei cluster ottenuti.



- Spectral clustering: Effective in tasks like image processing
- Scalability challenge: Computing eigenvectors on a large matrix is costly
- Can be combined with other clustering methods, such as bi-clustering

Constrained Cluster Analysis

Constraint-based Cluster Analysis

Questa tipologia di clustering richiedono il feedback dell'utente. Questo ha senso perché gli utenti conoscono le loro applicazioni al meglio. Si considerano dei vincoli per il clustering che stiamo andando ad applicare.

Tipologie di vincoli

- Vincoli sulle istanze:** Specificano come un paio o un insieme di istanze dovrebbero essere raggruppate nella cluster analysis
 - must-link* (x, y : x e y devono essere raggruppati in un cluster - esempio: *must-link*(x, y) if $dist(x, y) < 5$)
 - cannot-link* (x, y : x e y non devono essere raggruppati in un cluster - esempio: *cannot-link*(x, y) if $dist(x, y) > 5$)
- Vincoli sui cluster:** Specificano un requisito sul cluster (numero minimo di oggetti per cluster, massimo diametro del cluster, forma del cluster, numero di clusters)
 - δ -constraint (minimum separation)**
 - For any two clusters $S_i, S_j \forall i, j$
 - For any two instances $s_p \in S_i, s_q \in S_j, \forall p, q$

} Indica la distanza minima tra due punti di cluster diversi
 - ϵ -constraint (compattezza del cluster)**
 - For any cluster $S_i, |S_i| > 1$
 - $\forall p, s_p \in S_i, \exists s_q \in S_i: \epsilon \geq D(s_p, s_q), s_p <> s_q$

} Indica la distanza max tra due punti dello stesso cluster
- Vincoli su misure di similarità:** Specificano un requisito che il calcolo della similarità deve rispettare (ad esempio per una misura di distanza, il vincolo di non poter passare attraverso i muri)

Vincoli forti(hard) e deboli(soft)

- Vincolo forte:** Un vincolo forte se un clustering che viola il vincolo non può essere accettato
- Vincolo debole:** Un vincolo è debole se un clustering che viola il vincolo non è preferibile ma è accettabile quando nessuna soluzione migliore può essere trovata. I vincoli soft (debolli) possono essere chiamati anche preferences.

Clustering with constraints

Fare clustering con vincoli consiste nel partizionare i dati senza label in cluster e usare i vincoli per aiutare e inserire bias nel clustering. L'obiettivo è di raggruppare gli esempi simili nello stesso cluster, dividerli da quelli diversi e rispettare i vincoli.

Far rispettare i vincoli

- Strict enforcement:** Trovare il miglior clustering che rispetta tutti i vincoli
- Partial enforcement:** Trovare il miglior clustering possibile che rispetta il massimo numero di vincoli

Come misurare la qualità e l'utilità di un insieme di vincoli?

- Informativeness:** La quantità d'informazioni portata dai vincoli che sono al di là del clustering model.
- Dato un dataset D , un cluster method A_c e un insieme di vincoli C . L'informativeness equivale alla frazione di vincoli in C che non sono soddisfatti dal clustering method A applicato su D .
- Coerenza di un insieme di vincoli:** Il grado di accordo tra i vincoli stessi ("quanto vanno d'accordo")

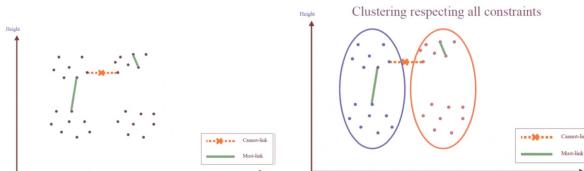
L'effetto dei vincoli sulla soluzione del clustering

I vincoli dividono l'insieme delle possibili soluzioni in due insiemi: fattibile e infattibile.

Sia S l'insieme di tutte le soluzioni: $S = S_F \cup S_I$

I vincoli riducono lo spazio di ricerca da S a S_F . Tutte le soluzioni all'interno di S_F hanno la proprietà comune di rispettare i vincoli.

OSS: Non è inaspettato trovare una soluzione con una proprietà desiderata e trovare questa soluzione velocemente. Questo perché andiamo a ridurre lo spazio delle soluzioni totali.



Constraint-Based Clustering Methods - Handling Hard Constraints

Rispettare strettamente i vincoli durante il clustering.

COP-k-means algorithm (variazione del k-means)

PASSO 1 - Generare una "super-istanza" per i vincoli must-link

- Si considerano le istanze che sono incluse in un must-link
- Si crea una super-istanza, che rimpiazza queste istanze, che rappresenta la media tra esse (In questo modo "vincoliamo" il fatto che queste istanze debbono essere all'interno dello stesso cluster).
- La super-istanza ha anche un peso che equivale al numero di oggetti che essa rappresenta (Importante perché quando eseguiamo il k-means, la super-istanza deve "attrarre" di più il centroide rispetto alle altre istanze essendo che rappresenta un gruppo di istanze).

PASSO 2 - Condurre un k-means modificato rispettando i vincoli cannot-link

Assegniamo ogni punto al centroide più vicino che rispetta i vincoli di "cannot-link".

sia

Constraint-Based Clustering Methods - Handling Soft Constraints

Viene trattato come un problema di ottimizzazione, quando un clustering viola un soft constraint, una penalità viene imposta al clustering.

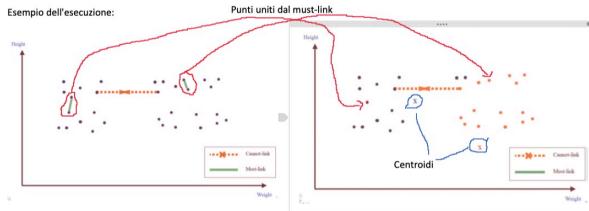
L'obiettivo generale è quindi di ottimizzare la qualità del clustering e minimizzare la penalità dovuta ai vincoli violati.

CVQE algorithm (Constrained Vector Quantization Error)
Conduce il k-means clustering mettendo delle penalità sulle violazioni dei vincoli.

Funzione obiettivo (cost function): Somma delle distanze usata in k-means, aggiustata dalle penalità che vengono calcolate nel seguente modo:

- Penalità della violazione di un vincolo must-link**
Se gli oggetti x e y devono essere collegati, ma vengono assegnati a due centroidi differenti c_1 e c_2 → Viene aggiunta come penalità alla funzione obiettivo la distanza tra i due centroidi: $dist(c_1, c_2)$.

- Penalità della violazione di un vincolo cannot-link**
Se gli oggetti x e y non devono essere collegati ma vengono assegnati allo stesso centroide c → Viene aggiunta come penalità alla funzione obiettivo la distanza, $dist(c, c')$, tra il centroide c e il cluster più vicino a c , chiamato c' .



Velocizzare il Constrained Clustering

E' costoso computare i constrained clustering perché dobbiamo sia eseguire il clustering che imporre i vincoli.
Definizione di visibilità: Un punto p è visibile da un altro punto q se la linea che congiunge p e q non interseca un ostacolo.

Visibility Graph

E' un grafo $VG = (V, E)$ t.c ogni vertice degli ostacoli ha un corrispondente nodo contenuto in V e due nodi, v_1 e v_2 , in V sono uniti da un arco contenuto in E se e solo se i vertici corrispondenti che rappresentano sono "visibili" tra loro.

Sia $VG' = (V', E')$ un visibility graph creato a partire da VG aggiungendo due punti addizionali p e q in V' . E' contiene un arco che unisce due punti in V' se due punti sono mutualmente visibili.

Il path più corto tra i due punti p e q sarà un subpath di VG' .

Ridurre il costo della computazione della distanza

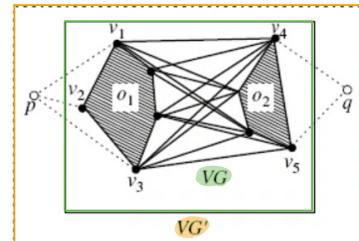
Per ridurre il costo della computazione della distanza tra qualsiasi paio di oggetti/punti, un metodo è di raggruppare i punti che sono vicini in dei microclusters. Questo è possibile andando a dividere in triangoli la regione R e successivamente raggruppando i punti vicini che sono nello stesso triangolo in microclusters, usando un metodo simile a BIRCH o DBSCAN.

Processando i microclusters invece che i punti individuali, il costo computazionale si riduce.

La precomputazione può essere svolta andando a costruire due tipi di "join indices" basati sulla computazione degli shortest paths:

- **WV indices** -> Per ogni paio di vertici degli ostacoli
- **MV indices** -> Per ogni paio tra i vertici del microcluster e vertici degli ostacoli.

Usando questi indici ci permette di ottimizzare la performance.



Outlier

Cos'è un outlier?

Un outlier è un oggetto che devia significativamente da un oggetto normale come se fosse stato generato da un diverso meccanismo.

Sono differenti dai dati rumorosi i quali rappresentano errori randomici durante la misurazione. Il rumore dovrebbe essere rimosso PRIMA di fare operazioni di rilevazione degli outlier.

Tipi di Outliers

Global Outliers	Sono oggetti che deviano significativamente dal resto dei dati nel dataset -> Bisogna trovare una misura appropriata per misurare la deviazione	Ad esempio l'identificazione di un intrusione all'interno di una rete informatica
Contextual Outliers	Sono oggetti che deviano significativamente basandosi su un determinato contesto -> Bisogna definire o formulare un contesto significativo	Ad esempio 35° gradi a Pisa di inverno, fossero stati di estate non ci sarebbero stati problemi
Collective Outliers	Sottoinsieme di dati che devia significativamente dall'intero dataset, anche se i singoli dati all'interno del sottoinsieme potrebbero non essere outliers singolarmente -> Necessita di avere un conoscenza sulla relazione tra i dati del dataset, come la distanza o similarità.	

Un dataset potrebbe avere diversi tipi di outlier. Un oggetto potrebbe appartenere a più di un tipo di outlier.

Outlier Detection

I metodi di rilevazione degli outlier possono essere:

1) Supervised Methods

Modellare l'outlier detection come un problema di classificazione.

Problemi possibili:

- Classi non bilanciate: Andiamo a generare artificial outliers per andare ad aumentare il numero di outliers e bilanciare le classi.
- Catturare il maggior numero di outliers: Il recall è più importante dell'accuracy, cioè non vogliamo classificare i nuovi oggetti "normali" come outliers.

OSS: Naturalmente non è solito andare a gestire questi problemi come supervised e sono difficili da gestire dovuto alla netta differenza di istanze tra le classi.

2) Unsupervised Methods

Assumere che gli oggetti "normali" sono raggruppati in cluster, ognuno dei quali ha qualche feature distintiva.

O spettiamo che un outlier sia lontano da ogni cluster di oggetti normali (non appartiene a nessun cluster)

DEBOLEZZA: Non possiamo rilevare la tipologia "Collective outliers" efficientemente.

PROBLEMI:

- 1) Difficile distinguere il rumore dagli outliers
- 2) Costoso perché dobbiamo prima eseguire il clustering e successivamente trovare gli outliers.
 - a. I metodi più nuovi, ad esempio DBSCAN cercano gli outliers direttamente.

3) Semi-Supervised Methods

Quando il numero di labeled data è poco numeroso.

- Se alcuni labeled normal object sono disponibili, si usano quest'ultimi per allenare un modello che riconosca i normal objects. Gli oggetti non riconosciuti dal modello vengono classificati come outliers.
- Se alcuni labeled outliers sono disponibili, potrebbero non essere necessari per allenare un modello che riconosca gli outlier. Solitamente ci si aiuta andando ad utilizzare dei modelli per oggetti normali trained tramite unsupervised methods.

4) Statistical Techniques

Assumiamo che il dati "normali" seguono un modello statistico (stochastic model) - la distribuzione che seguono.

I dati che non seguono questo modello vengono considerati come outliers.

EFFICACIA: Questo metodo dipende altamente dalla proximity measure scelta.

5) Proximity-Based Techniques

Sono tecniche basate sulla prossimità. Un oggetto è un outliers se il suo nearest neighbors è lontano, cioè la prossimità dell'oggetto devia significativamente dalla prossimità degli altri oggetti nello stesso dataset.

EFFICACIA: Questo metodo dipende altamente dalla proximity measure scelta.

PROBLEMI:

- In alcune applicazioni, la proximity/distance measure non può essere ottenuta facilmente.
- Solitamente ha difficoltà nel trovare gruppi di outliers che stanno vicini tra loro.

6) Clustering-Based Methods

Metodi basati sul clustering.

I dati normali appartengono a cluster grandi e densi, mentre gli outliers appartengono a cluster piccoli e sparsi, o non appartengono a nessun cluster.

PROBLEMA: Il clustering è costoso e non scala bene con grandi quantità di dati.

Approcci Statistici (Statistical techniques)

Gli approcci statistici assumono che gli oggetti nel dataset siano generati tramite un processo stocastico.

L'idea è di generare un modello che fitta i dati del dataset e usarlo per identificare gli oggetti che sono in aree con poca probabilità per quel modello, come outliers.

I metodi statistici si dividono in:

- **Parametrici:** Assumono che i dati "normali" sono generati da una distribuzione parametrica con parametro Theta. La probability density function $f(x, \theta)$ dà la probabilità che un certo oggetto x è generato da quella distribuzione. Più piccolo è il valore della funzione dato x , più è probabile che x sia un outlier.
- **Non-Parametrici:** Non assumono un modello statistico a-priori e determinano il modello dai dati in input).

Metodi Parametrici

Come rilevare gli outliers basati su una distribuzione normale in un dataset univariato (composto da un solo attributo)

Soltanente si assume che i dati siano generati da una normal distribution, si imparano i parametri dai dati in input e si identificano i punti con una bassa probabilità come outliers.

PASSO 1 - Determinare se la distribuzione è normale

Per determinare se la distribuzione è normale si possono usare due metodi:

- 1) **Shapiro-Wilk Test:** È un test per verificare se un esempio random proviene da una distribuzione normale
- 2) **Q-Q (Quantile-Quantile) plot:** Per verificare se un campione randomico viene da una distribuzione normale, si fa il plot delle quantiles corrispondenti ai campioni randomici in confronto alle quantiles di una distribuzione normale.

Sia il dataset composto da n samples randomici.

ESEMPIO:

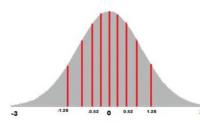
Siano gli n valori

7.19, 6.31, 5.89, 4.5, 3.77, 4.25, 5.19, 5.79, 6.79

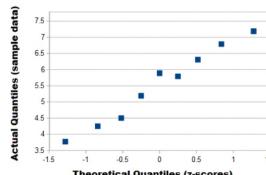
- Si ordinano gli n valori dal più piccolo al più grande
3.77, 4.25, 4.50, 5.19, 5.79, 5.89, 6.31, 6.79, 7.19

- Si disegna una normal distribution curve e la si divide in $n+1$ segmenti, dove n è il numero di valori. (Nel nostro caso, essendo 9 valori, ogni segmento è un 10% dell'area totale)

10% = -1.28
20% = -0.84
30% = -0.52
40% = -0.25
50% = 0
60% = 0.25
70% = 0.52
80% = 0.84
90% = 1.28
100% = 3.0



- Si fa il plot dei valori del dataset mettendoli a confronto con i punti di taglio della normal distribution. Una (quasi) linea dritta (diagonale) sul Q-Q plot indica che la distribuzione dei dati è approssimativamente normale.



Dopo aver verificato che la distribuzione sia normale si procede con i metodi parametrici.

PASSO 2 - Trovare gli outliers

1 Metodo - Modellare la distribuzione normale e considerare la probabilità che i punti appartengano a quella distribuzione

- 1) Si trovano μ e σ
 $\bar{x} = \text{media}$
 $s = \sqrt{\frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2}$
- 2) Si calcola per un determinato punto x :
 $y = \frac{x-\mu}{\sigma}$
- 3) Considerando che il 99.7% dei dati sono contenuti nella regione descritta da $\mu \pm 3\sigma$
L'oggetto x è un outlier se:
- $y > \mu + 3\sigma$
- $y < \mu - 3\sigma$

ESEMPIO:

145
125
190
135
220
130
210
3
165
165
150

Statistiche sul dataset

Valori del dataset

2 metodo - Il Grubbs' test (maximum normed residual test) - Il test trova se un valore minimo o massimo è un outlier.

Il test verifica la presenza di outliers calcolando il valore massimo delle differenze (in valore assoluto) tra i valori e il valore medio.

- 1) **Calcolare il G test statistic**
 - Ordinare i data points dal più piccolo al più grande
 - Trovare il valore medio (\bar{x}) e la standard deviation (s) dei dati
 - Calcolare il G test statistic, usando la seguente equazione:

$$G = \frac{\max_{i=1,\dots,N} |Y_i - \bar{Y}|}{s}$$

Ciò prende il max value della differenza tra un sample e la media, e lo divide per la standard deviation (s). Analizzeremo poi se il sample, corrispondente al max value, è un outlier o meno.

- 2) **Calcolare il G critical value**

Esistono diverse tabelle per trovare il critical value per il Grubbs' test.

Si usano un confidence level fisso e il numero di samples nel dataset per trovare il critical G-value.

- 3) **Comparare il test statistic (il G calcolato da noi) al G critical value:**
 - Se $G_{test} < G_{critical}$: Mantieni il sample nel dataset, esso non è un outlier (per il confidence level che abbiamo settato)
 - Se $G_{test} \geq G_{critical}$: Rimuovi il sample come outlier

corrispondente a G

Nella tabella troviamo che $G_{crit} = 2.23$.

Essendo che:

- $G = 2.52$
- $G_{crit} = 2.23$

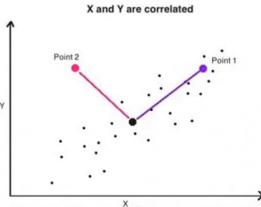
$G > G_{crit}$, di conseguenza G è un outlier (cioè il valore 3 è un outlier)

Come rilevare gli outliers basati su una distribuzione normale in un dataset multivariato (composto da più attributi)

Si trasforma il task del multivariate outlier detection in un task univariate outlier detection.

Metodo 1 - Usare la Mahalanobis distance

La Mahalanobis distance è la distanza tra un punto e una distribuzione (non tra due punti distinti). Essa è la versione multivariata della distanza euclidiana.
Non possiamo usare quest'ultima perché non riesce a contraddistinguere se due punti avendo la stessa distanza dal centroide di una distribuzione appartengono o meno alla distribuzione stessa.



Entrambi i punti hanno la stessa distanza del centroide, ma uno appartiene al clustering e uno no. L'euclidean distance non riesce a distinguere questi casi.

La mahalanobis distance esegue i seguenti step:

- 1) Transforma le colonne in variabili non correlate
- 2) Scala le colonne in modo da rendere la loro varianza uguale a 1
- 3) Infine, calcola la Euclidean distance

Mahalanobis distance

Sia \bar{o} il vettore media di un dataset multivariato.

La mahalanobis distance di un oggetto o da \bar{o} è definita come:

$$MDist(o, \bar{o}) = (o - \bar{o})^T S^{-1}(o - \bar{o})$$

Dove:

S = covariance matrix

OSS: Dividere per la matrice di covarianza è essenzialmente la versione multivariata della standardizzazione $z = \frac{x - \text{mean}}{\text{std}}$

Se le variabili sono fortemente correlate, la covarianza sarà alta e la distanza sarà ridotta. Invece, se la covarianza è bassa la distanza non verrà ridotta. (in questo modo viene gestita la distanza, considerando l'appartenenza ad un cluster).

Rilevazione degli outliers utilizzando la mahalanobis distance

- 1) Calcolare il vettore delle medie per un dataset multivariato
- 2) Per ogni oggetto o, calcolare $MDist(o, \bar{o})$
- 3) Rilevare gli outliers nel transformed dataset univariato. Questo dataset è composto dalle $MDist(o, \bar{o})$ dei vari oggetti. In questo modo trasformiamo il problema da multivariato a univariato.
- 4) Se $MDist(o, \bar{o})$ è considerato un outlier, allora anche o è considerato un outlier. Per trovare gli outlier si usa il Grubb's test sulle $MDist(o, \bar{o})$.

Debolezze

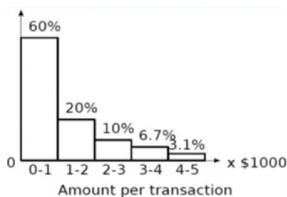
- Computazionalmente costoso (calcolo della matrice di covarianza e dell'inversa)
- Necessità di salvare la covariance matrix e la sua inversa in memoria

Metodi non parametrici

Il modello normale dei dati è imparato direttamente dai dati in input, senza una struttura apriori. Non facciamo assunzioni sulla distribuzione dei dati.

Detection using Histogram

Consideriamo che i dati non inclusi nei bin (o con % molto piccole) sono outliers.



PROBLEMA: Scegliere la giusta dimensione dei bin per l'histogram

- Se scegliamo bin troppo piccoli: I normal object in bin vuoti/rari vengono considerati come outliers (falsi positivi)
- Se scegliamo bin troppo grandi: Gli outlier all'interno di bin frequenti vengono considerati oggetti normali (falsi negativi).

SOLUZIONE: Adottare un kernel density estimation per stimare la distribuzione della densità di probabilità dei dati. Se la densità stimata è alta, allora è probabile che l'oggetto sia normale. Altrimenti, è probabile sia un outlier.

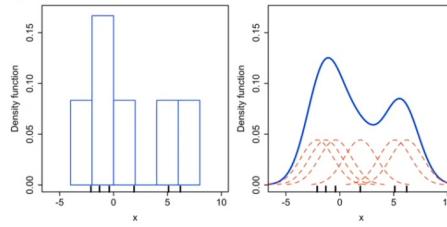
$\hat{f}_h(o)$ alta \rightarrow obj normale, $\hat{f}_h(o)$ bassa \rightarrow obj outlier

Kernel Density Estimation

$$\hat{f}_h(x) = \frac{1}{n} \sum_{i=1}^n K_h(x - x_i) = \frac{1}{nh} \sum_{i=1}^n K\left(\frac{x - x_i}{h}\right),$$

↑
La Kernel function viene utilizzata per modellare l'influenza
di un punto sul suo vicinato

where $K(\cdot)$ is the kernel — a symmetric but not necessarily positive function that integrates to one — and $h > 0$ is a smoothing parameter called the bandwidth.



Costo computazionale dei metodi statistici

Il costo computazionale dei modelli statistici dipende dal tipo di modello.

I modelli parametrici semplici (ad esempio Gaussian), richiedono tempo lineare.

I modelli più sofisticati (ad esempio kernel density estimation) il costo può essere anche quadratico.

Quando un modello è allenato, il costo di predizione di un outlier è solitamente molto basso per un dato oggetto.

Approcci proximity-based

Questi approcci si basano sull'assunzione che gli outlier siano significativamente distanti dagli altri punti del dataset. Due tipi di metodi: **Distance-based** e **Density-based**.

Distance based

Un oggetto o è considerato un outlier se il suo vicinato non contiene abbastanza altri punti.

Per ogni oggetto o , si esamina il numero di oggetti che ci sono nel suo r -neighborhood, dove r è una soglia specificata dall'utente che indica la distanza massima.

Un oggetto o è un outlier se la maggior parte (prendendo π come una soglia "frazionaria" che indica la soglia minima di punti nel vicinato di o) degli oggetti in D sono lontani da o , cioè non sono in r -neighborhood di o . Abbiamo quindi due parametri da settare:

r	Indica la massima distanza tra due oggetti, cioè qual è il raggio del vicinato di un certo oggetto o
π	Indica il minimo numero di oggetti che devono essere nel vicinato dell'oggetto o

Tradotto in formula, o è un outlier se:

$$\frac{\|\{o' | dist(o, o') \leq r\}\|}{\|D\|} \leq \pi$$

solo forma di ratio
[o, 1]

PROBLEMA: Andare a settare i parametri nel modo corretto, a seconda di come li setto trovo outliers diversi.

Equivalentemente, si può verificare se o è un outlier anche andando a verificare la distanza tra o e il suo k -th vicino più vicino o_k , dove

$$k = \lceil \pi \|D\| \rceil$$

o è un outlier se $dist(o, o_k) > r$

ATTENZIONE: L'algoritmo è costoso perché per ogni oggetto dobbiamo considerare la distanza tra l'oggetto e gli altri oggetti, il vicinato ecc...

Velocizzare la computazione

- 1) Per ogni oggetto o_i , viene calcolata la sua distanza dagli altri oggetti e il # di altri oggetti nel suo r -neighborhood.
- 2) Se $\pi * n$ altri oggetti sono all'interno del vicinato, terminiamo il loop interno $\rightarrow o_i$ non è un outlier

Altrimenti o_i è un outlier

COSTO: Lineare nella dimensione dei dati essendo che per la maggior parte degli oggetti che non sono outlier, il loop interno termina prima.

Method:

```

for i = 1 to n do
    count ← 0
    for j = 1 to n do
        if i ≠ j and dist(oi, oj) ≤ r then
            count ← count + 1 → Conto il numero di vicini
        if count ≥ π · n then
            exit {oi; cannot be a DB(r, π) outlier}
        endif
    endif
endfor
print oi; {oi is a DB(r, π) outlier according to (Eq. 12.10)}
endfor;

```

outlier

Non è comunque abbastanza perché l'insieme di oggetti non può essere mantenuto interamente in memoria e quindi abbiamo il costo dovuto allo swapping tra la memoria e il disco. Dovremmo andare a testare ogni oggetto solo con i suoi vicini e fare il check non oggetto per oggetto ma in gruppi. Usiamo il Grid-based method (CELL).

Density based

Un oggetto o è un outlier se la sua densità è relativamente molto inferiore rispetto ai suoi vicini.

Si considera la densità IN RELAZIONE con i suoi vicini perché se siamo in una regione dello spazio in cui i dati sono molto sparsi, la distanza tra un oggetto e gli altri oggetti sarà maggiore. Di conseguenza si considera la densità anche dei vicini per controllare se la densità è bassa anche per loro.

Intuizione: La densità intorno ad un outlier è significativamente diversa dalla densità intorno ai suoi vicini.

Metodo: Usare la densità relativa di un oggetto rispetto ai suoi vicini come indicatore del grado per cui un oggetto è un outlier.

Definizioni:

$$dist_k(o) \rightarrow N_k(o) = \{o' | o' \in D, dist(o, o') \leq dist_k(o)\}$$

Distanza tra o e il suo k -th nearest neighbor
oggetto all'interno del raggio definito dalla dist con le k -NN

Reachability distance tra o' e o :

$$reachdist_k(o \leftarrow o') = \max\{dist_k(o), dist(o, o')\}$$

Local reachability density di o :

$$lrd_k(o) = \frac{\|N_k(o)\|}{\sum_{o' \in N_k(o)} reachdist_k(o' \leftarrow o)}$$

più è grande, più è prob. che l'obj sia un outlier

Local Outlier Factor (LOF)

Il LOF di un oggetto è la media della local reachability di o e i k -nearest neighbors di o

Più piccola è la local reachability density di o , maggiore è la local reachability density dei KNN di o → LOF è maggiore

Questo cattura un local outlier la cui local density è relativamente bassa in comparazione con le local density dei suoi KNN.

LOF ci permette di rilevare gli outlier a seconda della densità intorno ad essi. È uno degli approcci più popolari per trovare gli outliers. Facile da implementare, computazione costosa, ma può essere molto effettivo soprattutto nei casi dei local outliers.

* For object o , let

$$direct_{min}(o) = \min\{reachdist_k(o' \leftarrow o) | o' \in N_k(o)\}$$

$$direct_{max}(o) = \max\{reachdist_k(o' \leftarrow o) | o' \in N_k(o)\}$$

be the minimum and maximum
reachability distance. For the
neighbor of o 's k -nearest neighbors

$$indirect_{min}(o) = \min\{reachdist_k(o'' \leftarrow o') | o' \in N_k(o) \text{ and } o'' \in N_k(o')\}$$

$$indirect_{max}(o) = \max\{reachdist_k(o'' \leftarrow o') | o' \in N_k(o) \text{ and } o'' \in N_k(o')\}$$

In can be shown that LOF(o) is bounded

$$\frac{direct_{min}(o)}{indirect_{max}(o)} \leq LOF(o) \leq \frac{direct_{max}(o)}{indirect_{min}(o)}$$

LOF captures the relative
distance of an object

Esempio:

Consider the following 4 data points

$$a(0,0), b(0,1), c(1,1), d(3,0)$$

Compute the distance between the four points (Manhattan distance)

$$dist(a,b) = 1$$

$$dist(a,c) = 2$$

$$dist(a,d) = 3$$

$$dist(b,c) = 1$$

$$dist(b,d) = 4$$

$$dist(c,d) = 3$$

Let us consider $k=2$. Then

$$dist_2(a) = dist(a,c) = 2 \quad N_2(a) = \{b,c\}$$

$$dist_2(b) = dist(b,a) = 1 \quad N_2(b) = \{a,c\}$$

$$dist_2(c) = dist(c,a) = 2 \quad N_2(c) = \{b,a\}$$

$$dist_2(d) = dist(d,a) = 3 \quad N_2(d) = \{a,c\}$$



Grid-based method (CELL)

Lo spazio dei dati viene partitionato in una griglia multi dimensionale.

Ogni cella è un hypercube con una lunghezza diagonale pari a $\frac{r}{2}$.

Proprietà delle celle di livello-1 e di livello-2:

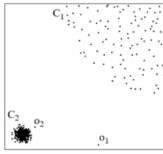
- **Livello-1:** Per ogni possibile punto x nella cella C e ogni altro possibile punto y a una cella di distanza da C la distanza è così descritta: $dist(x, y) \geq r$
 - **Livello-2:** Per ogni possibile punto x nella cella C e ogni altro possibile punto y a due celle di distanza da C , la distanza è così descritta: $dist(x, y) \geq r$
- Sia a il numero di oggetti nella cella C
Sia b_1 il numero totale di oggetti nelle celle level-1
Sia b_2 il numero totale di oggetti nelle celle level-2
- Sia usano le seguenti regole di pruning:
- **Level-1 pruning rule:** Se $a + b_1 > [m]$, allora ogni oggetto o in C non è un outlier perché tutti questi oggetti in C e le celle level-1 sono nel r -neighborhood di o e ci sono almeno $[m]$ vicini.
 - **Level-2 pruning rule:** Se $a + b_1 + b_2 < [m] + 1$, allora ogni oggetto in C sono outliers perché ogni vicinato di ogni oggetto ha meno di $[m]$ altri oggetti.

In questo modo dobbiamo controllare solo gli oggetti che non possono essere pruned, e per ognuno di questi oggetti o , dobbiamo solo calcolare la distanza tra o e gli oggetti nelle celle di level-2 (essendo che nelle celle di level-2, la distanza da o è maggiore di r).

Problema dei metodi che usano la distanza per rilevare gli outlier

Essendo che questi metodi utilizzano solo la distanza, abbiamo dei problemi quando dobbiamo gestire zone con diverse densità. Sto considerando dei parametri globali per l'intero spazio).

Quello che può risultare un outlier in uno spazio molto denso, potrebbe non essere un outlier in uno spazio poco denso.



In questo esempio:
 o_2 risulta essere un outlier per il cluster C_2 ma ha la stessa densità del cluster C_1

SOLUZIONE: Outlierness.

Let $N_k(x_i)$ be the k -nearest neighbors of x_i

Let $D_k(x_i)$ be the average distance to k -nearest neighbors

$$D_k(x_i) = \frac{1}{k} \sum_{j \in N_k(x_i)} \|x_i - x_j\|$$

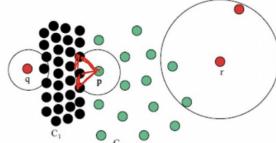
Outlierness is the ratio of $D_k(x_i)$ to average $D_k(x_j)$ for its neighbors j

$$O_k(x_i) = \frac{D_k(x_i)}{\frac{1}{k} \sum_{j \in N_k(x_i)} D_k(x_j)} \rightarrow \text{Media della } D_k(x_j) \text{ dove } x_j \text{ è un vicino di } x_i \text{ e } D_k(x_j) \text{ corrisponde alla distanza media tra } x_j \text{ e i suoi vicini.}$$

If outlierness > 1 , x_i is further away from neighbors than expected.

L'outlierness ci permette di comparare l'average distance dell'oggetto ai suoi vicini, con l'average distance dei suoi vicini, ai loro vicini.

PROBLEMA: Quando i clusters sono troppo vicini, outlierness ci restituisce risultati non intuitivi.



In questo esempio:
Il vicinato di P è composto da punti che fanno parte di una zona con alta densità e quindi la distanza tra questi punti è molto piccola.
Di conseguenza essendo che la distanza tra P e questi punti è più grande della distanza tra i punti fra loro, P viene considerato un outlier.

$$lrd_2(a) = \frac{\|N_2(a)\|}{reachdist_2(b \leftarrow a) + reachdist_2(c \leftarrow a)}$$

$$reachdist_2(b \leftarrow a) = \max\{dist_2(b), dist(b, a)\} = \max\{1, 1\} = 1$$

$$reachdist_2(c \leftarrow a) = \max\{dist_2(c), dist(c, a)\} = \max\{2, 2\} = 2$$

Thus

$$lrd_2(a) = \frac{\|N_2(a)\|}{reachdist_2(b \leftarrow a) + reachdist_2(c \leftarrow a)} = \frac{2}{1+2} = 0.667$$

$$lrd_2(b) = \frac{\|N_2(b)\|}{reachdist_2(a \leftarrow b) + reachdist_2(c \leftarrow b)} = \frac{2}{2+2} = 0.5$$

$$lrd_2(c) = \frac{\|N_2(c)\|}{reachdist_2(b \leftarrow c) + reachdist_2(a \leftarrow c)} = \frac{2}{1+2} = 0.667$$

$$lrd_2(d) = \frac{\|N_2(d)\|}{reachdist_2(a \leftarrow d) + reachdist_2(c \leftarrow d)} = \frac{2}{3+3} = 0.33$$

$$LOF_2(a) = (lrd_2(b) + lrd_2(c)) (reachdist_2(b \leftarrow a) + reachdist_2(c \leftarrow a)) \\ = (0.5 + 0.667)(1+2) = 3.501$$

$$LOF_2(b) = (lrd_2(a) + lrd_2(c)) (reachdist_2(a \leftarrow b) + reachdist_2(c \leftarrow b)) \\ = (0.667 + 0.667)(2+2) = 5.336$$

$$LOF_2(c) = (lrd_2(b) + lrd_2(a)) (reachdist_2(b \leftarrow c) + reachdist_2(a \leftarrow c)) \\ = (0.5 + 0.667)(1+2) = 3.501$$

$$LOF_2(d) = (lrd_2(a) + lrd_2(c)) (reachdist_2(a \leftarrow d) + reachdist_2(c \leftarrow d)) \\ = (0.667 + 0.667)(3+3) = 8.004$$

The sorted order is,

$$LOF_2(d) = 8.004$$

$$LOF_2(b) = 5.336$$

$$LOF_2(a) = 3.501$$

$$LOF_2(c) = 3.501$$

Obviously, top 1 outlier is point d .

Clustering-based methods

I cluster-based methods classificano un oggetto come outlier se

1) NON APPARTIENE A NESSUN CLUSTER

Si usano clustering methods density-based, come ad esempio il DBSCAN

PROBLEMA: Dobbiamo identificare i giusti valori di EPS e Min_{pts} , di conseguenza è un metodo molto dipendente dai parametri scelti.

2) C'è una grossa distanza tra un oggetto e il cluster più vicino

Si usa il k-means partizionando i dati in cluster e per ogni oggetto o , si assegna un "outlier score" basato sulla distanza dal c_{vicino} più vicino.

Se $\frac{dist(o, c_o)}{avg.dist(c_o)}$ è grande (maggiore di una certa soglia fissata), allora è probabile che o sia un outlier.

PROBLEMA: Determinare il giusto valore di K , fissare una soglia per determinare se un oggetto è un outlier.

3) Appartiene a un cluster piccolo o sparso -> Si usa il CBLOF

CBLOF (Cluster-Based Local Outlier Factor)

Tutti gli approcci visti finora non permettono la rilevazione di cluster di outliers.

CBLOF è un outlier factor che è associato ad ogni oggetto e misura sia la grandezza del cluster a cui l'oggetto appartiene, sia la distanza tra l'oggetto e il cluster più vicino.

- 1) Trova i cluster in un dataset e li ordina per decreasing size.
- 2) L'algoritmo assume che la maggior parte dei punti non siano outliers e definisce un parametro alpha [0,1] per distinguere i cluster grandi da quelli piccoli.
 - a. Large clusters (LC) = Clusters che contengono almeno una percentuale alpha (ad esempio il 90%) dei dati del dataset principale.
 - b. Small clusters (SC) = i restanti clusters.

Per ogni oggetto o, assegno un CBLOF, definito come:

$$CBLOF(o) = \begin{cases} |C_i| \cdot \min(\text{distance}(o, c_j)) \text{ dove } o \in C_i, C_i \in SC \text{ e } C_j \in LC & \rightarrow \text{caso l'oggetto } o \text{ appartiene a uno small cluster} \\ |C_i| \cdot \min(\text{distance}(o, c_i)) \text{ dove } o \in C_i, C_i \in LC & \rightarrow \text{caso l'oggetto } o \text{ appartiene a un large cluster} \end{cases} \rightarrow \text{Prodotto dimensione cluster}$$

e la similitudine tra il punto
e il LC più vicino

Per il calcolo della distanza tra l'oggetto e il cluster, è sufficiente adottare la misura di similarità usata nel cluster algorithm

FindCLOB algorithm:

- 1) Fare il cluster del dataset
- 2) Calcolare il valore CBLOF per ogni oggetto

Se CBLOF è un valore alto, allora o potrebbe essere un outlier.

Classification-based Methods

Metodo 1 - One-class Model

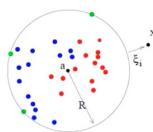
Un classificatore viene costruito per descrivere solo la classe "normale".

Tutti gli oggetti che non appartengono alla classe normale vengono considerati come outliers.

L'approccio più popolare è il Support Vector Data Description (SVDD).

Support Vector Data Description (SVDD)

- 1) Costruisce una hyper-sphere (di raggio minimo) intorno alla classe positiva dei dati che circonda quasi tutti i punti del dataset.
- 2) Il classificatore SVDD rifiuta un test point come outlier se esso è fuori dalla hyper-sphere. Tuttavia, SVDD può rifiutare alcune frazioni di dati classificati positivi quando il volume della hyper-sphere diminuisce.



Un altro approccio è il Nearest Neighbour Description (NN-d), una variante del Nearest Neighbor Method

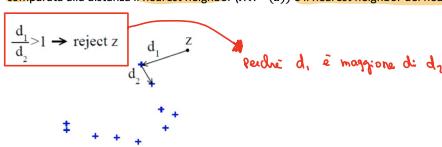
Nearest Neighbour Description (NN-d)

Un test object z è accettato come membro della target class quando la sua densità locale è maggiore o uguale alla densità locale del suo nearest neighbor del training set.

Funzione di accettazione:

$$f_{NN^{tr}}(z) = I\left(\frac{\|z - NN^{tr}(z)\|}{\|NN^{tr}(z) - NN^{tr}(NN^{tr}(z))\|}\right)$$

Essa rappresenta che la distanza da un oggetto z e il suo nearest neighbor nel training set ($NN^{tr}(z)$) è comparsa alla distanza il nearest neighbor ($NN^{tr}(z)$) e il nearest neighbor del nearest neighbor.



Mining Collective Outliers

Collective outlier sono gruppi di oggetti che deviano significativamente dall'intero dataset. Necessita di esaminare la struttura del dataset, cioè la relazione tra oggetti multipli.

Ognuna di queste strutture è innata alla rispettivo tipo di dati:

- Temporal data = Esploriamo strutture che sono formate dal tempo (time-series)
- Spatial data = Esploriamo aree locali
- Graph e network data = Esploriamo subgraphs

Le strutture spesso non sono esplicitamente definite, ma devono essere scoperte.

Ci sono due tipi di metodi per gestire i collective outliers:

- 1) Ridurre il problema ad un convenzionale outlier detection
 - a. Si identificano le strutture, si trattano come data object ed estraiamo gli attributi
 - b. Successivamente eseguiamo l'outlier detection su un insieme di "oggetti strutturati" usando le features estratte
 - 2) Modellare direttamente le strutture, la collective outlier detection è sottile a causa dell'esplorazione delle strutture.
 - L'esplorazione è euristica e può essere application dependent
 - Il costo computazionale è spesso alto dovuto al sofisticato mining process
- Esempio:
- Ex. 1: Detect collective outliers in online social network of customers
- Treat each possible subgraph of the network as a structure unit
 - Collective outlier: An outlier subgraph in the social network
 - Small subgraphs that are of very low frequency
 - Large subgraphs that are surprisingly frequent

Vantaggi

- Permette di rilevare gli outlier senza che i dati siano labeled
- Permette di lavorare con diversi tipi di dati
- I cluster possono essere considerati come riassunti dei dati
- Quando un cluster è ottenuto, dobbiamo solo comparare un oggetto rispetto ad un cluster per determinare se è un outlier (veloce)

Svantaggi

- L'efficacia dipende altamente dal tipo di clustering method usato (potrebbero non essere ottimizzati per l'outlier detection, forma dei cluster ecc.)
- Computazionalmente costoso perché dobbiamo determinare i clusters
 - Per ridurre il costo un metodo è il Fixed-width clustering
 - Un punto è assegnato a un cluster se il centro del cluster è a una distanza minore di una certa soglia dal punto
 - Se il punto non può essere assegnato a nessun cluster esistente, un nuovo cluster viene creato e la soglia della distanza potrebbe essere imparata dal training data sotto alcune condizioni

Prodotto dimensione cluster

e la similitudine tra il punto

e il LC più vicino

Metodo 2 - Semi-Supervised Learning

Combinando i metodi classification-based e clustering-based.

Usiamo i labeled data dopo aver applicato l'algoritmo di clustering per determinare gli outliers.

- 1) Usare un approccio clustering-based per trovare un grosso cluster C e un piccolo cluster C₁
- 2) Essendo che alcuni oggetti in C portano il label "normal", andiamo a trattare gli oggetti in C come normali
- 3) Tutti gli oggetti che finiscono al di fuori di C vengono considerati come outliers (quelli in C₁ perché alcuni oggetti avranno la label "outlier")
- 4) Usiamo un one-class model su C per trainare un modello che riconosca gli outliers (per andare a determinarli su nuove istanze - test set)

Vantaggi e svantaggi dei metodi classification-based per l'outlier detection

Vantaggi

- L'outlier detection è veloce

Svantaggi

- La qualità dipende altamente dalla disponibilità e qualità del training set. È solitamente difficile ottenere training data rappresentativa e di buona qualità

Mining contextual outliers

Un oggetto è un contextual outlier se devia significativamente rispetto al contesto dell'oggetto. Il contesto è definito dai contextual attributes (tempo, zona geografica ecc...).

Le caratteristiche degli oggetti vengono definite attraverso i behavioral attributes e vengono utilizzati inoltre per valutare se un oggetto è un outlier nel contesto in cui appartiene.

METODO 1 - Il contesto può essere chiaramente identificato

Quando il contesto può essere facilmente identificato, dato un oggetto analizziamo i suoi valori (per i behavioral attributes) e li confrontiamo con i valori degli altri oggetti che appartengono allo stesso gruppo.

- 1) Identificare il contesto degli oggetti usando gli attributi contestuali
- 2) Usare un outlier detection method convenzionale sui behavioral attributes per determinare se l'oggetto è un outlier. (tenendo in considerazione il contesto)

PROBLEMA: Se il contesto che stiamo analizzando contiene veramente pochi oggetti o nessuno, non possiamo determinare se un oggetto è un outlier o meno.

SOLUZIONE: Generalizziamo il contesto

- 1) Imparare un mixture model U sugli attributi contestuali
- 2) Imparare un modello M sugli attributi comportamentali
- 3) Imparare una mappatura $p(V_i|U_j)$: La probabilità che un oggetto o che appartiene al cluster di attributi U_j è generato dal cluster di attributi V_i .
- 4) Outlier score

$$S(o) = \sum_{U_j} p(o \in U_j) \sum_{V_i} p(o \in V_i) p(V_i|U_j)$$

behavior attributes value ↑

METODO 2 - Il contesto non può essere chiaramente identificato

Si modella il comportamento "normale" in rispetto al contesto.

- 1) Usando un training set, alleniamo un modello che predice il comportamento aspettato in rispetto ai contextual attribute
- 2) Un oggetto è un contextual outlier se i suoi attributi comportamentali deviano significativamente dai valori predetti dal modello.

Usando un modello che linka contesto e comportamento, questi metodi permettono di non specificare il contesto.

Outlier detection in High Dimensional Data

All'aumentare della dimensionalità dei dati abbiamo il problema che i dati si allontanano, diventano più sparsi ed è quindi difficile utilizzare i concetti di distanza e densità.

Metodo 1 - Rilevare gli outliers nell'intero spazio

Si lavora direttamente nell'intero spazio con l'algoritmo chiamato HillOut

HillOut

Cerca gli outliers basati sulla distanza usando ranks of distance invece che l'absolute distance durante l'outlier detection:

- Per ogni oggetto o , trova i KNN di o ($nn_1(o), \dots, nn_k(o)$)
- Il peso dell'oggetto o è descritto come:

$$w(o) = \sum_{i=1}^k dist(o, nn_i(o)) \quad \left. \begin{array}{l} \text{Somma delle distanze con i vicini} \\ \text{vicini} \end{array} \right.$$

- Tutti gli oggetti sono rankati in ordine decrescente del peso
- I primi l oggetti sono considerati outlier \rightarrow perché hanno una dist. magg. con i vicini

ATTENZIONE: l è un parametro specificato dall'utente

Metodo 3 - Cercare gli outliers in subspaces

Ci permette di trovare gli outlier in sottospazi di dimensionalità minore. Questo ci permette di interpretare in modo facile perché è in che misura quell'oggetto è un outlier.

Grid-based subspace outlier detection method

Proiettare i dati in vari subspaces per trovare un'area dove la densità è minore alla media

- Discretizzare i dati in griglie (regioni) con φ ugual frequenza (approssimativamente lo stesso numero di istanze)
 - Cerchiamo regioni che sono significativamente sparse
 - Consideriamo un $k-d$ cube: K ranges su K dimension con n oggetti
 - Se gli oggetti sono indipendentemente distribuiti, il numero previsto di oggetti che cade in una regione a $k-d$ dimensione è:
$$\binom{1}{\varphi} n = f^k n \rightarrow \varphi = \frac{1}{f}$$

La deviazione standard è:

$$\sqrt{f^k(1-f^k)n}$$
 - Il coefficiente di sparsità del cubo C equivale a:
- $$S(C) = \frac{n(C) - f^k n}{\sqrt{f^k(1-f^k)n}}$$
- d. Se $S(C) < 0$, C contiene meno oggetti del previsto. Più $S(C)$ è negativo, più C è sparso e più è probabile che gli oggetti in C siano outliers nel sottospazio.

OSS:

- L'algoritmo trova le m grid cells (proiezioni) con il minor coefficiente di sparsità
- Costo: $O(\varphi^k)$
- Parametri:
 - m = Numero di griglie restituite
 - φ = Dimensionalità delle celle
- Modello grossolano

Metodo 2 - Dimensionality reduction

E' possibile utilizzare questo metodo solo quando andando a ridurre la dimensionalità dello spazio è comunque possibile distinguere gli outliers dalle istanze normali.

PCA

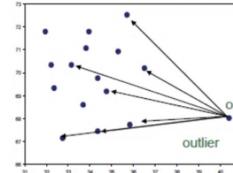
Eruristicamente, i principal components con bassa varianza sono preferibili perché, su tali dimensioni, gli oggetti normali sono probabilmente vicini l'uno all'altro e gli outlier spesso deviano dalla maggioranza

Metodo 4 - Cerca direttamente gli outliers in high-dimensional space

Non usa misure di prossimità, ma adotta una nuova euristica che non deteriora con high-dimensional data. Utilizza l'idea degli angoli. Gli angoli sono più stabili nelle distanze nei high dimensional spaces.

IDEA:

- Un oggetto o è un outlier se la maggior parte degli altri oggetti sono situati in una direzione simile (poca variazione dell'angolo della retta che collega l'oggetto agli altri oggetti).
- Un oggetto o non è un outlier se molti altri oggetti sono situati in direzioni diverse (molta variazione dell'angolo della retta che collega l'oggetto agli altri oggetti).



Modello:

- Dato un certo punto p si considera l'angolo tra px e py per ogni coppia xy del database.
- Si considera lo spettro di questi angoli
- L'ampiezza dello spettro è un valore per l'outlierness di un punto
- Si misura la varianza dello spettro dell'angolo pesata sulla distanza corrispondente
- Si calcola il Angle-based outlier factor (ABOF):

$$ABOF(o) = VAR_{x,y \in D, x \neq o, y \neq o} \frac{\langle ox, oy \rangle}{dist(o,x)^2 dist(o,y)^2}$$

- Proprietà:
 - ABOF piccolo \Rightarrow outlier \rightarrow angolo piccolo e dist. grande
 - ABOF grande \Rightarrow no outlier \rightarrow angolo grande e dist. piccola

Complessità: $O(n^3)$

OSS: Può essere generalizzato per gestire tipi arbitrari di dati



Clustering Graph and Network Data

I network vengono rappresentati dai grafi.

Basi di un network

Oggetti	Gli oggetti sono i vertici del grafo
Interazioni	Le interazioni vengono rappresentate dagli archi
Network	L'intero network è il grafo stesso

Clustering Graphs and Network Data

Possiamo applicare gli algoritmi di clustering standard introducendo una specifica definizione della misura di similarità:

- Geodesic distances
- Distance based on random walk (SimRank)

Geodesic Distance

La Geodesic Distance misura la distanza tra due vertici di un grafo, cioè il più breve path tra i due vertici.

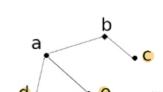
Geodesic_Distance(A,B)	Minimo numero di archi che collegano A e B. Se non sono connessi, il valore è infinito.
Eccentricity di v , eccen(v)	La più grande geodesic distance tra v e un qualsiasi vertice u appartenente al grafo
Raggio del grafo G	La minima eccentricity di tutti i vertici, cioè la distanza tra il punto più centrale e il suo bordo più lontano
Diametro del grafo G	La massima eccentricity di tutti i vertici, cioè più grande distanza tra ogni paio di vertici in G
Peripheral vertex	E' un vertice che raggiunge il diametro

E.g., eccen(a) = eccen(b) = 2; eccen(c) = eccen(d) = eccen(e) = 3

E.g., radius (g) = 2

E.g., diameter (g) = 3

E.g., Vertices c, d, and e are peripheral vertices



La geodesic distance ci dice solo qual è la distanza tra due vertici in termini di path più breve.
Dobbiamo estendere la geodesic distanze in modo da ottenere una misura per la similarità tra nodi (in modo da poter applicare i tipici clustering algorithms).

La geodesic distance può assumere due significati

- 1) **Structural context-based similarity:** Due vertici sono considerati simili se i loro vicini sono simili
- 2) **Similarity based on random walk:** La vicinanza tra due vertici viene misurata come la probabilità che un vertice riceva un'informazione da entrambi i vertici che stiamo analizzando

SimRank based on Structural context-based similarity

Sia un $G = (V, E)$ un grafo diretto:

- Individual **in-neighborhood** di un vertice v
 $I(v) = \{u \mid (u, v) \in E\}$
 (insieme dei nodi che hanno un arco entrante nel vertice v)
- Individual **out-neighborhood** di un vertice v
 $O(v) = \{w \mid (v, w) \in E\}$
 (insieme dei nodi che hanno un arco entrante dal vertice v)

Il SimRank è definito come:

Data una qualsiasi coppia di due vertici nel grafo, (u, v) , abbiamo:

- $s(u,v) = 0$ se i vertici u e v non hanno nessun vicino.
- $s(u,v) = 1$ se $u = v$

$$s(u, v) = \frac{C}{|I(u)||I(v)|} \sum_{x \in I(u)} \sum_{y \in I(v)} s(x, y) \quad \text{Se } u \neq v \quad C = \text{costante compresa tra 0 e 1}$$

OSS: SimRank è un valore compreso tra 0 e 1.

Come si computa il SimRank?

Iterativamente si computa l'equazione precedente fin quando un punto prefissato non viene raggiunto.

Sia n il numero di nodi nel grafo G

- 1) Per ogni iterazione i manteniamo n^2 entries $s_i(*, *)$, dove $s_i(u, v)$ da lo score tra u e v all'iterazione i (uno score per ogni coppia di vertici del grafo)
- 2) Iniziamo da $s_0(*, *)$ dove ogni $s_0(u, v)$ è un limite inferiore sull'attuale SimRank score $s(u, v)$:

$$s_0(u, v) = \begin{cases} 0 & \text{if } u \neq v \\ 1 & \text{if } u = v. \end{cases}$$

- 3) Per calcolare $s_{i+1}(u, v)$ da $s_i(*, *)$ usiamo:

$$s_{i+1}(u, v) = \frac{C}{|I(u)||I(v)|} \sum_{x \in I(u)} \sum_{y \in I(v)} s_i(x, y) \quad \text{if } u \neq v$$

$$s_{i+1}(u, v) = 1 \quad \text{if } u = v.$$

OSS: I valori di $s_i(*, *)$ non diminuiscono con l'aumentare di i

Al termine dell'algoritmo, per ogni pair all'interno al graph abbiamo il SimRank calcolato.

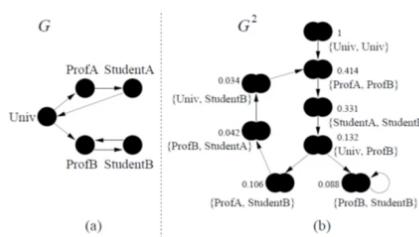
Complessità: $\mathcal{O}(Kn^2d_2)$ dove d_2 è la media di $|I(u)||I(v)|$, K è il numero di iterazioni e solitamente è pari a 5.

IMPORTANTE:

Implementando il SimRank stiamo implementando l'idea che due vertici sono simili se hanno vicini simili nel grafo.

Grazie all'implementazione del SimRank, possiamo usarla per applicare i classici clustering algorithms sui grafi.

Esempio:



SimRank based on random walk

In un grafo fortemente connesso, esiste sicuramente un path tra ogni paio di vertici.
Possiamo definire la **Expected distance** da u a v :

$$d(u, v) = \sum_{t: u \sim v} P[t]l(t) \quad \text{La expected distance da } u \text{ a } v \text{ è esattamente il numero di passi che ci si aspetta che un "random walker" percorra, dove a ogni passo segue un random "out-edge".}$$

- La somma è calcolata su tutti i tour (path) che iniziano da u e terminano a v (non toccano v se non alla fine).
- $l(t)$ è la lunghezza del tour (path).
- $P[t]$ è la probabilità che quel tour (path) venga percorso, ed equivale a:

$$P[t] = \begin{cases} \prod_{i=1}^{k-1} \frac{1}{|O(w_i)|} & \text{if } l(t) > 0 \\ 0 & \text{if } l(t) = 0 \quad \text{Out-neighbors of } w_i \end{cases}$$

Cioè è il prodotto di scelte indipendenti fatte ad ogni nodo del tour(path) in base agli out-going edges.

OSS: Si noti che il caso $u = v$ è un caso speciale della formula. In questo caso: $d(u, v) = 0$.

Expected meeting distance (EMD)

È il numero di passi che ci aspetta siano richiesti da due "random walkers" per incontrarsi (uno iniziando dal nodo u e uno dal nodo v).

Formalmente, la EMD è definito come:

$$m(u, v) = \sum_{t: (u, v) \sim (x, x)} P[t]l(t)$$

- $(u, v) \rightarrow (x, x)$ è una coppia di tour(path) $u \rightarrow x$ e $v \rightarrow x$ della stessa lunghezza.
- $P[t]$ indica la probabilità che questa coppia di tour(path) venga presa, cioè che u prenda il path t_1 e v prenda il path t_2 .
- $l(t)$ è la lunghezza dei due path

La somma è calcolata su tutti i tour (path) che iniziano da (u, v) che toccano un nodo singolo alla fine e solo alla fine.

Expected meeting distance

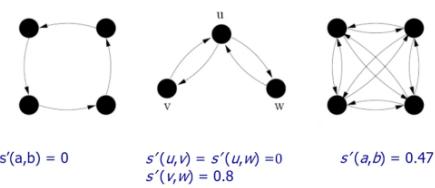
Definiamo adesso, usando una costante C (compresa tra 0 e 1) la expected meeting probability, la quale è la nostra similarity measure based on random walk:

$$s'(u, v) = \sum_{t: (u, v) \sim (x, x)} P[t]C^{l(t)} \quad C \in (0, 1)$$

- Il parametro C specifica la probabilità di continuare a camminare ad ogni step del tour.

OSS:

1. $s'(a, b) = 0 \Rightarrow$ Non ci sono tour da (a, b) a nodi singoli (x, x)
2. $s'(a, b) = 1 \Rightarrow a = b$
3. $s'(a, b) \in [0, 1] \Rightarrow$ per ogni (a, b)



Come possiamo eseguire il clustering direttamente sul grafo senza usare il SimRank?

Intuitivamente, dovremmo tagliare il grafo in pezzi, dove ogni pezzo è un cluster.

Le caratteristiche dei cluster dovrebbero essere:

- 1) I vertici all'interno del cluster sono ben connessi;
- 2) I vertici tra clusters diversi sono connessi in modo molto più debole.

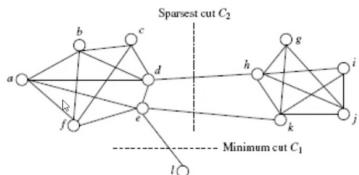
Sia $G = (V, E)$ un grafo diretto:

Taglio $C(S, T)$	Partitionamento del set di vertici V in G , tale che, $V = S \cup T$ e $S \cap T = \emptyset$
Cut set di un taglio	Insieme di archi $\{(u, v) \in E \mid u \in S, v \in T\} \rightarrow$ n° archi che sono stati tagliati;
Dimensione di un taglio (cut size)	Cardinalità del cut set (n° archi tagliati). Se gli archi sono pesati, il valore del taglio è la somma dei pesi.

Quali tagli possiamo utilizzare per ottenere i cluster all'interno del grafo?

Se usassimo il **minimum cut**, cioè il taglio di minima dimensione (che taglia meno archi) che divide il grafo in due parti, non otteniamo il risultato che vorremo.

Otteniamo un risultato migliore utilizzando la misura di sparsità, cioè lo **Sparsest Cut**.



Sparsest cut

L'intuizione è di scegliere un taglio dove, per ogni vertice u che è coinvolto in un arco contenuto nel cut set, la maggior parte degli altri archi che connettono u appartengono allo stesso cluster.

Formalmente:

La sparsità di un taglio $C = (S, T)$ è definita come:

$$\Phi = \frac{\text{cut size}}{\min(|S|, |T|)} \rightarrow \text{Vogliamo un valore basso di sparsità}$$

Un taglio è il più sparso se la sua sparsità non è maggiore rispetto a quella di un qualsiasi altro taglio.

COMPLESSITÀ: NP-HARD, $O(\sqrt{\log(n)})$

SCAN: Density-based clustering of networks

Sfrutta il concetto di densità sui grafi.

Terminologia:

Clique	Gruppo di vertici fortemente interconnessi tra loro
Hubs	Vertice che non appartiene a nessun gruppo ma interconnette gruppi diversi tra loro
Outliers	Vertici che sono lontani dagli altri vertici
Neighborhood di un vertice	Vertici connessi tramite un arco al vertice preso in considerazione. Viene definito con il simbolo $\Gamma(v)$
Structure Similarity	La structure similarity misura la somiglianza del vicinato tra due vertici. $\sigma(v, w) = \frac{ \Gamma(v) \cap \Gamma(w) }{\sqrt{ \Gamma(v) \Gamma(w) }}$ Dove: $\Gamma(u) = \{v \mid (u, v) \in E\} \cup \{u\}$ Structure similarity è grande per i membri di una clique e piccola per hubs e outliers.

SCAN usa un similarity threshold ε per definire l'appartenenza ad un cluster.

Per un vertice $v \in V$, il $\varepsilon - \text{Neighborhood}$ di v , è definito come:

$$N_\varepsilon(v) = \{w \in \Gamma(v) \mid \sigma(v, w) \geq \varepsilon\}$$

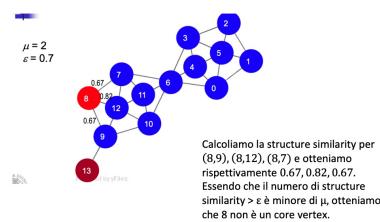
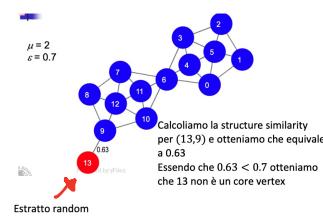
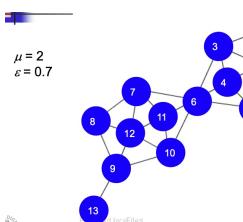
} vertici nel vicinato di v che hanno in comune con v un numero di vicini maggiore della soglia

$$\text{CORE}_{\varepsilon, \mu}(v) \Leftrightarrow |N_\varepsilon(v)| \geq \mu$$

Dove: μ è un popularity threshold

ATTENZIONE: ε e μ sono i parametri dell'algoritmo. Essi influiscono molto sull'output dell'algoritmo.

Esempio



Misurare la qualità di un clustering applicato sui grafi

Siano K i cluster, la qualità del clustering può essere valutata usando la "modularità di un clustering". La modularità di un clustering è definita come:

$$Q = \sum_{i=1}^k \left(\frac{l_i}{|E|} - \left(\frac{d_i}{2|E|} \right)^2 \right)$$

probability edge in cluster i
probability a random edge would fall into cluster i

Dove:

- l_i : Numero di archi nel $i - th$ cluster (tra i vertici del cluster)
- d_i : Somma dei gradi (numero di archi entranti/uscenti) dei vertici nel $i - th$ cluster

La modularità del clustering di un grafo è la differenza tra la probabilità che un arco sia nel cluster i e la probabilità che un arco casuale potrebbe cadere nel cluster i .

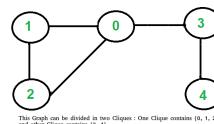
Il clustering ottimale di un grafo massimizza la modularità.

Dificoltà del graph clustering

- Costo computazionale alto
- Grafi "sofficiati" (che includono pesi e/o cicli)
- High dimensionality (Un grafo può avere molti vertici, in una matrice di similarità, un vertice è rappresentato come un vettore (riga della matrice) la quale dimensionalità è il numero di vertici nel grafo)
- Sparsità (Un grafo grande è spesso sparso, il che significa che ogni vertice in media è connesso a un basso numero di altri vertici) -> Una similarity matrix estratta da un grafo grande e sparso, può essere anche essa sparsa.

Graph clustering: Methods

- 1) Clustering methods per high-dimensional data
 - a. Si estraie la similarity matrix dal grafo usando la similarity measure (SimRank)
 - b. Si usa un clustering algorithm per high-dimensional data
- 2) Clustering methods usati direttamente per fare clustering sul grafo (SCAN)
 - a. Si sfruttano le peculiarità del grafo per performare un algoritmo di clustering



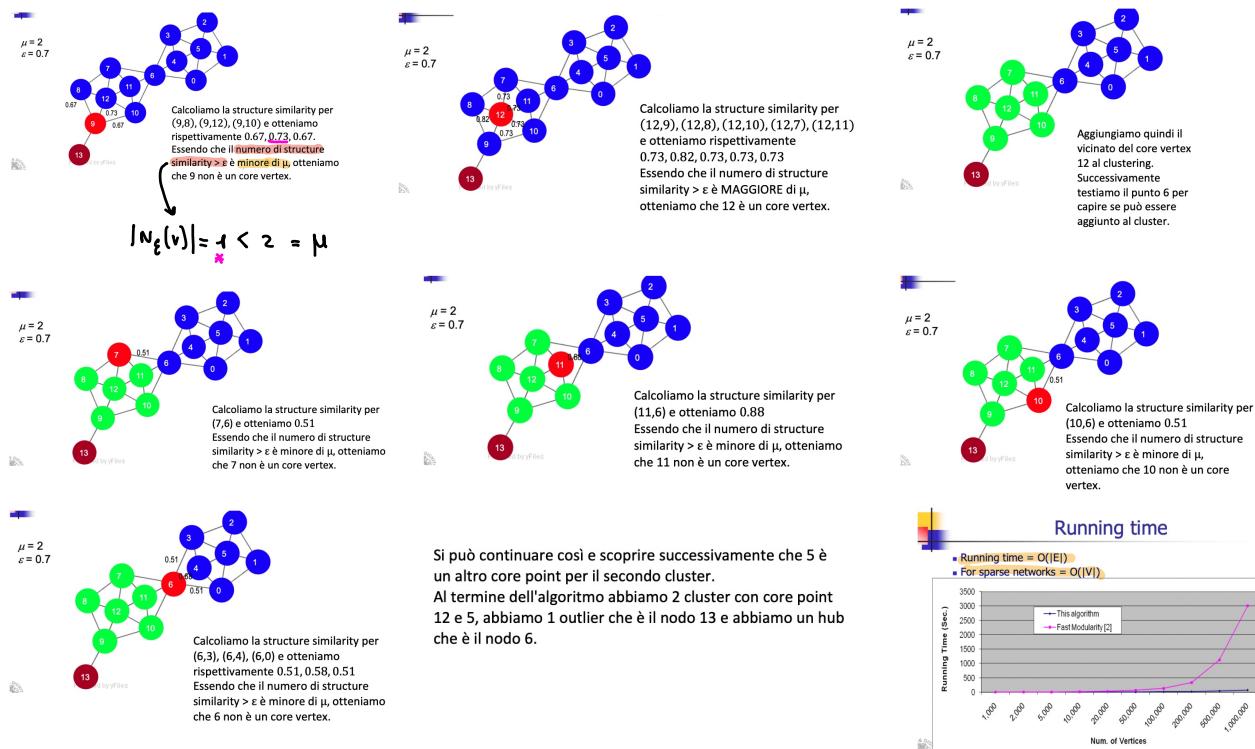
Algoritmo:

SCAN costruisce i cluster partendo dai **core vertices** (similmente al DBSCAN)

- 1) Se un vertice v è nel $\varepsilon - \text{Neighborhood}$ di un core vertex u , allora v è assegnato allo stesso cluster di u
- 2) Questo processo continua fin quando nessun vertice può essere unito a nessun cluster.

Al termine dell'algoritmo abbiamo che:

- 1) Ogni cluster ha le proprietà:
 - a. Connectivity: Per ogni coppia di vertici all'interno del cluster, si dicono connessi se esiste un core vertex u t.c. essi possono essere raggiunti da u
 - b. Maximality: Per ogni coppia di vertici (v, w) in V , se v può essere raggiunto da w vuol dire che v e w appartengono allo stesso cluster.
- 2) Ogni Hub ha le proprietà:
 - a. Non appartiene a nessun cluster
 - b. Fa da ponte tra più clusters
- 3) Ogni outlier ha le proprietà:
 - a. Non appartiene a nessun cluster
 - b. È connesso a un cluster

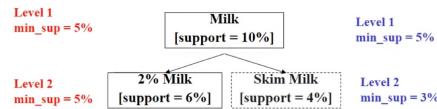


Advanced frequent pattern mining

Mining multiple-level association rules

Gli oggetti spesso formano delle gerarchie, gli items a livello inferiore hanno un supporto più basso. Bisogna usare soglie di min_sup inferiori a livelli inferiori.

uniform support



Usare un supporto più basso ci permette di analizzare oggetti che non sono solitamente molto frequenti, come ad esempio: "diamanti", "orologi", "videocamere" ecc..

Dovuto a questa gerarchia, potremmo avere delle regole che sono ridondanti, come ad esempio:

- milk \Rightarrow wheat bread [support = 8%, confidence = 70%]
- 2% milk \Rightarrow wheat bread [support = 2%, confidence = 72%]

La prima regola è un antenato rispetto alla seconda.

Redundant rule: Una regola è ridondante se il suo supporto e confidenza sono vicini a un expected value, basato sulla regola antenato.

ATTENZIONE: Possiamo evitare di definire le regole più specifiche perché possono essere calcolate direttamente dalla regola principale!

Mining Multi-Dimensional Association rules

Le regole multidimensionali sono composte da 2 o più dimensioni (predicati).

Esse si dividono in due tipologie:

- 1) Inter-dimensional association rules (non ci sono predici replicati)
 $age(X, "19-25") \wedge occupation(X, "student") \Rightarrow buys(X, "coke")$
- 2) Hybrid-dimensional association rules (ci sono predici replicati)
 $age(X, "19-25") \wedge buys(X, "popcorn") \Rightarrow buys(X, "coke")$

Non abbiamo bisogno di introdurre algoritmi di mining aggiuntivi per andare a minare questi tipi di pattern.

Cambiando semplicemente il significato delle nostre association rules.

Mining Quantitative Association rules

Minare regole che sono composte da attributi quantitativi, cioè che impongono un ordine tra i valori.

Per gestire questi attributi dobbiamo usare qualche tipo di discretizzazione, clustering, deviation:

- Statica (Data cube methods) = Basata su concetti di gerarchia predefiniti
- Dinamica (binning, clustering) = Basata sulla distribuzione
- Clustering (Distance-based association) = Applichiamo il clustering 1-dimensionale e poi creiamo le associazioni
- Deviation

Mining Negative correlated and Rare Patterns

Rare patterns	Sono pattern con basso supporto ma interessanti
Negative (correlated) patterns	Sono pattern che indicano la bassa correlazione tra due oggetti. I negative pattern che sono infrequent tendono ad essere più interessanti rispetto a quelli che sono frequenti.

Negative correlated patterns

Definizione 1.
Se due itemset X e Y sono entrambi frequenti ma è raro che siano insieme allora X e Y sono correlati negativamente.

$$sup(X \cup Y) < sup(X) * sup(Y)$$

PROBLEMA: La definizione tramite support non è null-invariant, dipende dal numero di transazioni che non contengono ne X ne Y!

Definizione 2.

Se due itemset X e Y sono fortemente correlati negativamente, allora:

$$sup(X \cup \bar{Y}) \times sup(\bar{X} \cup Y) \gg sup(X \cup Y) \times sup(\bar{X} \cup \bar{Y})$$

PROBLEMA: Anche questa definizione soffre del null-invariant problem!

Definizione 3.

Si usa una misura basata sulla Kulczynski measure. Se due itemset X e Y sono frequenti, ma

$$\frac{P(X|Y) + P(Y|X)}{2} < \varepsilon, \text{ dove } \varepsilon \text{ è una soglia per i negative patterns}$$

Allora X e Y sono correlati negativamente.

Con questa definizione non abbiamo il problema del null-invariant, infatti se per esempio:

Ex. For the same needle package problem, when no matter there are 200 or 10^5 transactions, if $min_sup = 0.01\%$ and $\varepsilon = 0.02$, we have

$$sup(A) = sup(B) = 100/200 = 0.5 > 0.01\%$$

$$(P(B|A) + P(A|B))/2 = (0.01 + 0.01)/2 < 0.02$$

$$sup(A) = sup(B) = 100/10^6 = 0.01\% \geq 0.01\%$$

$$(P(B|A) + P(A|B))/2 = (0.01 + 0.01)/2 < 0.02$$

Constraint-based Mining

E' possibile far trovare tutti i pattern in un database autonomamente? No, non è realistico.

Abbiamo bisogno che l'utente diriga cosa deve essere minato usando un data mining query language oppure un'interfaccia grafica.

Dovremo definire dei vincoli che sono utili per ridurre il costo computazionale.

- 1) User Flexibility: Vincoli su cosa deve essere mined
- 2) Optimization: Esplora i vincoli per rendere il mining efficiente.

Tipi di vincoli:

- 1) Knowledge type constraint: Classification, association, etc.
- 2) Data constraint: Si usano le query SQL-like per imporre dei vincoli alla ricerca (es: trovare i prodotti venduti in un certo stato)
- 3) Dimension/level constraint: In relazione alla regione, al prezzo, al marchio, alla categoria del cliente
- 4) Rule (or pattern) constraint: Ad esempio che l'acquisto di un prodotto poco costoso fa comprare un prodotto molto costoso
- 5) Interestingness constraint: rules forti ($\min_{\text{supp}} \geq 3\%$, $\min_{\text{confidence}} \geq 60\%$)

Pattern space pruning con anti-monotonicity constraint

Un constraint C è anti-monotone se il super pattern soddisfa C, tutti i suoi sub-pattern soddisfano C.

Di conseguenza, se un itemset S viola il constraint, allora lo violano anche i suoi supersets.

Gi permette di non generare pattern che non soddisfano i constraint.

Esempi di vincoli anti-monotone:

Anti-monotone perché se un pattern non soddisfa questo vincolo, aggiungendo un item otengo che il super-pattern non soddisfa il vincolo.

- 1) $\sum(I, \text{price}) \leq v$
- 2) $\text{range}(I, \text{profit}) \leq 15$
- 3) Support count

Esempi di vincoli che non sono anti-monotone:

- 1) $\sum(I, \text{price}) \geq v$

Pattern space pruning con monotonicity constraint

Un constraint C è monotone se il pattern soddisfa C, non abbiamo bisogno di controllare C nel subsequent mining.

Di conseguenza, se un itemset S soddisfa il constraint, allora lo soddisfano anche i suoi supersets.

Esempi di vincoli anti-monotone:

- 1) $\sum(I, \text{price}) \geq v$
- 2) $\min(I, \text{price}) \leq v$
- 3) $\text{range}(I, \text{profit}) \geq 15$

Pattern space pruning con succinctness constraint

Possiamo enumerare solo e soltanto i set che è garantito che soddisfino i constraints. Se un rule constraint è succinct, possiamo direttamente generare i sets che lo soddisfano, anche prima del support counting.

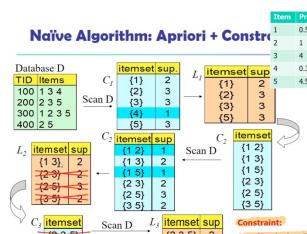
Questo permette di evitare overhead sul paradigma di generazione e test dei pattern.

Esempio di vincolo succinctness: $\min(I, \text{price}) \geq v$

(Possiamo generare esplicitamente e precisamente tutti gli itemset che soddisfano il constraint)

Esempio di vincolo non succinctness: $\sum(I, \text{price}) \geq v$

Esempio di Apriori con constraints



Handling multiple constraints

Constraint differenti possono avere richieste differenti o item-ordering differenti.

Se abbiamo un conflitto sull'ordinamento dei dati:

- 1) Provare a soddisfare prima un constraint
- 2) Usare l'ordine per l'altro constraint per provare a minare frequent itemset in corrispondenza del projected database

Quali constraints sono convertibili?

Constraint	Convertible anti-monotone	Convertible monotone	Strongly convertible
$\text{avg}(S) \leq v \geq v$	Yes	Yes	Yes
$\text{median}(S) \leq v \geq v$	Yes	Yes	Yes
$\text{sum}(S) \leq v$ (items could be of any value, $v \geq 0$)	Yes	No	No
$\text{sum}(S) \leq v$ (items could be of any value, $v \leq 0$)	No	Yes	No
$\text{sum}(S) \geq v$ (items could be of any value, $v \geq 0$)	No	Yes	No
$\text{sum}(S) \geq v$ (items could be of any value, $v \leq 0$)	Yes	No	No
.....			

Quando vogliamo applicare i constraint esploriamo le Metarule

Una metarule forma un'ipotesi per quanto riguarda la relazione che un user è interessato nel sondare o confermare. Metarule può essere nella regola con predici e vincoli parzialmente inizializzati.

$$P_1(X, Y) \wedge P_2(X, W) \Rightarrow \text{buys}(X, \text{"iPad"})$$

La regola risultato può essere:

$$\text{age}(X, "15-25") \wedge \text{profession}(X, \text{"student"}) \Rightarrow \text{buys}(X, \text{"iPad"})$$

Come usare i constraint per fare il prune dello spazio di ricerca

- 1) Pruning pattern search space: L'utente verifica i pattern candidati e decide se alcuni pattern possono essere pruned.

Tipi di constraints:
 a. Anti-monotonic: Se il constraint viene violato, la procedura di mining di quel pattern può essere terminata.
 b. Monotonic: Se il constraint c è soddisfatto, non abbiamo bisogno di verificare di nuovamente.
 c. Succinct: Il constraint c deve essere soddisfatto, così uno può iniziare con il dataset che soddisfa c.
 d. Convertible: Il constraint c non è monotonic né anti-monotonic, ma può essere convertito in monotonic/anti-monotonic se gli items nella transazione possono essere ordinati.

- 2) Pruning data search space: Verifica il dataset per determinare se un pezzo particolare dei dati può essere disponibile nel contribuire nella generazione di pattern soddisfacenti (per un particolare pattern) nel rimanente mining process.

Tipi di constraints:
 a. Data succinct: Il data spaces può essere pruned al pattern di inizio del data mining process.
 b. Data anti-monotonic: Se una transazione t non soddisfa c, t può essere pruned dai suoi mining successivi.

Data Space Pruning con Data anti-monotonicity

Un constraint c è data anti-monotone se per un pattern p che non può essere soddisfatto da una transazione t under c, anche p's superset non possono essere soddisfatti da t under c.

La chiave per il data anti-monotone è la riduzione dei dati ricorsiva (ridurre il numero di transazione da verificare).

Esempi di constraint data anti-monotone:

- 1) $\sum(I, \text{price}) \geq v$
- 2) $\min(I, \text{price}) \leq v$
- 3) $\text{range}(I, \text{profit}) \geq 25$

Attenzione: Questo tipo di pruning non può essere fatto all'inizio del mining ma devo eseguirlo durante l'esecuzione dell'algoritmo.

Pattern space pruning con convertible constraints

Convertire constraint difficili in anti-monotone o monotone ordinando correttamente gli items.

Esempio:

Sia C = $\text{avg}(S, \text{profit}) \geq 25$

- 1) Ordinare gli items in ordine decrescente di valore.
 $< a, f, g, d, b, h, c, e >$
- 2) Se un itemset afb viola C.
 - a. Allora anche afbh lo viola $\rightarrow afb'$
 - b. Diventa anti-monotone

Un constraint è strongly convertible quando esso è convertibile

Sia con ordine crescente che decrescente dei dati.

$\text{avg}(X) > 25$ è trasformabile in anti-monotone con ordine:

$< a, f, g, d, b, h, c, e >$

$\text{avg}(X) > 25$ è trasformabile in monotone con ordine:

$< e, c, h, b, d, g, f, a >$

Attenzione: I convertibile constraint non possono essere usati nell'apriori algorithm, ma possono essere usati nel frequent-pattern growth.

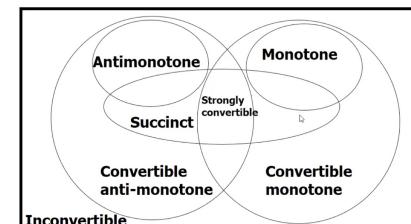
Item	Profit
a	40
b	0
c	-20
d	10
e	-30
f	30
g	20
h	-10

TID	Transaction
10	a, f, d, b, c
20	f, g, d, b, c
30	a, f, d, c, e
40	f, g, h, c, e

Item	Profit
a	40
b	0
c	-20
d	10
e	-30
f	30
g	20
h	-10

TID	Transaction
10	a, f, d, b, c
20	f, g, d, b, c
30	a, f, d, c, e
40	f, g, h, c, e

Classificazione dei constraints



- K è un parametro che indica il numero massimo di patterns to be mined.
- T è un parametro che indica il raggio della bounding ball.
- Se il pool è composto da più di K super-patterns, ne evengono estratti K a random.
- L'algoritmo terminerà perché andando avanti il pool sarà sempre più piccolo perché gli itemset avranno man mano un raggio sempre più piccolo!

Mining compressed Patterns

Pattern X per cui non esiste un super pattern Y che ha lo stesso supporto di X

Quando ritorniamo i pattern frequenti, se reportiamo i closed frequent pattern, essi possono essere grandi quantità e rischiamo che non siano significativi.

Definiamo la Pattern distance measure e il δ -clustering.

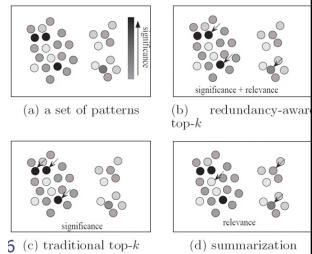
Pattern distance measure

$$D(P_1, P_2) = 1 - \frac{|T(P_1) \cap T(P_2)|}{|T(P_1) \cup T(P_2)|} \quad 0 \leq D \leq 1$$

δ -clustering

Per ogni pattern P , trovare tutti i pattern S tali che possono essere espressi da P e le loro distanze da P sono $\leq \delta$. Tutti i pattern nel cluster possono essere rappresentati da P .

Usando il δ -clustering ci permette di andare a definire un sottoinsieme di pattern frequenti che li vada a rappresentare con una minima perdita d'informazione (data dalla pattern distance).



Redundancy-Award Top-K Patterns

In molti casi, i frequent pattern non sono mutualmente indipendenti. Di conseguenza potrebbero portare le stesse informazioni.

Per risolvere il problema, i desired patterns sono così composti: **high significance e low redundancy**.

Significance measure (S): E' una funzione che mappa un pattern $p \in P$ a un valore reale t.c. $S(p)$ è il grado di utilità del pattern p .

Redundancy R tra due pattern p e q è definita come: $R(p, q) = S(p) + S(q) - S(p, q)$

La redundancy measure ideale $R(p, q)$ è solitamente difficile da ottenere. Tuttavia, possiamo approssimare la redundancy usando la distanza tra patterns. Il problema di trovare i redundancy-aware top-k patterns può così essere trasformato nel trovare l'insieme di k-pattern che maximizza la marginal significance.

Sequential Pattern Mining

Qual è la differenza tra il frequent pattern mining e il sequential pattern mining?

Nel frequent pattern mining non teniamo di conto della sequenza delle transazioni e il loro ordinamento.

Nel sequential pattern mining vogliamo tenere di conto di questo fattore **sequenziale** (se un cliente acquista un certo oggetto, se la seconda volta acquista un oggetto specifico ecc...).

Il sequential pattern mining si occupa di scoprire sottosequenze frequenti in un database di sequenze.

Un sequence database salva un numero di record, i quali sono sequenze ordinate di eventi, con o senza notazione discreta del tempo.

I record vengono salvati nel seguente modo: [Transazione/Customer_id, < Sequenza ordinata di eventi >]

Le sequenze nel nostro caso sono sequenze di itemset.

Definizione del problema

Dati:

- Un insieme di record sequenziali (sequenze) che rappresentano un sequential database D
- Una soglia minimum support chiamata min_supp
- Un insieme di K item uniti $I = \{i_1, \dots, i_k\}$

Vogliamo trovare l'insieme di tutte le frequent sequences S nel sequence database D che rispettano il min_supp .

Terminologia:

Un itemset è un insieme di items in I , denotati (i_1, i_2, \dots, i_k) , dove i_j è un item o un evento.

GLI ITEMSET SONO DENOTATI DALLE PARENTESI SE COMPOSTI DA + DI UN ELEMENTO!

Una sequenza S è denotata come una sequenza di elementi $< e_1, e_2, \dots, e_q >$, dove l'elemento e_j è un una lista ordinata in modo lessicografico di items. Una sequenza di K elementi è chiamata k -sequence.

Una sequenza $a = < e_1 e_2 \dots e_m >$ è una sottosequenza di un'altra sequenza $b = < e_1 e_2 \dots e_n >$ se $i_1 < \dots < i_m$ e tutti gli eventi $e_{i_j} \in a$ e $e_i \in b$ e $i_1 < i_2 < \dots < i_m < n$, tale che e_{i_j} contenuto in e_i .

Dati due itemset t e t' , $t < t'$ (t è lessicograficamente minore di t') sse, dato un numero intero $0 \leq h \leq \min\{k, l\}$, una delle seguenti proprietà è vera:

- 1) $i_r = j_r$ per $r < h$ e $i_h < j_h$
- 2) $k < l$, e $i_1 = j_1, i_2 = j_2, \dots, i_k = j_k$

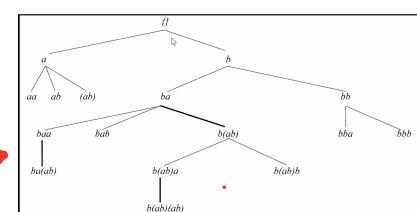
Un sequential pattern è massimale se non è una sottosequenza di nessun altro sequential pattern.

OSS: Il problema da risolvere è molto complesso, perché abbiamo un grande numero di sequenze che dobbiamo esplorare!

Lexicographic order (come si può notare il numero di sequenze che dobbiamo esplorare aumenta esponenzialmente)

A sequence database	
Seq. ID	Sequence
10	<u>$<(bd)cb(ac)>$</u>
20	$<(bf)(ce)b(fg)>$
30	$<(ah)(bf)abf>$
40	$<(be)(ce)d>$
50	$<a(bd)bcb(ade)>$

A sequence: $<(bd) c b (ac)>$
Elements
 $<ad(ae)>$ is a subsequence of $<a(bd)bcb(ade)>$
Given support threshold $min_sup = 2$, $<(bd)cb>$ is a sequential pattern



Support of a sequence [0.1]

Il supporto di una sequenza S , denotato come $\sigma(S)$, è il numero totale di sequenze nelle quali S è una sottosequenza, diviso il numero totale di sequenze nel database D .

Una sequenza è detta "frequente" se la sua frequenza non è minore di una certa soglia specificata dall'utente, chiamata $minimum support$, denotata come min_supp .

Frequent closed sequence

Una sequenza frequente S_a è chiamata frequent closed sequence se non esiste una super-sequenza di S_a con lo stesso supporto.

Cioè non esiste S_b t.c. $S_a \leq S_b$ e $\sigma(S_a) = \sigma(S_b)$.

Altrimenti si dice che la sequenza S_a è assorbita da S_b .

OSS:

- Se $\sigma(S_b) \geq \sigma(S_a) \Rightarrow S_a$ è una frequent closed sequence
- Non può succedere che $\sigma(S_a) < \sigma(S_b)$ perché S_b è una super-sequenza di S_a

In output solitamente vogliamo le frequent sequence che sono anche maximal!

ESEMPIO

Customer ID	Transaction Time	Items Bought	Customer ID	Customer Sequence
1	June 25 '93	30	1	{(30) (90)}
1	June 30 '93	90	2	{(10 20) (30) (40 60 70)}
2	June 10 '93	10,20	3	{(30) (50 70)}
2	June 15 '93	30	4	{(30) (40 70) (80)}
2	June 20 '93	40,60,70	5	{(90)}
3	June 25 '93	30,50,70		
4	June 25 '93	30		
4	June 30 '93	40,70		
4	July 25 '93	90		
5	June 12 '93	90		

Note: Use Minsup of 25%
{ (10 20) (30) } Does not have minsup (Only supported by Cust. 2)
{ (30) }, { (70) }, { (30) (40) } ... are not maximal.

ALGORITMI

Gli algoritmi che si approcciano nel sequential pattern mining sono composti solitamente da queste 5 fasi:

- 1) Sort Phase
- 2) Itemset Phase
- 3) Transformation Phase
- 4) Sequence Phase
- 5) Maximal Phase

Sort Phase

Sortiamo il dataset tenendo in considerazione il customer_id (come "chiave principale") e il timestamp (come "chiave secondaria").

Customer ID	Transaction Time	Items Bought
1	June 25 '93	30
1	June 30 '93	90
2	June 10 '93	10,20
2	June 15 '93	30
2	June 20 '93	40,60,70
3	June 25 '93	30,50,70
4	June 25 '93	30
4	June 30 '93	40,70
4	July 25 '93	90
5	June 12 '93	90

to e zo ad esempio non li prendiamo in considerazione perché compaiono solo una volta.

Transformation Phase

Convertire il database delle transazioni nel sequence database.

Customer ID	Original Customer Sequence	Sequenza senza item non frequenti	mapping
		Transformed Customer Sequence	
1	{ (30) (90) }	<{(30)} {(90)}>	<{1} {5}>
2	{ (10 20) (30) (40 60 70) }	<{(30)} {(40),(70),(40 70)}>	<{1} {2,3,4}>
3	{ (30) (50) (70) }	<{(30)} {(70)}>	<{1,3}>
4	{ (30) (40 70) (90) }	<{(30)} {(40),(70),(40 70)} {(90)}>	<{1} {2,3,4} {5}>
5	{ (90) }	<{(90)}>	<{5}>

Itemset Phase

IDEA: Se non abbiamo frequent itemset non abbiamo frequent sequences.

- 1) Contiamo il supporto e troviamo gli itemset frequenti. Considerare solo gli itemset frequenti permette di ridurre la complessità del nostro problema.
- 2) Ogni itemset frequente viene mappato con un insieme di interi contigui. Questa mappatura viene utilizzata per velocizzare la computazione.

Large Itemsets	Mapped To
(30)	1
(40)	2
(70)	3
(40 70)	4
(90)	5

Sequence Phase

- 1) Si genera l'insieme delle sequences candidate
- 2) Si conta il supporto per ogni sequenza candidata
- 3) Si eliminano le sequenze candidate che sono sotto la soglia del min_supp .

Gi sono due famiglie di algoritmi:

- a. **AprioriAll**
- b. **Count-some: C**erca di evitare di contare le sequenze non massimali contando le sequenze più lunghe per prime.
 - a. **AprioriSome**
 - b. **DynamicSome**

Maximal Phase

Qualsiasi sia l'algoritmo che viene utilizzato, per trovare le maximal sequences dobbiamo applicare il seguente procedimento:

Sia S l'insieme delle sequenze che sono frequenti

for ($k=1$; $k>1$; $k--$) do
 foreach k -sequence s_k do
 delete from S all subsequences of s_k

V sequenza s_k elimina le sub-sequences di s_k da S

Esistono delle strutture ed algoritmi per fare questo procedimento efficientemente (hash trees).

APRIORI_ALL (Count-all)

Generazione dei candidati C_k

- 1) Join due sequenze in L_{k-1} per generare C_k

- a. Per ogni sequenza in L_{k-1} che ha lo stesso 1st uguale a $k - 2$ th itemsets, seleziona da 1 a $k - 1$ itemset dalla prima sequenza, e joina con l'ultimo itemset da un'altra sequenza

Example

$$\begin{aligned} L_3 &= \{123\} \{234\} \{124\} \{134\} \{135\} \\ C_4 &= \{1\ 2\ 3\ 4\} \{1\ 3\ 4\ 5\} \{1\ 2\ 4\ 3\} \end{aligned}$$

Essendo che l'ordine conta, devo generare anche questi itemsets.

- 2) Elimina tutte le sequenze in C_k se alcune delle loro sotto-sequenze non sono in L_{k-1}

Example: $C_4 = \{1\ 2\ 3\ 4\}$

- $\{1, 3, 4, 5\}$ lo rimuovo perché $\{3, 4, 5\}$ non in L_3
- $\{1, 3, 5, 6\}$ lo rimuovo perché $\{3, 5, 6\}$ non in L_3
- $\{1, 2, 4, 5\}$ lo rimuovo perché $\{2, 4, 5\}$ non in L_3

```

L1 = {large 1-sequences}
for (k = 2; Lk-1 ≠ {}; k++) do
  begin
    Ck = New candidates generated from Lk-1
    foreach customer-sequence c in the database do
      Increment the count of all candidates in Ck
      that are contained in c.
    Lk = Candidates in Ck with minimum support.
  end
  Answer = Maximal Sequences in UkLk
  
```

Notation:

L_k : Set of all large k -sequences
 C_k : Set of candidate k -sequences

ESEMPIO APRIORI_ALL

Customer Sequences		L1		L2		L3	
		1-Seq	Support	2-Seq	Support	3-Seq	Support
<{1 5}>	{2} {3} {4}>	<1>	4	<1 2>	2	<1 2 3>	2
<{1}>	{3} {4}>	<2>	2	<1 3>	4	<1 2 4>	2
<{1}>	{2} {3} {4}>	<3>	4	<1 4>	3	<1 3 4>	3
<{1}>	{3} {5}>	<4>	4	<1 5>	3	<1 3 5>	2
<{4}>	{5}>	<5>	4	<2 3>	2	<2 3 4>	2
				<2 4>	2	<3 4 5>	1
				<3 4>	3		
				<3 5>	2		
				<4 5>	2		
				<2 5>	0		

L4	
4-Seq	Support
<1 2 3 4>	2
<2 5>	0

Min_supp = 25%

PROBLEMA APRIORI_ALL: Generazione dei candidati, quando applichiamo l'algoritmo generiamo un gran numero di candidati che poi dobbiamo verificare.

APRIORI_SOME

Cerca di evitare il counting delle sequenze non-maximal contando per prime le sequenze più lunghe.

FASI:

- 1) Forward phase = Trovare tutte le sequenze frequenti di una certa lunghezza
- 2) Backward phase = Trovare tutte le rimanenti sequenze frequenti (per esempio potremmo contare le sequenze lunghe 1,2,4,6 nella forward phase e 3,5 nella backward phase).

La lunghezza da contare al passo K è definita dalla funzione next(). In APRIORI-ALL questa funzione è definita nel seguente modo: $next(k) = k + 1$. Possiamo modificare il "+1" con interi più grandi per implementare l'APRIORI-SOME.

Terminato l'algoritmo ottengo come risultato (Maximal sequences):
 $\langle 1 2 3 4 \rangle, \langle 1 3 5 \rangle, \langle 4 5 \rangle$

Questo perché prendo la sequenza da 4 elementi, cioè $\langle 1 2 3 4 \rangle$ e verifico quali sequenze da 3 elementi non sono contenute al suo interno, e ottengo solo $\langle 1 3 5 \rangle$. Successivamente faccio la stessa cosa con $\langle 3 4 5 \rangle$ verificando quali sequenze da 2 elementi non sono contenute al suo interno, ed ottengo $\langle 4 5 \rangle$.

```
function next(k: integer)
begin
  if (hit_k < 0.666) return k + 1;
  elseif (hit_k < 0.75) return k + 2;
  elseif (hit_k < 0.80) return k + 3;
  elseif (hit_k < 0.85) return k + 4;
  else return k + 5;
end
```

Dove: $hit_k = \frac{|L_k|}{|C_k|}$

cardinalità L_k
 cardinalità C_k

Intuizione: Come hit_k aumenta, il tempo perso nel contare le estensioni di piccoli candidati diminuisce.

Algoritmo (Forward Phase)

```
L1 = {large 1-sequences}
C1 = L1
last = 1
for (k = 2; Ck-1 ≠ {} and Llast ≠ {} ; k++) do
  begin
    if (Lk-1 known) then
      Ck = New candidates generated from Lk-1
    else
      Ck = New candidates generated from Ck-1
    if (k==next(last)) then begin // (next k to count?)
      foreach customer-sequence c in the database do
        Increment the count of all candidates in Ck
        that are contained in c.
      Lk = Candidates in Ck with minimum support.
      last = k;
    end
  end
end
```

28

Algoritmo (Backward Phase)

```
for (k--; k>=1; k--) do
① if (Lk not found in forward phase) then begin
  Delete all sequences in Ck contained in
  some Li i>k;
  foreach customer-sequence c in Dt do
    Increment the count of all candidates in Ck
    that are contained in c
  Lk = Candidates in Ck with minimum support
end
② else // lk already known
  Delete all sequences in Ck contained in
  some Li i>k;
Answer = UkLk // (Maximal Phase not Needed)
```

Notation: D_t : Transformed database

ESEMPIO

Forward Phase:		next(k) = 2k		minsup = 2	
L_1	1-Sequences	L_2	2-Sequences	C_3	L_4
(1)	4	(1 2)	2	(1 2 3)	(1 2 3 4)
(2)	2	(1 3)	4	(1 2 4)	(1 2 3 5)
(3)	4	(1 4)	3	(1 2 5)	(1 3 4)
(4)	4	(1 5)	3	(1 3 5)	(1 3 4 5)
(5)	4	(2 3)	2	(2 3 4)	(2 3 4 5)
		(2 4)	2	(3 4 5)	(3 4 5)
		(3 4)	3		
		(3 5)	2		
		(4 5)	2		

Figure 8: Customer Sequences

Backward Phase: $next(k) = 2k$
 minsup = 2

Figure 8: Customer Sequences

Sarà il counting del sequenze lunghe 3, le genera e basta.

① Rimuove le sottosequenze che sono in $\{1 2 3 4\}$ o $\{1 3 5\}$,
 rimuove le sequenze che sono sottosequenze di $\{1 2 3 4\}$,
 conta il supporto per le rimanenti e rimuove quelle che non
 hanno il supp \geq min-sup.

Attenzione: Generiamo più candidate sequences rispetto all'APRIORI-ALL perché per il passo K generiamo le sequenze usando K-1 candidati e questo numero di candidati è maggiore delle frequent sequences.
 Qual è il vantaggio? Con la backward phase rimuoviamo dei candidati che non sono massimali.

APRIORI DYNAMIC SOME

Come l'AprioriSome, salta il contare le candidate sequences di una certa lunghezza nella forward phase.
Le candidate sequences che vengono contate sono determinate dal "variable step".

- Inizializzazione:** Tutte le candidate sequences di lunghezza fino a e includendo step sono contate.
ES: Se step = 3 -> Vengono contate le sequence di lunghezza 1,2,3.
- Forward phase:** Tutte le sequence le quali lunghezze sono multiple di step sono contate.
ES: Se step = 3 -> Vengono contate le sequence di lunghezza 6,9,12, ... Possiamo generare le sequence lunghe 6 con le sequence lunghe 3 ecc..
- Backward phase:** Si contano le sequence per le lunghezze che sono state saltate durante la forward phase. (la differenza rispetto all'aprioriSome è che queste candidate sequences non erano state generate nella forward phase).
ES: Se contiamo L_1, L_2 , e L_3 , generiamo nella intermediate phase phase C_7, C_8 e successivamente contiamo C_8 seguita da C_7 dopo aver eliminato le non-maximal sequences (Backward phase).
Questo processo viene poi ripetuto per C_4 e C_5 .

OSS: La Join condition necessita di essere cambiata

per fare il join di sequence che hanno una ugualanza nei primi $k-j$ termini.

Initialization:

Example		L_1		L_2	
1-Sequences	Support	2-Sequences	Supp-qt		
(1)	4	(1 2)	2		
(2)	2	(1 3)	4		
(3)	4	(1 4)	3		
(4)	4	(1 5)	3		
(5)	4	(2 3)	2		
		(2 4)	2		
		(3 4)	3		
		(3 5)	2		
		(4 5)	2		

Forward phase (step=2): $C_4: \{1\ 2\ 3\ 4\}$ $L_4: \{1\ 2\ 3\ 4\}$ $C_6(\cdot)$

Intermediate phase: $C_5: \{1\ 2\ 3\ 4\}$ $C_6(\cdot)$

Backward phase: we count only C_3

Performance degli algoritmi APRIORI

Apriori_some non porta vantaggi perché:

- Genera più candidati
- I candidati rimangono in memoria anche se vengono skippati.

Apriori Dynamic some lavora meglio.

In ogni caso, il problema di tutti gli algoritmi APRIORI è il fatto di:

- Dovere generare molti candidati
- Fare molte scan del database
- Incontra difficoltà quando si cercano di minare long sequential patterns.

g e h rimossi poiché non frequenti

1 1

f_list: b:5, c:4, a:3, d:3, e:3, f:2

Sequence Database SDB

< (bd) c b (ac) >
< (bf) (ce) b (fg) >
< (ah) (bf) a b f >
< (be) (ce) d >
< a (bd) b c b (ade) >

All seq. pat. can be divided into 6 subsets:
 • Seq. pat. containing item *f*
 • Those containing *e* but no *f*
 • Those containing *d* but no *e* nor *f*
 • Those containing *a* but no *d*, *e* or *f*
 • Those containing *c* but no *a*, *d*, *e* or *f*
 • Those containing only item *b*

2

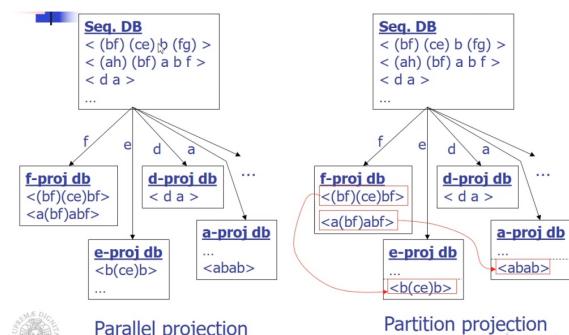
Example: <(ah)(bf)abf> is projected to *f*projected database as <a(bf)abf>, and to *a*-projected database as <abab>, and to *b*-projected database as <bb>

3

FreeSpan: FP-growth per il sequential pattern mining

Vogliamo proiettare il sequence database in un projected sequence database.

- 1) Troviamo gli items frequenti nel db e li listiamo in ordine decrescente del supporto (la lista viene chiamata *f_list*)
- 2) Tutti i sequential patterns possono essere divisi in diversi subset senza sovrapposizioni.
- 3) Le sequenze vengono proiettate come s_i nel i - projected database se c'è almeno un item *i* in s . s_i è la copia di s rimuovendo da s tutti gli item non frequenti e qualsiasi item *j* frequente che segue *i* nella *f_list* (che ha supporto minore).



Tipi di proiezione

1) Parallel Projection:

- Scansionare il db una volta, creando tutti i projected dbs
- Potrebbe produrre molti e relativamente grandi projected dbs se la sequenza in media contiene molti frequent items.

Supponiamo che una frequenza contenga in media l items frequenti. Una transizione è proiettata in $l-1$ projected dbs. La grandezza totale dei dati proiettati è $1 + 2 + \dots + (l-1) = \frac{l(l-1)}{2}$.

Questo implica che la dimensione totale di un singolo item sui projected dbs è $\frac{l(l-1)}{2}$ volte più grande rispetto al db originale.

Ottieniamo quindi che abbiamo molto overhead sull'occupazione di memoria.
Per risolvere questo problema usiamo il Partition Projection method.

2) Partition Projection

- Proiettare la sequenza al projected database dell'ultimo item frequente della sequenza
- Quando viene scansionato il db per essere proiettato, una transizione T è proiettata nel a_i - projected db solo se a_i è un item frequente in T e non c'è nessun altro item dopo a_i nella lista degli item frequenti che appare nella transazione.

Essendo che una transazione è proiettata solo in un projected db, il database dopo la scan è partizionato da proiezioni in un insieme di projected dbs, e questo è chiamato partition projection.

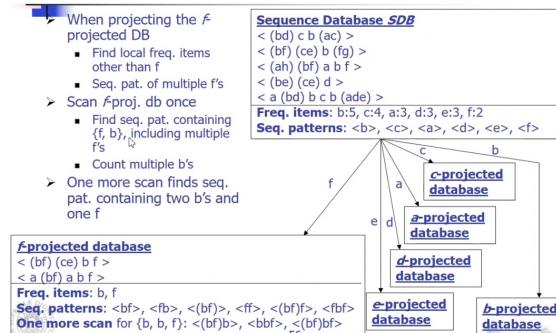
Ogni volta che un projected db viene processato, per assicurare che i rimanenti projected dbs ottengono l'informazione completa, ogni transazione in esso è proiettata verso l' a_j - projected db, dove a_j è l'item nella transazione tale che non ci sono altri item dopo a_j nella lista dei frequent items che appaiono nella transazione.

1

Minare i sequential patterns

I sequential patterns vengono minati nel dbs proiettati (evitiamo di scansionare il db originale più volte)

- When projecting the f projected DB
 - Find local freq. items other than f
 - Seq. pat. of multiple f 's
 - Scan f_{proj} , db once
 - Find seq. pat. containing $\{f, b\}$, including multiple f 's
 - Count multiple b 's
 - One more scan finds seq. pat. containing two b 's and one f

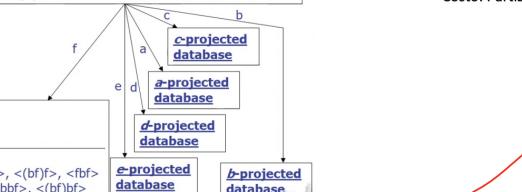


Algoritmo

- 1) Scansionare il db originale 1 volta, e trovare i frequent items e generare f-list
 - 2) Recurivamente, fare la db projection livello per livello

Vantaggi

- Necessità di trovare solo i frequent items per ogni projected db, invece di generare ed esplorare le sequenze candidate
 - Il numero di combinazioni è minore rispetto a tutte le possibili combinazioni
 - Lavora bene in db sparsi
 - Costo: Partizionamento e proiezione del db



Mining by Alternative Level Projected Database

Migliorare l'algoritmo Free-span.

Fragment Item Matrix

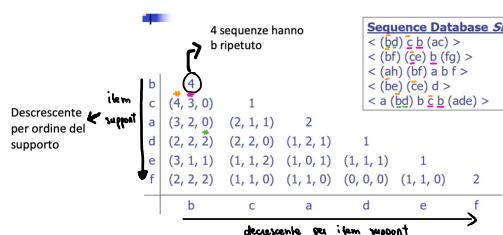
E' una matrice triangolare $F[j, k]$, dove $1 \leq j \leq m$ e $1 \leq k \leq j$, dove m è il numero di frequent items.
 Essa rappresenta i frequent items.

$F[j, j]$ ha un solo contatore che registra se abbiamo la ripetizione di questo item nella sequenza $\langle j, i \rangle$

- $F[j, k]$ ha 3 contatori (A,B,C) dove:

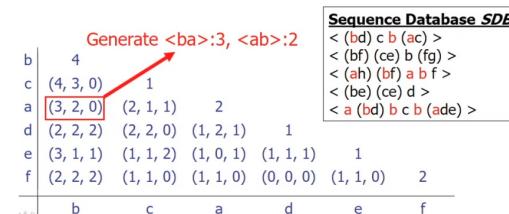
 - A: Numero di occorrenze dove k occorre dopo j ($< jk >$)
 - B: Numero di occorrenze dove k occorre prima di j ($< kj >$)
 - C: Numero di occorrenze dove j occorre consecutivamente con k ($< (jk) >$)

ESEMPIO:



Questa matrice viene utilizzata per dare un'idea di quale candidate sequences devono essere generate. Infatti, essa è usata per generare i sequential patterns di lunghezza 2 e un insieme di projected obs, i quali poi vengono utilizzati per generare le sequence lunghe 3 ecc.

Generazione dei length-2 sequential patterns



Set of annotations: Indica quale insieme di items o sequenze dovrebbero essere esaminate nella

proiezione e nel min

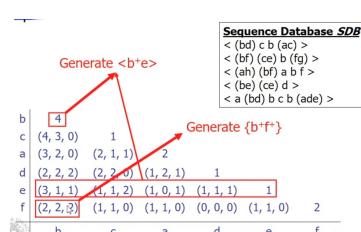
- Tipi di annotazioni:**

 - 1) Annotations of item-repeating patterns
 - 2) Annotations of projected databases

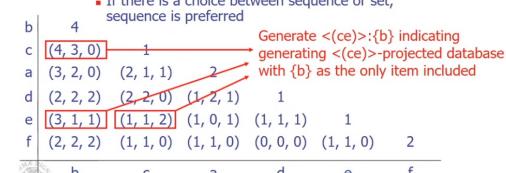
4) Convenzione delle coordinate sui items con attivazione Patterns

5

- If $f[i, j] = \text{min_sup}$, generate $\langle i | j \rangle$
 - The count of $\langle i | j \rangle, \langle i | j | k \rangle, \dots$ should be registered in the next round
 - For a column $i > j$,
 - if $f[i, j] = \text{min_sup}$, generate $i +$
 - There are potentially more than one i appearing in the sequential pattern
 - If $f[j, j] = \text{min_sup}$, generate $j +$
 - If only one of the three counters of $f[i, j]$ is frequent, sequence is used as the annotation; Otherwise, set is used.
 - This distinction is used to enhance string filtering: annotation $\langle b \rangle$ indicates there is no chance for the subsequences $\langle c \rangle b \rangle$ to



- For row j
 - For each $i < j$, if $F[i, j]$, $F[k, j]$ and $F[i, k](k < i)$ may form a pattern generating triple (i.e., all the corresponding pairs are frequent), k should be added to i 's projected column set
 - If there is a choice between sequence or set,



Generazione dei length-2 Patterns e annotations

Item	Length-2 seq. pat.	Ann. on rep. Items	Ann. on proj. DBs
f	<bf>:2, <fb>:2, <(bf)>:2	f>	None
e	<be>:3, <(ce)>:2	e>	<(ce)>:<(b)
d	<bd>:2, <db>:2, <(bd)>:2, <(db)>:2, <cd>:2, <dc>:2	{b+d}, <da>	<da>:<(b,c), {cd}:<(b)
a	<ba>:3, <ab>:2, <ca>:2, <aa>, <a+b>	<ca>:<(b)	<da>:<(b,c), {cd}:<(b)
c	<bc>:3, <cb>:3	{b+c}	None
b	<bb>:4	<bb>	None

Seq. Database SDB					
b	4	(4, 3, 0)	1		
c		(3, 2, 0)	(2, 1, 1)	2	
a		(2, 2, 2)	(2, 2, 0)	(1, 2, 1)	1
d		(3, 1, 1)	(1, 1, 2)	(1, 0, 1)	
e		(2, 2, 2)	(1, 1, 0)	(0, 0, 0)	1
f				(1, 1, 0)	2

- Based on the annotations for item-repeating patterns and projected databases, **S** is scanned one more time.

- The set item-repeating patterns generated is $\{<\text{bf}>:2, <\text{fb}>:2, <(\text{bf})\text{f}>:2, <(\text{bf})(\text{bf})>:2, <(\text{bf})(\text{bf})\text{f}>:2, <(\text{bf})(\text{bf})(\text{bf})>:2, <\text{ba}\text{e}>:2, <\text{ab}\text{a}>:2, <\text{cb}\text{c}>:3, <\text{bc}\text{c}>:2\}$
- There are four projected databases: $\langle (\text{ce}) \rangle : \{b\}$, $\langle \text{da} \rangle : \{b,c\}$, $\langle \text{cd} \rangle : \{b\}$ and $\langle \text{ca} \rangle : \{b\}$.

- For a projected database whose annotation contains exactly three items, its associated sequential patterns can be obtained by a simple scan of the projected database.

- For a projected database whose annotation contains more than three items, one can construct frequent item matrix for this projected database and recursively mine its sequential patterns by the alternative-level projection technique.

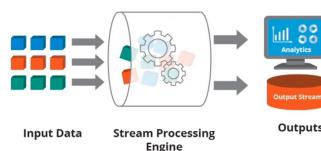
Why FreeSpan outperforms Apriori-like Methods?

- Projects a large sequence database recursively into a set of small projected sequence databases based on the currently mined frequent sets
- The alternatively-level projection in FreeSpan reduces the cost of scanning multiple projected databases and takes advantages of Apriori-like 3-way candidate filtering

Data Stream Analysis

Data stream: Una potenziale sequenza di istanze non limitata.

Può essere rappresentata come $S = \{x_1, \dots, x_n\}$ dove x_i è un'istanza composta da d features e n va a infinito.



Classificazione: Nel caso della classificazione, iniziamo il processo di costruzione del modello quando arriva la prima istanza e continuamo man mano con l'arrivo delle nuove istanze.

Clustering: Nel caso di clustering, iniziamo il processo di costruzione del modello quando arriva la prima istanza e continuamo man mano. La forma dei cluster può cambiare man mano con l'arrivo delle nuove istanze.

PROBLEMI:

- Gestire un numero infinito di istanze
- Costruire il cluster/classifier man mano in modo veloce
- E' impossibile salvare l'intero dataset
- Non possiamo fare scan multiple sull'intero dataset

Concept Drift Problem

I dati che arrivano man mano nel data stream possono avere cambiamenti nelle proprietà statistiche. Il modello generato potrebbe non essere più valido per i nuovi dati che arrivano.

Tipi di concept drift:

- Sudden concept drift:** Tra due istanze consecutive, il cambiamento avviene solo una volta, e dopo questo cambiamento solo istanze della nuova classe sono ricevute.
- Gradual concept drift:** Il numero di istanze che appartengono alla scorsa classe diminuisce gradualmente mentre il numero di nuove istanze che appartengono alla nuova classe aumentano nel tempo.
- Incremental concept drift:** Le istanze che appartengono alla classe precedente evolvono in una nuova classe passo per passo. Dopo che il concept drift è completato, la classe precedente scompare. Le istanze che arrivano durante il concept drift sono in una forma "di passaggio" e non appartengono a nessuna classe.
- Recurring concept drift:** Le istanze cambiano tra due o più caratteristiche statistiche più volte nel tempo. Nessuna classe scompare permanentemente ma arrivano a turni.



Data structures

Non è possibile salvare e gestire l'intero numero di dati. Abbiamo bisogno di sintetizzare i dati che arrivano.

Alcune data structure vengono utilizzate con questi obiettivi.

- Feature vectors = Sintesi delle data instances
- Prototype Arrays = Mantiene solo un numero di istanze rappresentative
- Coreset trees = Mantiene la sintesi in una struttura ad albero
- Grids = Mantiene la data density nel feature space.

I machine learning algorithms devono poter lavorare su questi tipi di data structures e non sulle istanze.

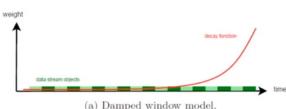
Window Model

Le nuove istanze portano più "peso" rispetto ai vecchi dati, l'importanza delle istanze decresce nel tempo.

Damped window model
Decay functions che scale down il peso delle istanze, dipendentemente dal tempo che passa da quando sono state ricevute.

$$f(t) = 2^{-\lambda t}$$

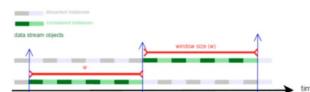
Dove λ è il decay rate, più esso è alto più il peso delle istanze nel tempo decresce rapidamente.



Landmark window model
I dati tra due landmarki sono inclusi nel processo e tutte le istanze hanno ugual peso. Le window consecutive non si intersecano e le nuove window iniziano dal punto da dove le window precedenti terminano.

Sia w la lunghezza della finestra, le istanze che appartengono alle $m - t$ windows sono calcolate usando:

$$W_m = [x_{m+w}, \dots, x_{(m+1)*w-1}] \quad m = \left\lfloor \frac{i}{w} \right\rfloor$$



Sliding windows model

Le finestre scappano una istanza ad ogni step, l'istanza più vecchia va fuori dalla finestra e la più recente entra all'interno della finestra (FIFO).

Tutte le istanze nella finestra hanno ugual peso e finestre consecutive si sovrappongono.

Sia w la lunghezza della finestra, le istanze che appartengono alla $m - t$ window sono calcolate usando:

$$W_m = [x_m, \dots, x_{(m+w-1)}]$$

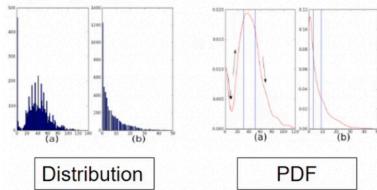


CLUSTERING

Rispetto al problema di classificazione, il clustering è più interessante. Questo perché in un applicazione reale, i dati nel data stream arrivano solitamente senza ground truth.

Adaptive Streaming K-means

- ① Dato un parametro I , inizialmente si accumulano I istanze che verranno usate per costruire il modello iniziale (calcolo dei centroidi iniziali).
- ② Successivamente con l'arrivo dei nuovi dati, se vengono trovati dei cambiamenti, vengono ricalcolati nuovamente i centroidi con i nuovi dati.
 - 1) **Initialization phase:** In questa fase viene determinato K e vengono computati i K centroidi.
 - a. Stimare la probability density function (PDF) dei dati per ogni attributo
 - b. Determinare i cambi direzionali della curva PDF. Ogni cambio identifica una nuova regione. La regione può essere definita come l'area tra due cambi direzionali consecutivi della PDF curve.
 - c. Il numero di regioni è considerato come un candidato per K e i centri di queste regioni sono considerati come possibili centroidi iniziali.



d. Features differenti generalmente mostrano differenti distribuzioni e differenti centroidi

- f. I risultati di clustering con differenti K values vengono comparati usando il silhouette coefficient e il miglior K viene selezionato con i suoi corrispondenti centroidi.

Continuous clustering phase:

- a. Standard deviation e mean dell'input dei dati vengono salvati durante l'esecuzione
- b. L'algoritmo monitora come questi due valori cambiano nel tempo e predice un concept drift in accordo con un cambiamento.
- c. Quando un concept drift viene rilevato, i centroidi correnti non sono più validi. Quindi si ricalcolano usando I istanze.

COMPLESSITÀ: $O(k) + O(d * l) + O(d * k * cs)$ → Fare l'inizializzazione naturalmente costa un po', però allocare le nuove istanze nei cluster è molto veloce.

Algorithm 4 MuDi-Stream offline phase (core mini-clusters)

```

Input: core mini-clusters
1: Mark all cmcs as unvisited ↴
2: repeat
3: Randomly choose an unvisited cmc, called cmcp
4: Mark cmcp as visited
5: if cmcp has neighbors then
6:   Create new final cluster C
7:   Add cmcp to C
8:   Add neighbors of cmcp to C
9:   for each cmc in C do
10:    if cmc is unvisited then
11:      Mark cmc as visited
12:      Add neighbors of cmc to C
13:    end if
14:   end for
15: else
16:   Mark cmcp as noise
17: end if
18: until All cmcs are visited
  
```



CEDAS

E' un algoritmo density based per il clustering di clusters con forma arbitraria. Utilizza una damped windows model lineare invece che esponenziale (linear decay).

Parametri:

- Density threshold (α)
- Decay rate (λ)
- Micro-cluster radius (r_0)

ALGORITMO:

```

Algorithm 5 CEDAS ( $S, \alpha, \lambda, r_0$ )
Input:  $S$ : the input data stream
Input:  $\alpha$ : density threshold
Input:  $\lambda$ : decay rate
Input:  $r_0$ : micro-cluster radius
1: Initialize the micro-cluster structure
2: loop
3:   Read next data instance  $x$  from data stream  $S$ 
4:    $d_{min} =$  Find the distance from  $x$  to the nearest micro-cluster center
5:   if  $d_{min} < r_0$  then
6:     Add  $x$  to the nearest micro-cluster
7:     Energy of the updated micro-cluster = 1
8:   else
9:     Create new micro-cluster with  $x$ 
10:    Energy of the new micro-cluster = 1
11:  end if
12:  Reduce energy of all micro-clusters by  $\lambda$ 
13:  Remove negative energy micro-clusters
14:  if micro-clusters merged then
15:    Update graph structure with micro-clusters that surpass  $\alpha$ 
16:  end if
17: end loop
  
```

COMPLESSITÀ: $O(c)$

Improved Data Stream Clustering Algorithm*

La online phase è composta due fasi principali:

- 1) Initialization phase
- 2) Continuous clustering phase

Vengono utilizzati:

- Major micro-clusters: Alta densità e vengono inclusi nel processo finale di clustering
- Critical micro-clusters: Bassa densità e vengono trattati come possibili outliers.

Damped window viene utilizzata per rimuovere periodicamente Major e Critical micro-clusters.

Parametri:

- Valori di soglia per Major e Critical micro-clusters
- l = lunghezza della data sequenze per l'inizializzazione
- Decay rate (λ)

ALGORITMO:

```

Algorithm 6 Improved data stream clustering online phase ( $S, l, \lambda$ )
Input:  $S$ : the input data stream
Input:  $l$ : length of data sequence used for initialization
Input:  $\lambda$ : decay rate
1: Initialize the micro-cluster structure
2: Run DBSCAN on  $l$  number of data instances
3: % Continuous clustering phase
4: loop
5:   Read next data instance  $x$  from data stream  $S$ 
6:   Add  $x$  to the nearest major micro-cluster OR
7:   Create a new major micro-cluster OR
8:   Create a new micro-cluster with  $x$ 
9:   if it is pruning period then
10:    Remove low weighted critical micro-clusters
11:   end if
12:   Remove low weighted critical micro-clusters
13:   end loop
  
```

COMPLESSITÀ: $O(c)$

Algorithm 7 Improved data stream clustering offline phase (micro-clusters)

Input: S : the input data stream

1: Mark all mcs as unvisited

2: repeat

3: randomly choose an unvisited mc, called *mey*

4: if *mey* is major micro-cluster then

5: Add *mey* to the nearest major micro-cluster reachable to *mey*

6: Create a final cluster by them

7: else if *mey* is critical micro-cluster then

8: Remove the next cycle

9: end if

10: until All mcs are visited

Se la distanza tra un micro-cluster e un altro major micro-cluster è minore o uguale alla somma dei loro raggi, allora sono direttamente density reachable.

Se qualiasi due cluster in un insieme di micro-clusters sono direttamente density reachable, allora l'insieme di micro-clusters è density reachable.

DBIECM

E' un algoritmo distance-based.

Esplora il Davies Bouldin Index (DBI) il quale è utilizzato come criterio di valutazione.

$$V_{DB} = \frac{1}{k} \sum_{i=1}^k R_i$$

where $R_i = \max_{i \neq j} R_{ij}$

Separation

Dispersion

$$S_i = \left(\frac{1}{|C_i|} \sum_{j \in C_i} D^p(x_j, v_i) \right)^{\frac{1}{p}}, p > 0$$

Number of objects in cluster C_i

Centroid of cluster C_i

Algoritmo

Algorithm 8 DBIECM (S, r_0)

Input: S : the input data stream

Input: r_0 : max cluster radius

1: Initialize the cluster structure

2: loop

3: Read next data instance x from data stream S

4: dis_i = Find the distance from x to all cluster centers $C_i, i \in [1, k]$

5: if $dis_i < \text{radius of } C_i$ then

6: Add x to C_i

7: else if $dis_i > r_0$ for all $i \in [1, k]$ then

8: Create new micro-cluster with x

9: else % There exist clusters such that radius of $C_i < dis_i < r_0$

10: Find all clusters such that radius of $C_i < dis_i$

11: Add x to the best cluster, according to DBI

12: end if

13: end loop

Richiede il max cluster radius come parametro. Questo parametro influisce sul numero di cluster finali e di conseguenza sulla qualità del clustering.

Maximum cluster radius dipende fortemente dai dati in input.

DBIECM essendo basato sulla distanza può trovare solo cluster di forma sferica. Inoltre non implementa nessun tipo di window model e di conseguenza tiene di conto di tutti i dati nel final clustering.

Nessun meccanismo di outlier detection è implementato, anche se è possibile specificare un outlier threshold value e marcare i cluster con bassa cardinalità come outliers.

COMPLESSITÀ: $O(k^3)$

CLASSIFICATION

Il nostro obiettivo è di disegnare decision tree learners che leggano ogni esempio al massimo una volta, e usano una piccola costante di tempo nel processarlo.

IDEA: Per trovare il miglior attributo ad uno specifico nodo, può essere sufficiente considerare solo un piccolo subset dei training examples che passano da quel nodo.

- Dato uno stream di esempi, usare i primi per scegliere il root attribute
- Quando il root attribute è stato scelto, i successivi esempi sono passati verso il basso (verso le foglie) e usati per scegliere l'attributo li.

Si usa il Hoeffding bound per decidere quanti esempi sono abbastanza per ogni nodo.

ATTENZIONE: Gli esempi che vedremo sono stati dimostrati teoricamente errati (dovuto all'errore nell'assunzione che vedremo).

Siano:

X	Random attribute che varia in un range R
n	Numero di osservazioni di X
\bar{x}	Valore medio su n osservazioni

Hoeffding bound afferma che con probabilità $1 - \delta$, la media \bar{X} di X è almeno $\bar{x} - \varepsilon$ dove:

$$\varepsilon = \sqrt{\frac{R^2 \ln 1/\delta}{2n}}$$

Se settiamo la probabilità e settiamo ε (errore) possiamo calcolarci quanti esempi ci servono per ogni nodo.

Come viene usato il Hoeffding bound nel decision tree?

Sia:

- $G(X_i)$ la misura euristica usata per testare gli attributi
- X_A l'attributo con valutazione più alta dopo aver visto n samples
- X_B il secondo attributo con valutazione più alta dopo aver visto n esempi.

Dato una probabilità desiderata δ , se dopo aver visto n examples in un nodo:

- $\Delta\bar{G} = \bar{G}(X_A) - \bar{G}(X_B) > \varepsilon$
- $R = \ln(c)$ dove c è il numero di classi

L'Hoeffding bound garantisce che $\Delta\bar{G} \geq \Delta\bar{G} - \varepsilon > 0$, con probabilità $1 - \delta$.

Questo nodo può essere spartito usando X_A e gli esempi successivi saranno passati alle nuove foglie.

I-HASTREAM

E' un algoritmo density based e hierarchical based diviso in due fasi.

- 1) Online phase: Un riassunto dei dati è creato come micro-clusters
- 2) Offline phase: I micro-clusters sono mantenuti in una struttura a grafo come il minimo albero di copertura e il hierarchical clustering è impiegato per il clustering finale.

Algorithm 9 I-HASTREAM offline phase (micro-clusters, α)

Input: micro-clusters

Input: α : weight threshold

1: MST = Update minimum spanning tree(MST , micro-clusters)

2: HC = Emply hierarchical clustering(MST, α)

3: Extract final clustering(HC)

COMPARARE GLI ALGORITMI DI CLUSTERING

Algorithm	Year	Base Algorithm	Phases	Window Model	Cluster Count	Cluster Shape
Adaptive Streaming k-Means	2017	Partitioning based	Online	Sliding	Auto	Hyper-spherical
MuDi-Stream	2016	Density based	Online-offline	Damped	Auto	Arbitrary
CEDAS	2016	Density based	Online	Damped	Auto	Arbitrary
Improved Data Stream Clustering	2017	Density based	Online-offline	Damped	Auto	Arbitrary
DBIECM	2017	Distance based	Online	None	Auto	Hyper-spherical
I-HASTREAM	2015	Density based	Online-offline	Damped	Auto	Arbitrary

Algorithm	Multi Density Clusters	High Dimensional Data	Outlier Detection	Drift Adaption	Expert Knowledge
Adaptive Streaming k-Means	Yes	Suitable	No	Yes	No
MuDi-Stream	Yes	Not suitable	Yes	Yes	Required
CEDAS	No	Suitable	Yes	Yes	Required
Improved Data Stream Clustering	No	Suitable	Yes	Yes	No
DBIECM	Yes (not multi size)	Suitable	No	Yes	Required
I-HASTREAM	Yes	Suitable	Yes	Yes	No

PROBLEMA: Split measures, come information gain e gain index, non può essere espresso come somma S di elementi di \mathcal{Y}_i . (non possono essere rappresentati come somma di elementi). Per questo motivo teoricamente è sbagliato.

La formula corretta è:

$$\epsilon = C_{Gain}(K, N) \sqrt{\frac{\ln(1/\delta)}{2N}} \quad \text{where} \quad C_{Gain}(K, N) = 6(K \log_2 eN + \log_2 2N) + 2 \log_2 K.$$

Pre-pruning viene svolto considerando ad ogni nodo un attributo X_A a "null" che consiste nel non spartire il nodo. Lo split viene formato se, con confidenza $1 - \delta$, il miglior split trovato è migliore in accordo con G rispetto a non spartire.

OSS: E' costoso ricalcolare G per ogni nuovo esempio, per questo motivo VFDT learner permette all'utente di specificare una soglia minima di esempi da raccogliere prima di ricalcolare G .

ALGORITMO SPIEGATO A PAROLE:

- 1) Calcolare l'information gain per gli attributi e determinare i migliori due attributi. (Pre-pruning: Considera un attributo "null" che consiste nel non spartire il nodo).
- 2) Ad ogni nodo, verificare la condizione $\Delta\bar{G} = \bar{G}(X_A) - \bar{G}(X_B) > \epsilon$
- 3) Verificare la condizione:
 - a. Se la condizione è soddisfatta, creare nodi figli basati sul test sul nodo.
 - b. Se la condizione non è soddisfatta, prendere più nuovi esempi e ricalcolare fin quando non viene soddisfatta.

Possibile miglioramento (CVFDT):

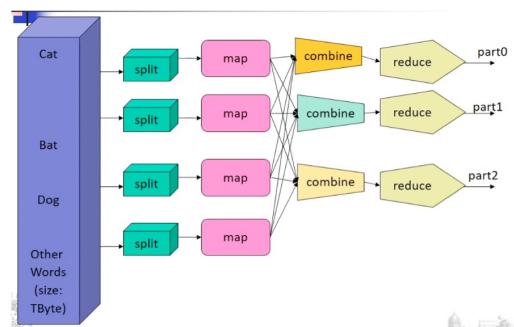
Con il passare del tempo alcuni split del decision tree potrebbero non essere più i migliori:

- 1) Crescere un subtree alternativo con il nuovo best attribute come root quando il vecchio attributo sembra vecchio.
- 2) Periodicamente si usa un insieme di esempi per valutare la qualità dell'albero. Rimpiazzare il vecchio subtree quando la sua alternativa diventa più accurata.

MapReduce and Hadoop

MapReduce è un paradigma il quale si occupa di:

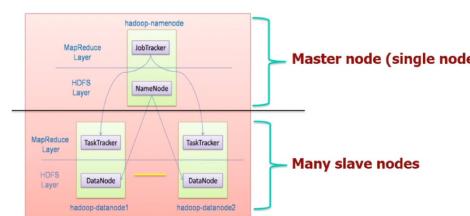
- Parallelizzare la computazione su un cluster di macchine
- Handle machine failures, efficient communications e performance issues.



Cos'è Hadoop?

Hadoop è un framework per processare in modo distribuito large datasets. Hadoop usa il map-reduce paradigm.

Architettura: Master-slave shared-nothing

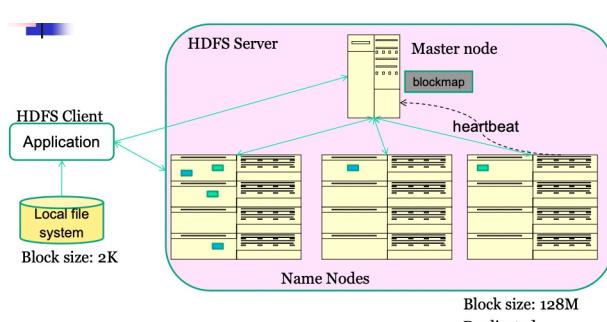


Master: Gestisce i name node, contiene il file system (HDFS). Inoltre contiene il JobTracker che distribuisce i lavori agli slaves (MapReduce).

Slaves: Contengono i dati. Svolgono il lavoro che gli arriva dal master.

IMPORTANTE: Gli slave non devono aver bisogno di condividere i dati.

- Determine if the problem is parallelizable and solvable using MapReduce (ex: Is the data WORM?, large data set).
- Design and implement solution as Mapper classes and Reducer class.
- Compile the source code with Hadoop core.
- Package the code as jar executable.
- Configure the application (job) to the number of mappers and reducers (tasks), input and output streams
- Load the data (or use it on previously available data)
- Launch the job and monitor.
- Study the result.



Hadoop Distributed File System (HDFS)

Il file system di hadoop è ottimizzato per l'alto throughput.
Esso è scalabile (replicas) e affidabile (fault tolerance).

Data Characteristics

La quantità di dati è enorme (giga to tera), utilizza l'idea del WORM (Write once read many), cioè quando un file viene creato, scritto e chiuso, necessita di non essere cambiato (quest'assunzione semplifica la coerenza).

NameNodes e DataNodes

L'HDFS è composto da:

- Un singolo NameNode (master) che gestisce il file system namespace e l'accesso ai file da parte dei clienti. Si occupano delle: Letture sui server, Scritture sui server, creazione, eliminazione e replicazione dei blocchi.
- Più DataNodes (solitamente uno per cluster) i quali gestiscono lo store dei dati tra i server che compongono il cluster. I file vengono divisi in uno o più blocchi ed ogni insieme di blocchi viene storato nei DataNodes.

Namenode

Il namenode usa un transaction log chiamato EditLog per tracciare ogni cambiamento che occorre nel filesystem metadata. L>EditLog è salvato nel Namenode's local filesystem. L'intero filesystem namespace (che include mapping dei block to files e le proprietà del file system) è salvato in un file Fslimage.

Il namenode mantiene in memoria un'immagine dell'intero file system namespace e file blockmap.

4GB di RAM local è sufficiente.

Quando un Namenode si avvia:

- 1) Ottiene il Fslimage e EditLog dal suo local file system
- 2) Aggiorna il Fslimage con l>EditLog information
- 3) Stores una copia del Fslimage come checkpoint. (Il checkpoint viene creato periodicamente per poter fare il restore del sistema).

Fault Tolerance

Un fallimento in un sistema distribuito con molti componenti è molto probabile. Il sistema deve essere in grado di rilevare e recuperare automaticamente il fallimento.

Fallimento sul DataNode

- 1) Il NameNode si accorge del fallimento del DataNode dall'assenza dei Heartbeat messages.
- 2) Il NameNode marks Datanode senza heartbeat e non invia nessuna richiesta I/O a loro.

Fallimento sui Metadati

Fslimage e EditLog sono le data structure centrali del HDFS, per questo motivo un Namenode può essere configurato per mantenere copie multiple di essi per evitare che ci corrompano.

Principi di design di Hadoop:

- Processare grandi quantità di dati
- Computazione suddivisa su più macchine più economiche (scalabilità orizzontale)
- Parallelizzazione e distribuzione automatica (non viene "vista" dagli user)
- Programming abstraction semplice e pulita (L'utente fornisce solo due funzioni "map" e "reduce").

Replicazione

HDFS è disegnato per salvare grandi quantità di files su più macchine in grossi cluster.

- Ogni file è una sequenza di blocchi.
- Tutti i blocchi di un file, eccetto l'ultimo, hanno uguali dimensioni.

I blocchi vengono replicati per gestire la fault tolerance.

I Namenode ricevono un Heartbeat e un BlockReport da ogni DataNode nel cluster:

- BlockReport contiene tutti i blocchi su un Datanode.

Rack-aware replica placement

Il piazzamento di un replica è un'operazione critica per le performance e affidabilità del HDFS. L'ottimizzazione di questo processo distingue HDFS dagli altri file system distribuiti.

Rack-aware replica placement

- 1) Il NameNode determina il rack id per ogni DataNode
- 2) Le repliche sono piazzate:
 - a. Una su un nodo del rack locale
 - b. Una su un nodo differente nel rack locale
 - c. Una su un nodo in un rack differente

Data Blocks

La dimensione dei blocchi è tipicamente 64MB (o in caso 128MB).

Un file viene diviso in blocchi da 64MB e salvato.

Staging

Quando un client richiede la creazione di un file esso non raggiunge immediatamente il NameNode.

- 1) Il client HDFS salva in cache il file temporaneamente
- 2) Quando i dati raggiungono la dimensione di un blocco HDFS vengono inviati al NameNode
- 3) Il Namenode inserisce il filename nella sua gerarchia e alloca il datablock per il file
- 4) Il Namenode risponde al client con l'identità del DataNode e la destinazione dei replicas del blocco.
- 5) Il client libera la sua memoria eliminando il file
- 6) Il client invia un messaggio che il file è "chiuso"

Se il Namenode fallisce prima che il file è chiuso, il file viene perso.

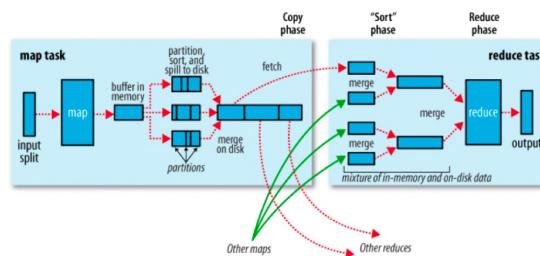
Limitazioni

HDFS: Pecca di alta disponibilità, inefficiente nel gestire piccoli file.

MapReduce: Non permette gli use-case che necessitano di real-time data access, disegnare dei buoni algoritmi map-reduce.

Security

MapReduce



L'utente si deve occupare di definire le funzioni di map e reduce.

Parallel K-MEANS

La computazione più costosa è il calcolo delle distanze. Per ogni iterazione, richiede di calcolare n*k distanze dove n è il numero degli oggetti e k è il numero dei cluster che devono essere creati.

Intuizione: I calcoli della distanza di due oggetti differenti rispetto al centro possono essere eseguite in parallelo.

MAP:

- Key = Offset in bytes del record rispetto all'inizio dell'input file
- Value = Una stringa contenente le info relative al sample

- 1) Dato un array contenente il centro dei cluster, il mapper calcola il centroide più vicino per ogni sample in input.
- 2) Produce in output:
 - a. Key = Index del centroide più vicino
 - b. Value = Sample informations

COMBINE: Il combiner fa la somma dei valori dei punti assegnati allo stesso cluster e il loro numero.

REDUCE: Aggiorna i nuovi centri.

- 1) Somma tutti i samples e calcolare il numero totale di samples assegnati allo stesso cluster
- 2) Calcola i nuovi centri che saranno utilizzati per la nuova iterazione

Calcola la dist. da ogni centroide e trova il centroide + vicino

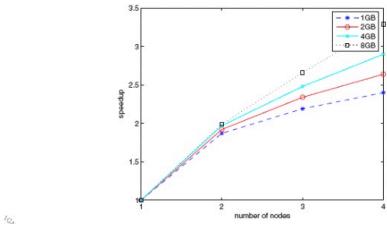


Algorithm 1. map (key, value)

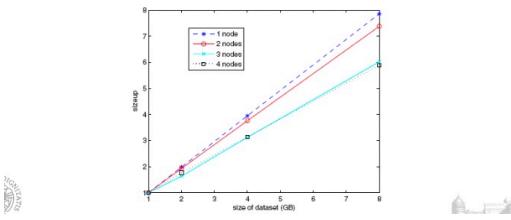
Input: Global variable *centers*, the offset *key*, the sample *value*
Output: <key', value'> pair, where the key' is the index of the closest center point and value' is a string comprise of sample information

1. Construct the sample instance from *value*;
2. *minDis* = Double.MAX_VALUE;
3. *index* = -1;
4. For *i*=0 to *centers.length* do
 - dis* = ComputeDist(*instance*, *centers*[*i*]);
 - If *dis* < *minDis* {
 - minDis* = *dis*;
 - index* = *i*;
}
5. End For
6. Take *index* as *key'*;
7. Construct *value'* as a string comprise of the values of different dimensions;
8. output <*key'*, *value'*> pair;
9. End

- **Speedup:** measures how much a parallel algorithm is faster than a corresponding sequential algorithm



- **Scaleup:** holds the number of computers in the system constant, and grows the size of the datasets by the factor m. Scaleup measures how much longer it takes on a given system, when the dataset size is m-times larger than the original dataset.



Parallel FP-Growth

L'algoritmo parallelizzato di FP-Growth è composto da 5 fasi, di cui tre map-reduce tasks:

- 1) Sharding: Dividere il DB in parti e salvare le varie parti su computer differenti. Ogni parte è chiamata "shard"
- 2) Parallel Counting: MAP-REDUCE task che conta il valore di supporto di tutti gli item che compaiono nel DB. Ogni mapper prende in input una shard del DB. Il risultato è una F-list.
OSS: Questo step implicitamente scopre anche il "vocabolario degli item" I, che solitamente è sconosciuto per grandi DB.

```

Procedure: Mapper(key, value=Ti)
foreach item ai in Ti do
    Call Output((ai, 1));
end
Procedure: Reducer(key=ai, value=S(ai))
C ← 0;
foreach item '1' in Ti do
    C ← C + 1;
end
Call Output((null, ai + C));

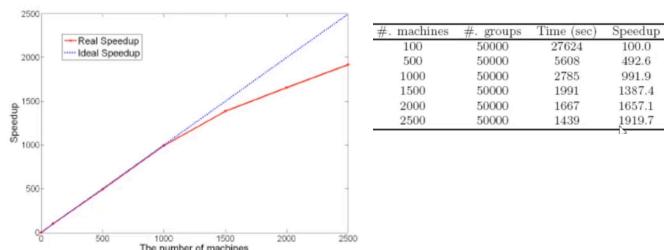
```

Per ogni item nella transaction produci <:item, 1>

conta occorrente per ogni item

- 3) Grouping Items: Divide tutti gli items in Q gruppi:
La lista dei gruppi è chiamata "G-list" e ad ogni gruppo viene assegnato un group ID univoco (gid). Essendo che F-list e G-list sono entrambe piccole e la complessità in tempo è O(|I|), allora questo step può essere completato su un singolo computer in pochi secondi.

SPEEDUP:



Perché Parallelizzare FP-Growth?

Storage: Per grossi DB, il corrispondente FP-tree potrebbe non entrare in memoria.

Computation distribution: Tutti gli step dell'FP-growth possono essere parallelizzati, specialmente le chiamate ricorsive.

Costly communication: Se il partizionamento del DB è in gruppi di transizioni successive, gli FP-trees distribuiti possono essere inter-dependent, quindi richiedono frequenti sincronizzazioni tra esecuzioni parallele.

Support Value: Per DB grandi, deve essere grande abbastanza o l'FP-tree supererebbe lo spazio di archiviazione.

Algorithm 2. combine (key, V)

Input: key is the index of the cluster, V is the list of the samples assigned to the same cluster
Output: <key', value'> pair, where the key' is the index of the cluster, value' is a string comprised of sum of the samples in the same cluster and the sample number

```

1. Initialize one array to record the sum of value of each dimensions of the samples contained in the same cluster, i.e. the samples in the list V;
2. Initialize a counter num as 0 to record the sum of sample number in the same cluster;
3. while(V.hasNext()){
    Construct the sample instance from V.next();
    Add the values of different dimensions of instance to the array
    num++;
4. }
5. Take key as key';
6. Construct value' as a string comprised of the sum values of different dimensions and num;
7. output <key', value'> pair;
8. End

```

Algorithm 3. reduce (key, V)

Input: key is the index of the cluster, V is the list of the partial sums from different host
Output: <key', value'> pair, where the key' is the index of the cluster, value' is a string representing the new center

```

1. Initialize one array record the sum of value of each dimensions of the samples contained in the same cluster, e.g. the samples in the list V;
2. Initialize a counter NUM as 0 to record the sum of sample number in the same cluster;
3. while(V.hasNext()){
    Construct the sample instance from V.next();
    Add the values of different dimensions of instance to the array
    NUM += num;
4. }
5. Divide the entries of the array by NUM to get the new center's coordinates;
6. Take key as key';
7. Construct value' as a string comprise of the center's coordinates;
8. output <key', value'> pair;
9. End

```

4) Parallel FP-Growth:

- a. Mapper: Ha come obiettivo il generare Group-dependent transactions.
Prende in input una shard del DB, legge la G-list e produce in output una o più key-value pairs dove:
- i. Key = Group-ID
 - ii. Value = Generated group-dependent Transaction

```

Procedure: Mapper(key, value=Ti)
Load G-List;
Generate Hash Table H from G-List;
a[] ← Split(Ti);
for j = |Ti| - 1 to 0 do
    HashNum ← getHashNum(H, a[j]);
    if HashNum ≠ Null then
        Delete all pairs which hash value is HashNum
        in H;
    Call
    Output((HashNum, a[0] + a[1] + ... + a[j]));
end
end

```

- b. Combiner: Ragggruppa tutte le corrispondenti group-dependent transactions in una shard di group-dependent transactions (basandosi sul group-ID).
- c. Reducer: Applica la FP-Growth su un group-dependent shard.
Ogni reducer processa una o più group-dependent shard, una alla volta. Per ogni shard, il reducer costruisce un FP-tree locale e cresce i suoi conditional FP-trees ricorsivamente. Durante il processo ricorsivo, può outputtare i pattern scoperti.

```

Procedure: Reducer(key=gid,value=DBgid)
Load G-List;
nowGroup ← G-Listgid;
LocalFPtree ← clear;
foreach Ti in DBgid do
    Call insert - build - fp - tree(LocalFPtree,Ti);
end
foreach ai in nowGroup do
    Define and clear a size K max heap : HP;
    Call TopKFPGrowth(LocalFPtree,ai,HP);
    foreach vi in HP do
        Call Output((null, vi + supp(vi)));
    end
end

```

5) Aggregating:

```

Procedure: Mapper(key, value=v + supp(v))
foreach item ai in v do
    Call Output((ai, v + supp(v)));
end
Procedure: Reducer(key=ai, value=S(v + supp(v)))
Define and clear a size K max heap : HP;
foreach pattern v in v + supp(v) do
    if |HP| < K then
        insert v + supp(v) into HP;
    else
        if supp(HP[0].v) < supp(v) then
            delete top element in HP;
            insert v + supp(v) into HP;
        end
    end
end
Call Output((null, ai + C));

```