Machine Learning - Lezione 8 (a partire da)

Parte I
Selezione delle Features

Un alto numero di features riduce l'interpretabilità del modello e consente una maggiore efficienza; rimuovere troppe variabili però può provocare perdite di informazione. È necessario dunque togliere solamente le variabili ridondanti o poco informative. Un attributo irrilevante non è detto che non sia collegato alla variabile risposta.

Alcuni algoritmi di Machine Learning effettuano già una loro selezione delle features (Naive Bayes), mentre per altri è necessario verificare a mano le variabili. Testare col bruteforce ogni modello possibile è dispendioso computazionalmente, dunque si preferisce filtrare le variabili prima della fase di training (filter approach) o ancora dichiarando il modello e testando la performance del classificatore per testare la migliore combinazione di attributi.

Filtro. Con un algoritmo di filtro, il dataset è analizzato da un algoritmo ricorsivo che cerca le variabili da inserire nel modello e restituisce l'elenco delle features. L'algoritmo genera una lista di features da passare ad una funzione obiettivo da massimizzare/minimizzare.

Wrapper. L'algoritmo di wrapper funziona allo stesso modo, ma la funzione obiettivo è direttamente un classificatore per cui è analizzata la prestazione.

Le variabili sono inserite nel modello se l'aumento dell'informazione riportato supera una certa soglia o solo semplicemente selezionate le k variaibli più significative.
Con un filtro univariato, è facile selezionare le variabili irrilevanti ma non quelle ridondanti: per fare ciò è necessario usare un filtro multivariato. Un buon attributo deve essere fortemente associato con la variabile di classe e debolmente associato con le altre variabili del modello.
I test univariati (parametrici) sono i test t, Anova, (non parametrici) Mann-Whitney, Kruskal-Wallis e Permutation test; i test multivariati sono i Correlation Feature Selection, Relief e Blanket.
I vantaggi dei metodi univariati sono la velocità e la scabilità, inoltre sono indipendenti dalla natura del classificatore (cosa che può anche essere uno svantaggio); tuttavia ignora dipendenze tra gli attributi e le interazioni col classificatore.
I metodi multivariati modellano le dipendenze tra gli attributi e offrono una prestazione a metà tra i modelli Wrapper e i modelli univariati; sono algoritmi più pesanti rispetto ai filtri univariati.

I vantaggi della riduzione del numero delle features sono la riduzione della dimensione dei dati, la scalabilità dell'algoritmo, una maggiore interpretabilità e un miglioramento dell'accuratezza dovuta alla riduzione di dimensionalità del dataset (è meno probabile un overfitting).

1 Creazione di nuove Features

In base alla conoscenza del dataset, è possibile creare nuove features partendo da quelle esistenti tramite semplici operazioni matematiche o rimappature dello spazio originale.

2 Classificazione non binaria

La classificazione non binaria può verificarsi in due forme:

- Multiclasse

- Multilabel

Una divisione multiclasse si verifica quando la variaible risposta è composta da classi complementari ed esaustive (salvo l'aggiunta di un valore nullo); una divisione multilabel si verifica quando la variaible risposta può assumere più di un valore. Un altro caso si verifica per la classificazione di votazioni.

Per la classificazione multiclasse, è facile ipotizzare un modello per ogni variabile risposta (One-vs-All): la variabile risposta assume dunque il valore più probabile o tutti i valori che superano una certa soglia.

3 Regolarizzazione

Nelle reti neurali, come in altri algoritmi, devono essere regolarizzati:

$$E(w, \lambda) = \frac{1}{m} \sum_{i=1}^{m} (y_i - \widehat{y_i})^2 + \frac{\lambda}{2} \sum_{j=1}^{K} w_i^2$$

Il parametro λ regola il lisciamento della formula: bisogna selezionare un valore di λ che sia ottimale per il compito svolto (ridurre l'errore per un nuovo insieme di dati).

- K: numero di parametri liberi;
- λ: parametro di regolarizzazione.

Lo schema train-test dataset non è più valido per la stima del parametro λ: è necessario distinguere il train-dataset dal validation-dataset per evitare l'overfitting. Tuttavia non è sempre possibile effettuare questa divisione a causa della scarsità dei dati.
Per la scelta delle features tramite il metodo wrapper, il calcolo di λ deve già essere effettuato nell'algoritmo di ricerca.

4 Regressione lineare

La variabile dipendente (o *di risposta*) in un modello di regressione lineare è una variabile continua. Il modello spiega una variabile dall'insieme degli attributi del dataset. Si cerca una funzione:

$$f : R^k \to R$$
$$\widehat{Y} = f(\underline{X}) + \epsilon$$

a cui si aggiunge un errore (o residuo) ϵ che rappresenta lo scarto del modello, dovuto a incertezza o ignoranza della formula reale.
Un modello di regressione lineare è una funzione lineare di una matrice (*del disegno*) di valori X pesata con un vettore w. Se la matrice del disegno è composta da una sola variabile indipendente, il modello è detto *modello lineare semplice*: non ha funzioni pratiche ma è interessante dal punto di vista teorico perchè presenta le stesse problematiche dei modelli multivariati, polinomiali e con componenti rettangolari.
L'obiettivo è minimizzare l'errore standard commesso nel prevedere y tramite la funzione f:

$$\begin{aligned} SSE &= \sum_{i=1}^{m} e_i^2 \\ &= \sum_{i=1}^{m} [y_i - f(\underline{X})]^2 \\ &= \sum_{i=1}^{m} [y_i - x_i w_i - b]^2 \end{aligned}$$

Aggiungere variabili indipendenti non modifica il ragionamento nè la formulazione matematica del problema. Si possono aggiungere altre variabili dall'elevamento a potenza di quelle presenti o dalla loro moltiplicazione: i parametri sono detti *gradi di libertà* del modello. Aggiungere troppi gradi di libertà però può provocare overfitting dei dati, oltre a ridurre l'interpretabilità del modello.

4.1 Critica del modello

Il modello lineare, pur essendo molto semplice, si basa su delle assunzioni molto forti.

Assunzioni relative ai residui. I residui devono avere media nulla e devono essere indipendenti (e quindi anche incorrelati) per tutti i valori di x. Inoltre la varianza dei residui deve essere costante (devono cioè essere *omoschedastici*).

$$\underline{\epsilon} = N(\underline{\mu}, \sigma I)$$

Per verificare queste ipotesi, non esiste un test considerato valido, tuttavia si usano strumenti grafici e test statistici, parametrici o non parametrici.
Per verificare la distribuzione del vettore dei residui, si usano i test Kolmogorov-Smirnoff o Shapiro-Wilk, mentre Durbin-Watson è considerato valido per calcolare l'interdipendenza dei residui. La distanza di Cook, inoltre, stabilisce se un'osservazione è particolarmente influente per la stima del modello.
È possibile ridurre l'eteroschedasticità del modello operando sul logaritmo della variabile.

Significatività dei coefficienti. Un coefficiente può anche non essere particolarmente significativo nel modello (ha cioè peso pari a 0). Bisogna dunque calcolare quali coefficienti sono significativamente non nulli; tuttavia, avendo a disposizione solamente la stima, sono necessari test statistici: basta calcolare un intervallo di confidenza ed escludere, con α pari a quello dell'intervallo, i coefficienti per cui il valore 0 è interno all'intervallo. Altro approccio è effettuare un test statistico (t-test) sulla significatività del singolo coefficiente ed escludere con p-value $> \alpha$.

4.2 Valutazione del modello

Il coefficiente $\tilde{R}^2$ di determinazione rappresenta la percentuale di varianza totale spiegata dal modello:

$$\tilde{R}^2 = 1 - \frac{RSS}{TSS} = 1 - \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} (y_i - \bar{y}_i)^2}$$

Si può comunque verificare l'overfitting: si usa il coefficiente aggiustato.

$$\tilde{R}^2 = 1 - (1 - R^2)\frac{m-1}{m-k-1}$$

che generalmente assume valori minori di $\tilde{R}^2$. Se la differenza tra i due valori non è particolarmente alta, (probabilmente) non si verifica l'overfitting.

Intervalli di confidenza e di previsione. In un modello lineare semplice, si distinguono l'intervallo di confidenza, che contiene il valore medio $E[Y|\underline{X}]$, e l'intervallo di previsione, che contiene la singola realizzazione Y.

Selezione variabili. Sono possibili una selezione *forward* e una selezione *backward*. La prima parte dal modello vuoto e inserisce variabili a ogni passo del ciclo; si usa con un grande numero di variabili esplicative per semplificare il modello. L'approccio backward invece parte dal modello pieno ed esclude variabili a ogni ciclo; si usa con poche variabili per eliminare quelle poco significative.

Parte II
Clustering.

La *cluster analysis* non è risolvibile perchè la divisione in gruppi non è deterministica ma è un'interpretazione dello spettatore; tuttavia esistono metodi per verificare la bontà della divisione. La *cluster analysis* serve per dividere le osservazioni in gruppi omogenei diversi tra di loro. Così si riduce la dimensionalità del dataset: è proposta una singola osservazione per ogni gruppo (si cerca l'archetipo). Maggiore è l'omogeneità interna al gruppo, maggiore è la differenza tra i gruppi (è più precisa la classificazione).
Il problema è capire cosa è simile e cosa diverso: si usano algoritmi di clustering ma non è detto che i dati siano divisibili (e nemmeno che i parametri inseriti siano utili). Gli algoritmi si dividono in diverse tipologie:

- partizionale / gerarchico;
- escusivo / overlapping / fuzzy;
- completo / parziale.

Partizionale. I cluster sono esaustivi e mutualmente esclusivi tra di loro; il problema è individuare il numero di cluster.

Gerarchico. Si forma una struttura gerarchica (deondogramma) che divide in cluster e sotto-cluster le osservazioni; il metodo più semplice è quello agglomerativo.

Esclusivo. Ogni osservazione è assegnata ad un singolo cluster.

Overlapping. Un'osservazione può essere assegnata anche a più cluster.

Fuzzy. Ogni osservazione è assegnata ad ogni cluster con un grado di *membership* compreso in $[0; 1]$.

Completo. Ogni osservazione è assegnata ad (almeno) un cluster.

Parziale. Ci possono essere osservazioni non appartenenti a nessun cluster.

In caso di cluster ben separati, la similitudine di un'osservazione con la più dissimile osservazione del proprio cluster è comunque maggiore rispetto alla similitudine con l'osservazione più simile di un'altra classe. Nel caso di cluster globulari (a iper-sfera, iper-ellissoide) ogni osservazione è più simile al prototipo del proprio cluster rispetto a quelli di altri cluster.

Gli algoritmi a centroide definiscono un prototipo (che può anche non essere un'osservazione) e verificano la distanza delle osservazioni da questi. Secondo gli algoritmi di densità, un cluster è definito come una zona ad alta densità di osservazioni delimitata da zone di scarsa densità. Gli algoritmi a grafo invece connettono osservazioni vicine tra di loro con un collegamento: ogni grafo è un cluster.

Per effettuare una cluster analysis bisogna comunque fare una selezione delle features: possono essere incluse solo certe caratteristiche o possono essere introdotte nuove. Quindi si applica l'algoritmo di clustering selezionando la misura di prossimità e l'algoritmo corretto. Infine bisogna validare il risultato ottenuto dall'algoritmo: si hanno una serie di indici di prestazione. Infine si devono interpretare i dati (con l'aiuto di un esperto di dominio).

Un'analisi di clustering difficilmente è significativa a livallo scientifico ma rappresenta delle linee guida per studi successivi del settore.

1 Prossimità.

La prossimità è calcolabile sia come similarità che come dissimilarità: la matrice di prossimità sostituisce completamente il dataset per utilità. Generalmente con una delle due misure si può calcolare anche l'altra se la scala è lineare; sono previste anche scale non lineari (che appiattiscono distanze troppo elevate).

La prossimità si calcola in modo diverso in base alla natura della variabile:

Nominali. Per osservazioni nominali si calcola 1 se le variabili hanno lo stesso valore (per la similità, per la dissimilarità il contrario), 0 altrimenti.

Ordinali. Per preservare l'ordine, si misura la differenza in ordine assoluto: ogni valore è quantificato, in base all'ordine, come numero da 1 a p (dove p è il numero di valori possibili). Si calcola dunque

$$d = \frac{|x-y|}{n-1}$$
$$s = 1-d$$

Si presuppone che la distanza tra due elementi continui sia sempre la medesima. Per ovviare a questo problema, si usa una determinata funzione di utilità per mappare i valori della variabile ordinale.

Variabili continue. Generalmente si usa la distanza in ordine assoluto o particolari tipi di distanza (Minkowski).

2 Distanze e indici.

2.1 Distanza di Minkowski.

Per calcolare distanze nel caso in cui le variabili siano tutte numeriche, si usa la distanza di Minkowski:

$$d(x,y) = \sqrt[r]{\sum_{k=1}^{n} |x_k - y_k|^r}$$

Se $r = 1$, si calcola la distanza di Manhattan; per $r = 2$ la distanza euclidea e per $n- > \inf$ si ha la distanza suprema.

La distanza di Monkowski è una *distanza* perchè ha alcune caratteristiche: non è mai negativa ed è simmetrica; inoltre vale la disuguaglianza triangolare.

2.2 SMC.

Il Simple Matching Coefficient calcola il numero di match (in percentuale) tra due osservazioni:

$$SMC(x,y) = \frac{\text{n matching attributes}}{\text{n attributes}}$$

Possono essere calcolati anche solamente i match se per qualche variabile il valore è > 0 se la mancanza dell'attributo non è sensata.

2.3 Coefficiente di Jaccard.

2.4 Cosine similarity.

Si definisce la similarità al coseno (molto utilizzata in information retrival) che consente di calcolare da distanza tra due documenti.

$$cos(x,y) = \frac{x\dot{y}}{||x||||y||}$$

È utile per calcolare informazioni sparse. Assume valori nell'intervallo $[0;1]$ se i dati x e y sono dati di conteggio (quindi non negativi), altrimenti l'intervallo è tra valori è $[-1;+1]$.

2.5 Correlazione.

Si usa la correlazione di Pearson (sotto l'ipotesi di distribuzione gaussiana multivariata) per calcolare la prossimità tra due vettori. Assume valori nell'intervallo $[-1;+1]$.

2.6 Distanza di Mahlanobis.

La distanza di Mahlanobis permette l'analisi di attributi con ordini di grandezza diversi: la similarità è data dalla prossimità dei vettori.

$$Mahal(x,y) = (x-y)\Sigma^{-1}(x-y)^T$$

Per calcolare la dissimilarità tra attributi diversi tra di loro, si definisce una variabile δ_k che assume valore 0 se entrambe le variabili hanno il k-esimo attributo pari a 0 o se uno dei due valori è missing, 1 altrimenti.

$$similarity(x,y) = \frac{\sum_{k=1}^{n} \delta_k \cdot S_k(x,y)}{\sum_{k=1}^{n} \delta_k}$$

Per pesare in modo diverso gli attriubuti basta inserire pesi nella formula precedente.

3 Algoritmi di clustering.

3.1 K-Means.

È un algoritmo di clustering esclusivo e completo che definisce un prototipo di osservazione per ogni cluster (centroide) non necessariamente coincidente con un'osservazione. Un'osservazione è assegnata al cluster il cui centroide ha distanza minore. I centroidi

sono posizionati in modo iterativo e rappresentano la media degli attributi delle osservazioni appartenenti alla classe. L'algoritmo piazza casualmente un numero k (arbitrario) di centroidi, spostandoli ad ogni iterazione per far sì che coincidano col valore medio delle osservazioni di appartenenza; dunque si riassegnano le osservazioni e si spostano nuovamente i centroidi finchè due cicli successivi non convergono.

L'algoritmo è molto utilizzato, grazie alla rapida velocità di esecuzione e della semplicità dell'algoritmo. Tuttavia presenta alcuni problemi: la scelta del numero di cluster e delle osservazioni di partenza sono cruciali e condizionano l'output. L'algoritmo soffre la presenza di outliers o cluster vuoti; può anche essere utilizzato per individuare outliers: facendo partire l'algoritmo, si considerano *outlier* i membri di cluster con una singola osservazione. L'algoritmo individua solamente cluster iper-sferici e può dare problemi quando il numero di osservazioni nei vari cluster è significativamente diverso. È possibile, alla fine dell'algoritmo, riunire cluster i cui centroidi distano meno di una certa soglia: può darsi che dividere le osservazioni in k gruppi si insufficiente e sia opportuno usare un numero maggiore per poi fondere i cluster. Una variante più robusta è usare la variante PAM (*Partitioning Around Metoid*): al posto dei centroidi usa le osservazioni medie, ma l'algoritmo risulta appesantito.

3.2 Fuzzy c-means.

Quando il cluster di un'osservazione non è ben definito, l'appartenenza ad uno dei due è arbitrario: è utilizzato il concetto di *membership* che indica il grado di appartenenza di un'osservazione a un dato cluster. Per ogni coppia è assegnato un valore $w_{i,j} \geq 0$ (dove i indica l'osservazione i-esima, j il j-esimo cluster). In questo modo non si formano cluster vuoti e ogni osservazione è assegnata a tutti i cluster.

L'algoritmo ha tutti i vantaggi e svantaggi del k-means, essendone la versione fuzzy, ma è computazionalmente più pesante.

3.3 Massima aspettativa.

Si considera così un problema diverso: il dataset è concepito come un aggregato di distribuzioni di probabilità (spesso si usa la normale-multivariata) e si calcolano centroidi e distribuzioni di ogni osservazione. Il ragionamento è generativo: i cluster esistono in quanto l'osservazione è regolato dalla componente che genera la distribuzione. Il processo generativo dunque avviene secondo la formula:

$$p(\underline{x}|\Theta) = \sum_{j=1}^{k} w_j(\underline{x}|\theta_j)$$

Tuttavia le distribuzioni non sono note: è possibile stimarle grazie ad un algoritmo iterativo. L'algoritmo di *expectation maximization* ipotizza inizialmente una distribuzione per poi adattarla ai dati a ogni iterazione. Le osservazioni sono assegnate al cluster con la probabilità più alta.

L'algoritmo è molto pesante computazionalmente, e un'operazione di training è molto lenta ed onerosa (si rischia inoltre di raggiungere un minimo locale che non sia una stima della distribuzione reale); inoltre si applica bene solamente a dataset molto numerosi senza fenomeni di collinearità, ma l'algoritmo è più flessibile di k-means o c-means.

3.4 Kohonen MAPS o SOMs.

Questi algoritmi usano reti neurali (convunazionali) che mappano lo spazio e assegnano l'osservazione ad un cluster. Ci si muove da uno spazio continuo (altamente dimensionale) ad uno spazio discreto (a bassa dimensionalità).

Una mappa auto-organizzante (*self organizing*) assegna a cluster vicini osservazioni simili e a cluster lontani osservazioni lontane. La forma della mappa non è significativa: possono essere usate mappe di ogni dimensionalità. L'osservazione è analizzata da ogni neurone e associata al neurone che massimizza la probabilità di appartenenza; dunque sono aggiornati i neuroni vicini e quello di associazione.

I neuroni dunque sono mappati per trovare l'appartenenza a un cluster. La facilità di interpretazione è alta: si riduce notevolmente la complessità spaziale.

4 Classificazione a grafo.

Il grafo della distribuzione, grazie al concetto di *vicinato*, è sfruttato per contare il numero di *vicini* comuni a due osservazioni. Il vicinato è definito *sparsificando* il grafo, cioè eliminando archi superiori

a una certa soglia (il valore dell'arco corrisponde alla distanza tra le due osservazioni); i valori minori di tale soglia sono rimossi (cioè posti a 0). La matrice risultante non è più altrettanto densa, quindi il grafo è nettamente semplificato (maggiore è il valore soglia, maggiore sarà la sparsificazione del nuovo grafo).
Il grafo può essere sparsificato o calcolando il valore soglia o considerando solamente le k osservazioni più vicine. Se la sparsificazione è efficace, si può già avere una struttura dei cluster: possono essere rimosse le osservazioni la cui distanza è particolarmente bassa. Così si riduce la dimensione del dataset: si eliminano le osservazioni la cui distanza è nulla; sono facilmente individuabili osservazioni outliers e rumorosi; e inoltre esiste già un'abbondante letteratura sulla gestione dei grafi (per spezzarli e trattarli in generale). Questi algoritmi si basano sul *taglio minimo*, cioè si cercano degli archi che, se rimossi, provocano meno danni nel processo di clusterizzazione. Il concetto è simile a quello degli algoritmi gerarchici: non è più un algoritmo agglomerativo ma divisivo. Se il grafo risultante è ancora troppo coeso per effettuare una clusterizzazione, è sufficiente iterare più volte.

4.1 MST.

Il *Minimum Spanning Tree* individua cluster in un grafo minimizzando la distanza tra le osservazioni in un albero di supporto. L'albero non può contenere cicli e deve contenere tutte le osservazioni; calcolando la somma dei pesi tra gli archi deve, tra tutte le classi di alberi, essere il minore possibile. Sono considerati i collegamenti col valore minore, finchè non è costruito un albero: da ogni osservazione partono solamente i due archi col minore valore (a patto che non si formino cicli). L'albero così ottenuto è lavorato eliminando gli archi corrispondenti alla distanza maggiore: è costruito un dendogramma ma in modo divisivo.

4.2 Opossum.

Questo algoritmo funziona su grafi di grandi dimensioni calcolando il grafo di sparsificazione.

4.3 Chameleon.

Questo algoritmo unisce i cluster quando alcuni valori non sono particolarmente diversi tra di loro.

5 Cluster Validity.

Si analizzano i cluster per verificare che al loro interno contengano elementi simili e che cluster diversi contentgano elementi diversi. Per prima cosa bisogna chiedersi se si possono individuare strutture non randomiche alla distribuzione di cluster; inoltre è necessario verificare il numero ottimale di cluster. Per validare i cluster si usano indici esterni (o supervisionati), usati solamente a posteriori per verificare l'efficacia relativa rispetto ad un obiettivo; sono indici applicabili solametne a fine comparativo. Esistono anche indici interni (o non supervisionati), che misurano i concetti di coesione (similitudine degli elementi interni) o separazione (dissimilitudine tra i cluster). Le misure relative invece comparano differenti divisioni in cluster per verificare il numero ottimale di cluster.

5.1 Indici esterni.

Essendo già nota la divisione in cluster delle osservazioni, si può stabilire quanto la divisione data da un algoritmo sia efficace rispetto alla divisione *esterna* (conosciuta prioristicamente) del dataset. Si possono verificare più casi:

- x e y appartengono allo stesso cluster e alla stessa categoria a priori;
- x e y appartengono allo stesso cluster ma a diverse categorie a priori;
- x e y appartengono a diversi cluster ma alla stessa categoria a priori;
- x e y appartengono a diversi cluster e a diversi categorie a priori.

Iterando per tutte le coppie x, y, si può compilare una tabella analoga alla *confusion matrix* dei modelli di Machine Learning. La misura Rand è

concettualmente identica all'accuratezza:

$$M = \frac{m(m-1)}{2} = a + b + c + d$$
$$R = \frac{a+d}{M}$$

Si definiscono anche l'indice di Jaccard:

$$J = \frac{a}{a+b+c}$$

l'indice di Fowlkes and Mallows e il Γ statistic.

5.2 Indici interni.

Sono indici che, in generale, servono a valutare la bontà dell'adattamento. Tipicamente, la misura di validità è valutata come $sum_{i=1}^{k} w_i \Delta validity(c_i)$. I pesi generalmente sono uguali a 1 $w_i = 1 \forall i$. La coesione C_i è calcolata come la sommatoria delle prossimità (o similitudine) tra tutte le osservazioni:

$$C_i = \sum \text{similarity}(x, y) \forall x, y \in C_i$$

La separazione invece è valutata tra coppie di cluster:

$$\text{separation}(C_i, C_j) = \sum \text{similarity}(x, y) \forall x \in C_i, y \in C_j$$

In caso di algoritmi che definiscono metoidi o centroidi, la coesione si calcola sommando le distanze tra le osservazioni e i loro metoidi (o centroidi); mentre la separazione si calcola, oltre che con la formula precedente, calcolando il metoide (o centroide) della distribuzione e calcolando la distanza tra il prototipo del cluster e quello della distribuzione. In base al valore dei pesi w_i, l'interpretazione del valore cambia:

- $w_i = \frac{1}{m_i}$: si ha una coesione graph-based;
- $w_i = 1$: si ha una coesione prototype-based;
- $w_i = m_i$: si ha una coesione ...

Queste misure possono essere usate per migliorare la qualità dei cluster: cluster non molto coesi possono essere divisi in più cluster; analogamente possono essere uniti cluster molto vicini tra di loro. Si definisce la silhouette di ogni osservazione come indice di qualità del cluster per la singola osservazione:

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

dove a_i è la distanza media della i-esima osservazione rispetto alle altre osservazioni del cluster, e b_i è la distanza media minima dell'i-esima osservazione rispetto alle osservazioni degli altri cluster. L'osservazione appartiene tanto più al suo cluster quanto più s_i tende a +1. Si calcola anche il valore di sihoulette del cluster facendo la media dei valori delle osservazioni, così come può essere calcolato un valore per la distribuzione.

Per cluster gerarchici, questo indice è privo di senso: bisognerebbe calcolare in relazione dell'intero albero; dunque è usato l'indice cophenetico, che misura il livello di prossimità tra coppie di punti appartenenti per la prima volta al medesimo cluster. È sempre compreso tra -1 e $+1$, e serve per indicare quale divisione sia la migliore.

5.3 Test d'ipotesi.

Esistono dei test d'ipotesi per verificare che la struttura dei cluster è dovuta al caso o ad una particolare disposizione dei dati. L'ipotesi nulla (H_0) è che i dati siano distribuiti a caso e quindi non siano classificabili in cluster. È scelto un indice valido per determinare la qualità della divisione in cluster della distribuzione e quindi calcolato per le osservazioni. Si ricava quindi la distribuzione dell'indice, se H_0 è vero (quindi le osservazioni sono disposte in modo casuale) tramite il metodo Monte Carlo o Bootstrap. Col metodo Montecarlo sono campionati i dati in modo casuale da una distribuzione uniforme e l'indice è calcolato per confrontare con l'indice dei dati di partenza. Ottenuta la distribuzione si calcola il quantile (ovvero il p-value) di ottenere l'esito di clustering data vera H_0: è possibile dunque determinare se il clustering costruito sia dovuto al caso o meno (rigettando quindi H_0).

5.4 Misure relative.

Questi indici tentano di stabilire quale sia il numero ottimale di cluster in cui dividere la distribuzione, senza effettuare test d'ipotesi. Una tecnica consiste nel ridurre lo spazio in uno spazio bi (o tri) - dimensionale per poi effettuare la divisione in modo arbitrario. Esistono anche indici numerici, quali l'indice di Calinski e Harabasz, l'indice di Dunn o l'indice Davis-Bouldin; inoltre esistono altre misure associa-

te ad algoritmi probabilistici quali l'AIC, l'MDL o il BIC.

Sono fissati un valore k_{min} e un valore k_{max}, quindi costruite delle sequenze di strutture di clustering (per un numero r di volte) ed è calcolato un indice per ogni divisione in cluster. È adottata la divisione in cluster il cui valore, nelle r iterazioni, ha raggiunto il valore ottimale dell'indice.

Parte III
Association Analysis.

L'analisi delle associazioni analizza le transizioni effettaute, ovvero insiemi di elementi di lunghezza variabile che corrispondono a operazioni unitarie effettuate dagli utenti. Un esempio può essere la lista della spesa degli acquirenti di un supermercato. L'obiettivo è trovare rapporti di conseguenza (antecedente e conseguente) che offrano dei vantaggi strategici. I dati sono generalmente rappresentati come variabili booleane dove ogni riga rappresenta una transazione e le colonne rappresentano gli *item* presenti nella transazione.

Si indica con $I = \{i_1, i_2, ..., i_d\}$ l'insieme degli *item* e con $T = \{t_1, t_2, ..., t_n\}$ l'insieme delle transazioni. Si definisce *itemset* un insieme di item (k-itemset se contiene k item diversi); una transazione contiene più itemset (di dimensioni diverse, eliminando o aggiungendo elementi). La larghezza della transazione rappresenta il numero di item della transazione.

Il *support count* $\sigma(X) = |\{t_i | X \subseteq t_i, t_i \in T\}|$ conta il numero di k-itemset per determinare regole di associazione $A \to B$, a patto che $A \cap B = \emptyset$.

Si dice *confidenza* il numero di volte in cui l'associazione si ripete rispetto al numero di volte in cui appare l'antecedente, in percentuale:

$$c(A \to B) = \frac{\sigma(A \cup B)}{\sigma(A)}$$

Si definisce invece *supporto* il numero in cui una regola è applicata nell'intero dataset, in percentuale:

$$s(A \to B) = \frac{\sigma(A \cup B)}{N}$$

L'ideale è trovare regole con un grande supporto (perchè facilmente trovabili) e alta confidenza (perchè attendibili).

Il supporto serve anche a eliminare regole che occorrono per puro caso, o regole poco applicabili e dunque non interessanti nel mondo reale, riducendo quindi il numero di regole considerate.

Per individuare il numero di regole individuate è sconsigliato usare il sistema *brute-force* dato l'elevato numero di regole (che segue una legge esponenziale). Il problema è quindi spezzato in due modi: eliminando tutte le regole che si è sicuri non soddisfino una

soglia minima di frequenza (perchè poco presenti nell'itemset) e poi sono considerate le regole solo per gli elementi rimanenti.

1 Generazione di *Frequent Itemset.*

Si generano tutti gli itemset candidati a partire dall'insieme vuoto iniziando a generare insiemi di un elemento, poi di due elementi (non ordinati) e così via fino ad arrivare ad un unico itemset comprendentemente l'insieme degli item; il numero di sottoinsiemi generati è pari a $2^d - 1$ (perchè è escluso l'insieme vuoto). Con l'approccio brute-force si contana il numero di volte che l'itemset compare all'interno delle transazioni, calcolando quindi il supporto; si procede poi per tutti gli itemset; il costo computazionale di questo sistema è pari a $\mathcal{O}(NMw)$.

1.1 Apriori.

Di fatto si usano altri approcci: grazie al principio *apriori*, si può inferire che se un itemset è frequente (cioè ha una frequenza minore del minimo) allora tutti gli itemset suoi sottoinsiemi sono frequenti (il numero di comparse che fa può solamente aumentare). Quindi, partendo da itemset frequente con una certa popolosità, si va a ritroso affermando che tutti i suoi sottoinsiemi sono frequenti. In realtà però il principio *apriori* si usa al contrario: si presuppone che se un itemset non è frequente, allora tutti gli itemset che partono da lì non sono a loro volta frequenti (il numero di comparse può solamente diminuire). L'idea fondamentale è che sono alternati passi di generazioni di candidati a tagli per ridurre la complessità del problema.

I parametri dell'algoritmo sono il *minsup*, la frequenza minima di comparsa di un elemento per poterlo definire come frequente: con un valore alto, l'algoritmo è meno restrittivo e incude più itemset (per numero e per lunghezza). Inoltre il numero di item Ω impone la dimensione massima dell'albero che si può costruire. Il numero di transazioni e la loro lunghezza media anche influiscono sul tempo di esecuzione dell'algoritmo.

Prendendo un k-itemset con frequenza Y, questo può generare $2^k - 2$ regole di associazione scambiando antecedente e conseguente. Le regole sono generate come $X \to Y - X$, a patto che X e Y siano insiemi non vuoti. Tutte le regole così generate rispettano il *support threshold* perchè già testato, bisogna però verificare il rapporto di conseguenza.

Un *Maximal Frequent Itemset* è un itemset frequente oltre al quale, se esteso, non genera itemset frequenti; è una sorta di frontiera. Definire un nodo come *maximal frequent itemset* non aiuta sul supporto dei suoi sottoinsiemi. Per aumentare l'informazione, si definiscono *closed frequent itemset* gli itemset chiusi (cioè se tutti i sotto-itemset hanno un supporto minore) e frequenti: è possibile risparmiare passaggi di computazione se sono eliminati i sottoinsiemi di un *closed frequent itemset*. Si evitano, cioè, regole associative ridondanti.

$$
\begin{gathered}
X \to Y \\
X' \to Y' \\
\text{se:} \\
X \subseteq X' \\
Y \subseteq Y' \\
s(X \to Y) = s(X' \to Y') \\
c(X \to Y) = c(X' \to Y')
\end{gathered}
$$