Machine Learning

Parte I
Introduzione

Il machine learning cerca, mediante tecniche di data mining e altre tecniche statistiche, patterns nei dati per rispondere alle domande. Può essere predittivo, descrittivo o di rinforzo (per la decisione di eventi che possono avere ripercussioni lontane nel futuro).

La parte di *preprocessing* trasforma i dati grezzi in un formato appropriato per l'analisi, raccogliendo i dati dalle varie sorgenti, rimuovendo il rumore e selezionando *records* e *features* per l'analisi. Successivamente è allenato un *learner* mediante algoritmi di reti neurali o bayesiane (o altre tecniche tra cui clustering). Infine nel *post-processing* si controlla l'affidabilità del risultato e sono proposte soluzioni ai problemi. Tutto il procedimento non si risolve in una sola istanza ma in più iterazioni dello stesso algoritmo con parametri diversi, o di algoritmi diversi.

I *dataset* possono essere file di testo semplice, fogli elettronici o tabelle di database relazionali. Sono quindi divisi in colonne che caratterizzano ogni singola istanza. Ogni dato è di un particolare tipo, in base al quale sono definite le operazioni lecite.

- Nominale: consente solamente l'operazione di uguaglianza, si calcolano moda, entropia e tabelle di contingenza;
- Ordinale: consente ordinamenti, si possono calcolare anche percentili, correlazione di Sperman e si possono applicare test non parametrici;
- Intervalli: consentono operazioni di addizione e sottrazione, si possono calcolare media, devianza, correlazione di Pearson e si possono usare i test F e T;
- Rapporti: consentono tutte e quattro le operazioni, si possono calcolare tutte le medie, i percentili e i coefficienti di variazione.

1 Analisi esplorativa.

I dati, in fase esplorativa, sono analizzati mediante indici per ottenere una sintesi degli attributi. Generalmente per attributi qualitativi si usa la moda (o una tabella di frequenze) mentre per variabili quantitative si usano i quantili (per stime non parametriche). Stime parametriche (come quelle basate sui momenti) sono influenzate dalla presenza di *outliers*, osservazioni così anomale da far pensare che vengano da un'altra distribuzione di dati: si usano anche stime parametriche corrette, come la *trimmed mean* che esclude i valori più alti e più bassi in equal numero.

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

$$\overline{x}_{\text{trimmed}} = \frac{1}{N-2k}\sum_{i=k}^{N-k} x_{(i)}$$

Alla varianza (σ^2) e al range, essendo influenzati da outliers, sono preferiti la AAD (*Absolute Average Deviation*) o la MAD (*Median Absolute*

Deviation).

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

$$AAD = \frac{1}{N}\sum_{i=1}^{N}|x_i - \overline{x}|$$

$$MAD = \text{median}(\underline{x} - \overline{x})$$

Altro indice non parametrico molto usato è l'IQR (*Inter Quartile Range*), la distanza tra i percentili di ordine 0.25 (*Lower Quartile*) e 0.75 (*Upper Quartile*). Alla matrice di varianza-covarianza, nel caso l'ordine di grandezza tra le variabili sia molto diverso, è preferita la matrice di correlazione lineare (di Pearson).

$$\sigma_{\underline{x},\underline{y}} = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \overline{x}) \cdot (y_i - \overline{y})$$

$$\varrho_{\underline{x},\underline{y}} = \frac{\sigma_{\underline{x},\underline{y}}}{\sqrt{\sigma_{\underline{x}} \cdot \sigma_{\underline{y}}}}$$

Per rappresentare graficamente la distribuzione, si usa spesso l'istogramma, mentre per verificare gli indici non parametrici si usa la scatola con baffi (BoxPlot), che in un colpo solo mostra IQR, range e valori anomali (i baffi hanno lunghezza massima pari a $1.5 \cdot IQR$ a partire dal proprio quartile, sono considerati *anomali* i valori ancora più lontani dal centro della distribuzione).

Parte II
Preprocessing

Di importanza fondamentale per la riduzione dei dati, il preprocessing permette di sfuggire alla *maledizione di dimensionalità* mediante l'uso di tecniche di campionamento, aggregazione di dati o semplificazione dello spazio delle variabili (minimizzando la perdita di informazione). Infatti, spesso i modelli di machine learning non aumentano le prestazioni in modo lineare all'aumentare della popolosità del dataset, richiedendo però maggiori risorse computazionali. Inoltre è possibile anche creare nuove variabili, a partire dai dati di partenza, che aggiungano informazione alla distribuzione.

1 Semplificazione dello spazio.

Dataset con molte variabili possono portare alla *maledizione della dimensionalità* (*Curse of Dimensionality*): le performance del modello non crescono in modo lineare con la numerosità del dataset, ma possono addirittura diminuire; questo perchè lo spazio delle variabili è molto frammentato e difficilmente mappabile (parti dello spazio rimangono vuote, per riempirle servirebbe un numero di osservazioni pari a n^k con $k \to \infty$). Ciò non si verificherebbe, a livello teorico, con un dataset infinito ma è un fenomeno diffuso in situazioni reali.

1.1 Riduzione del numero di features.

I sistemi più usati per ridurre la dimensionalità sono l'analisi delle componenti principali (PCA) e la decomposizione dei valori singoli (SVD): si tenta di massimizzare l'informazione contenuta dal dataset riducendo però la dimensione dello spazio. In questo modo tuttavia, il modello perde di interpretabilità.

Per semplificare variabili nominali invece, una tecnica è quella della binarizzazione: la scala di valori (da 0 a $k-1$, dove k indica il numero di valori che può assumere la variabile) è sostituita da una serie di variabili dummies che ordinate rappresentano una sequenza binaria. Il numero

di variabili dummies così è ridotto da $k-1$ a $\frac{k}{2}$. Questo metodo però può distorcere il dataset.

1.2 Raggruppamenti.

La *discretizzazione* si applica ad attributi continui per costruire intervalli in cui mappare i valori delle features. Paradossalmente, maggiore è il numero di intervalli utilizzati, migliore è il risultato: si tratta di una procedura degenere. La discretizzazione può essere fatta in caso di estrema ignoranza della distribuzione (*unsupervized*) o avendone già qualche conoscenza. Nel primo caso, si divide il range in un numero k arbitrario di intervalli, in modo tale che questi o siano di pari dimensione (metodo poco efficace in caso di outliers) o di ugual numero di elementi contenuti (si usano dunque i quantili per definire i limiti degli intervalli, più efficace ma più pesante computazionalmente). Conoscendo invece attributi correlati, è possibile discretizzare in base ad un'altra variabile nota, facendo sì che sia massimizzato un indice di purezza (come l'entropia E o l'indice di Gini).

$$e_i = -\sum_{i=1}^{k} p_{k_i} \cdot \log_2 p_{k_i}$$

$$E = \sum_{i=1}^{K} \frac{N_k}{N} e_i$$

Nel caso invece che attributi categorici assumano troppi valori, è possibile tentare di aggregarli per semplificare lo spazio: se possibile, si possono usare criteri noti in alternativa ai medesimi indici citati per la discretizzazione.

2 Trasformazione dello spazio.

Per aumentare di significato, le features possono essere trasformate secondo funzioni matematiche quali logaritmo o radice quadrata, o anche la normalizzazione *Z-score* ($Z = \frac{x-\mu}{\sigma}$). Quest'ultima funzione è utile per porre sullo stesso piano variabili con ordini di grandezza differenti.

3 Gestione dei valori mancanti.

Se i dati mancanti sono dovuti al caso, e il loro numero è esiguo, è facile escludere le osservazioni non complete; tuttavia se i valori mancanti sono dipendenti dai caratteri osservati, è preferibile stimare il valore reale mediante alcune tecniche. Generalmente si usa la moda per le variabili qualitative e la media (eventualmente condizionata ad un'altra variabile correlata) per i valori quantitativi, ma il sistema migliore è effettuare una regressione lineare basandosi sugli altri valori fino ad arrivare ad una convergenza (*most probable replacement*).

4 Campionamento.

Per ridurre la popolosità del dataset, il modo migliore è effettuare un campionamento. Un campione sufficientemente grande dovrebbe avere le stesse caratteristiche del dataset di partenza, ma richiede minori risorse computazionali per l'esecuzione degli algoritmi. Il campionamento può essere fatto secondo diverse modalità, ma dovrebbe tenere conto anche di eventuali disparità di valori all'interno degli attributi. Il campione deve essere sufficientemente grande da essere rappresentativo ma non così grande da risultare problematico per la computazione.

Parte III
Classificazione

Uno degli obiettivi del machine learning è quello di classificare i dati in categorie. Si individua dunque una variabile risposta con cui classificare le osservazioni e si costruisce un modello di classificazione che lo stimi su nuove osservazioni. Un *classificatore* esegue una classificazione supervisionata. Gli scopi possono essere principalmente descrittivi (se conta più l'interpretazione del risultato) o predittivi (se il modello è utilizzato per classificare nuove osservazioni). Il modello è costruito su una porzione del dataset, chiamato *train-set* (generalmente pari a 1/3 dei dati), e testato su una seconda partizione del dataset (chiamata *test-set*, comprendente il resto), per evitare che le performance siano deformate dal fenomeno dell'*overfitting*. Il trainset è dato in pasto ad un *learner* (l'algoritmo di classificazione) che restituisce in output il vettore della variabile risposta stimata. L'algoritmo non è l'unico possibile ma solamente una realizzazione di esso (perchè non sono noti i parametri reali della distribuzione ma si ha solamente un campione della popolazione): è una istanza (*inducer*), costruito col trainset e interrogato col testset. Si possono quindi costruire tabelle a doppia entrata (*confusion matrix*) per verificare la performance del learner:

	$Y=0$	$Y=1$
$\hat{Y}=0$	TN	FP
$\hat{Y}=1$	FN	TP

Si calcolano dunque quante osservazioni rientrano in ciascuna categoria: da questi valori si possono calcolare vari indici:

$$accuracy = \frac{TN+TP}{TN+TP+FN+FP}$$

$$error = 1 - accuracy$$

Generalmente, elevati livelli di accuratezza sono dovuti a errori nel modello.

Gli algoritmi di classificazione si basano su euristiche, reti neurali o calcoli probabilistici per classificare le osservazioni.

1 Albero di classificazione.

Considerato uno dei più semplici algoritmi, l'albero di classificazione mappa lo spazio partendo da un nodo radice e dividendo a ogni ramificazione la distribuzione massimizzazando un'euristica: la variabile della divisione e il valore di split sono calcolati in modo iterativo. Lo split può essere effettuato anche su più variabili contemporaneamente (albero obliquo). Il dataset è diviso in questo modo fino al raggiungimento di un nodo foglia, la cui distribuzione di frequenze è usata per effettuare la classificazione (le osservazioni sono classificate con la classe più frequente). Il modello non è efficace se tutti i nodi foglia classificano allo stesso modo.

2 Regressione logistica.

Le due classi sono rappresentate da un'unica variabile che può assumere i valori 0 (per la classe più rappresentata) o 1. Ipotizzando che tutte le variabili siano continue, si calcolano le probabilità:

$$P(y=0|\underline{X}) = \frac{1}{1+\exp\{\underline{w}\cdot\underline{x}\}}$$

$$P(y=1|\underline{X}) = \frac{\exp\{\underline{w}\cdot\underline{x}\}}{1+\exp\{\underline{w}\cdot\underline{x}\}}$$

Si calcola cioè la probabilità a posteriori che Y assuma un dato valore: l'osservazione è classificata con la classe maggiormente probabile o con quella che supera una certa soglia. La funzione logistica varia in base al vettore dei pesi $\underline{w}$.

3 *Support Vector Machine.*

Con più variabili continue è possibile costruire un iperpiano che divida la distribuzione in classi. Per evitare che fenomeni di overfitting rovinino il modello, si usa il concetto di *rischio* per

effettuare una stima che sbagli il meno possibile (si prende quindi la retta col margine di errore δ maggiore). Le osservazioni sono poi classificate con una semplice operazione di segno (cioè si verifica da che parte stanno nello spazio rispetto all'iperpiano). L'iperpiano migliore per la classificazione (ovvero quello che concede il maggior margine di errore) si ottiene con la formula:

$$\frac{1}{2}\underline{w} \cdot \underline{w}'$$

Può darsi che la retta non sia utile nella classificazione, si preferisce dunque definire una regione ammissibile:

$$y_i(\underline{w} \cdot x_i + b) \geq 1 \qquad \forall i \in [1; m]$$

È dunque sufficiente eseguire un'operazione lineare per ottenere i valori stimati della retta. Se l'insieme delle rette che dividono lo spazio è vuoto (cioè la distribuzione non è linearmente separabile), si modifica la funzione obiettivo per minimizzare la distanza dalla corretta classificazione ξ_i:

$$\min\{\frac{1}{2}\underline{w} \cdot \underline{w}' + \Delta \sum_{i=1}^{N} \xi_i\}$$

con:

$$\forall_{i=1}^{N} : y_i(\underline{w} \cdot \underline{x}_i + b) \geq 1 - \xi_i$$
$$\forall_{i=1}^{N} : \xi_i \geq 0$$

Inoltre è possibile usare superfici curve rimappando lo spazio tramite funzioni Φ e calcolando nuovamente l'iperpiano: per fare questo generalmente si applica una *Kernel Function* al vettore dei dati $\underline{x}$.

4 *Multilayer Perceptron.*

Le reti neurali del tipo *multilayer perceptron* sono composte da una serie di nodi di input e output collegati tra di loro così da simulare una rete di neuroni biologici. Ogni neurone j, eccettuando i neuroni di input, riceve in ingresso un valore x ed esegue una funzione f, a cui è aggiunto un parametro θ (datto soglia di attivazione).

$$y_j = f(\sum w_{i,j}x_i - \theta_j)$$

Storicamente, le funzioni di attivazione più usate erano la logistica o la iperbole-tangente, ma non essendo particolarmente efficaci in casi particolari si sono iniziate ad usare altre funzioni semplici, costanti fino alla soglia e poi crescenti in modo lineare.

$$f_{\text{Hyperbolic tangent}}(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

$$f_{\text{Logistic}}(x) = \frac{1}{1 + e^{-x}}$$

Il modello prevede l'inserimento di uno o più livelli di *hidden neurons*, strati nascosti in cui ogni neurone è collegato a tutti i neuroni di input e che rilasciano l'output a tutti i neuroni dello strato successivo. I nodi non sono *biorientati*, ovvero il segnale si propaga dall'input verso l'output senza tornare indietro, inoltre neuroni dello stesso livello non comunicano tra di loro; tuttavia il segnale può saltare uno o più strati di neuroni nascosti. L'algoritmo è particolarmente performante perchè il calcolo è parallelizzabile sui vari processori, che così possono operare in modo indipendente. Non esiste un sistema per stimare il numero di strati nascosti o il numero di neuroni per ogni strato: si procede per tentativi.

5 Classificatore Bayesiano.

Questo modello sfrutta il concetto di probabilità condizionata del Teorema di Bayes ponendo una relazione tra il vettore risposta $\underline{y}$ e la matrice del disegno $\underline{X}$. Il modello calcola la probabilità a posteriori di $Y = y$ quando la matrice del disegno è composta da variabili dicotomiche:

$$P(y|\underline{X}) = \frac{P(\underline{X}|y) \cdot P(y)}{P(\underline{X})}$$

Il modello permette solamente classificazioni binarie calcolando la probabilità a posteriori dell'appartenenza a una data classe. Il fatto che il denominatore della formula non sia noto non influisce sull'esito, dato che conta solamente il valore del numeratore (il denominatore è semplificato nei passaggi successivi). Il numero di osservazioni richieste dall'algoritmo cresce in modo esponenziale al numero di features utilizzate

(2^k), quindi è ipotizzata a priori l'indipendenza condizionata dei caratteri. In questo modo la probabilità a posteriori può essere scritta come una semplice produttoria:

$$P(y|\underline{X}) = \prod P(\underline{X}|y)$$

la complessità cresce in modo lineare; l'ipotesi è forte ma il modello è molto efficace soprattutto con dimensionalità elevate.

Il classificatore bayesiano riesce a gestire anche variabili numeriche calcolandone la probabilità grazie all'ipotesi di normalità.

Tree Aumented.

Nativamente, il modello gestisce solamente variabili binarie nella matrice del disegno, ma grazie a distribuzioni di probabilità (come la gaussiana o la Kernel function) è possibile calcolare la probabilità di realizzazione di variabili non binarie.

A causa dell'ipotesi di indipendenza condizionale, il modello è considerato molto rigido. Tuttavia, generalizzando il modello con un grafo dove sono collegate tra di loro le variabili dipendenti (evitando però cicli), si riesce a ottenere una stima migliore facendo crescere la complessità del problema tra il lineare (del caso in cui tutte le variabili siano indipendenti) e l'esponenziale (nel caso in cui tutte le variabili siano tra di loro dipendenti). Il grafo, tolta la variabile risposta, mostra una sequenza di nodi paragonabile ad un albero: ogni nodo dipende dalla variabile risposta e da un altro attributo.

Parte IV
Postprocessing

1 Prestazione del modello.

È fondamentale stimare la prestazione del modello con più di un sistema (non effettuando quindi una previsione puntuale, che può risultare troppo ottimistica o pessimistica). Gli errori nel modello possono essere di training (quando il training-set è classificato male) o di generalizzazione (overfitting dei dati); un buon modello dovrebbe avere più o meno le stesse prestazioni su entrambi i dataset. L'overfitting del training-set accade spesso quando il modello è eccessivamente prestante rispetto alle previsioni: il modello è troppo complesso per effettuare previsioni.

Un modo semplice ma efficace per misurare la prestazione del modello è calcolare la percentuale di classificazioni corrette:

$$accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

Inoltre bisogna tenere conto anche del tempo impiegato dall'algoritmo a classificare i dati (potrebbero perdere di valore se il tempo impiegato è particolarmente lungo) e il fabbisogno di risorse richieste alla macchina. Il modello è definito *robusto* se non soffre per la presenza di valori anomali o mancanti, *scalabile* se le prestazioni sono equiparabili all'aumentare del numero di osservazioni da classificare e *interpretabile* se è comprensibile da esperti di dominio per prendere decisioni.

1.1 Metodo *Holdout.*

Il dataset è diviso in due parti per controllare che non si verifichi overfitting: training-set (generalmente composto da 1/3 dei dati) e test-set (comprendente gli altri 2/3), mutualmente complementari ed esaustivi. Il test-set ha il compito di testare le prestazioni del modello nella classificazione di nuovi dati, tuttavia la misura è influenzata dal campione selezionato. Si usa quindi la tecnica dell'*iterate holdout*, che prevede che

per un numero R di volte si divida il dataset e si calcoli poi la media delle accuratezze. Nulla garantisce che però le osservazioni si trovino solamente in una partizione e non in un'altra, essendo questo un metodo basato sul campionamento casuale.

1.2 Metodo *Cross Validation*.

Per ovviare ai problemi dell'*iterate holdout*, si usa il sistema del *Cross Validation*: il dataset è ripartito in k partizioni mutualmente distinte ed esaustive; al k-esimo passo è utilizzata come training-set la k-esima partizione e come test-set la rimanente parte delle osservazioni; si calcola infine la media delle prestazioni. Generalmente si divide il dataset in partizioni uguali per evitare che abbiano poi pesi diversi nel calcolo della media. Valori tipici di k sono 3, 5 o 10, ma esiste anche il *Leave One Out Cross Validation* che usa $k = N$ (è un sistema però molto pesante computazionalmente).

1.3 Confronto tra modelli.

L'accuratezza è modellizzabile come una variabile aleatoria binomiale:

$$\mu = p$$

$$\sigma^2 = \frac{p(1-p)}{N}$$

È dunque possibile costruire intervalli di confidenza per la previsione del valore reale di accuratezza del modello. Grazie all'approssimazione alla normale della variabile binomiale (*T. del Limite Centrale*), è possibile confrontare modelli tra loro diversi: si calcola la distanza $d = e_1 - e_2$ che si distribuisce come:

$$d \sim N(e_1 - e_2, \frac{e_1(1-e_1)}{n_1} + \frac{e_2(1-e_2)}{n_2})$$

Se l'intervallo di confidenza contiene il valore 0, la distanza non è statisticamente significativa da dimostrare che un modello sia maggiormente prestante di un altro.

1.4 Variabili sbilanciate.

Se la variabile risposta è molto sbilanciata, in linea teorica un modello che classifica tutte le osservazioni con la classe più frequente appare molto prestante. Oltre ad attuare campionamenti stratificati o *bootstrap*, si usano indici alternativi per misurare la prestazione:

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

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

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

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

Il primo indice (*precision*) misura il numero di positivi reali sul numero di positivi totali, mentre il secondo (*recall*) il numero di positivi catalogati tali. I due valori generalmente sono complementari. Per calcolare la prestazione del modello, si calcola la media armonica dei due valori:

$$F_i = 2\frac{r \cdot p}{r + p}$$

Oppure, in alternativa:

$$F_\beta = (\beta^2 + 1)\frac{r \cdot p}{r + p\beta^2}$$

$$\lim_{\beta \to 0} F_\beta = p$$
$$\lim_{\beta \to \infty} F_\beta = r$$

1.5 Costi.

In situazioni reali, non tutti i positivi possono essere gestiti come tali (generalmente a causa del costo della decisione): il modello usato è l'SMP, che tenta di minimizzare il costo dati i costi per ogni classificazione (matrice di costo). Il modello decisionale deve dunque avere un'accuratezza elevata e un costo ragionevole. I costi generalmente non sono noti: la matrice di costo è facilmente criticabile, ma è sensato supporre che il

modello attui in modo corretto per ogni matrice dei costi sensata.

In altri casi, il numero di azioni che si possono intraprendere è minore del numero di osservazioni classificate come positive: si selezionano dunque solamente le N_1 osservazioni che il modello ha stimato come più probabilmente positive. La classificazione in questo caso deve tenere conto anche del costo delle decisioni attuate in funzione dell'output del modello: il modello non deve solamente essere preciso ma deve anche garantire che non ci siano sprechi a livello economico.

Si possono fare confronti tra più modelli diversi, tenendo conto del costo, grazie al *cumulative gain*: ordinando le osservazioni secondo la probabilità di appartenenza alla classe risposta, si mette in relazione il numero di casi realmente positivi in percentuale sul totale ($\frac{\text{TP}}{\text{TP+FN}}$) rispetto alla percentuale delle osservazioni considerate sul totale. In questi modo si possono confrontare modelli diversi applicati a dataset di dimensioni diverse.

1.6 Curva ROC.

Sistema grafico per verificare la bontà di un modello è usare la curva ROC e l'indice AIC (*Area Under Curve*): ordinate le osservazioni per la probabilità di appartenenza alla classe risposta, si calcola la percentuale di TP sull'asse delle ordinate sulla percentuale di TN sull'asse delle ascisse. L'assenza di modello equivale alla bisettrice del primo e terzo quadrante ($y = x$). Più una curva ROC tende verso l'alto, maggiore sarà la bontà di classificazione: questa curva sale infatti all'aumentare dei positivi correttamente classificati e si sposta verso destra o a causa di un falso positivo o quando sono state registrate tutte le osservazioni positive. Un modello ottimo infatti avrebbe una curva ROC che prima sale in verticale fino a raggiungere le coordinate $(0, 1)$ e poi si sposta verso destra fino a raggiungere $((1, 1)$: in questo caso l'indice AIC sarebbe pari a 1, il suo valore massimo.

2 Gestione delle *features.*

2.1 Selezione delle *features.*

Un alto numero di features riduce l'interpretabilità del modello e consente una maggiore efficienza; rimuovere troppe variabili però può provocare perdita 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).

2.2 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.

3 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.

4 Regolarizzazione.

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

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

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.

5 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 $\underline{X}$ pesata con un vettore $\underline{w}$. 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.

5.1 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 $\underline{X}$. 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 $> \alpha$.

5.2 Valutazione del modello.

Il coefficiente R^2 di determinazione rappresenta la percentuale di varianza totale spiegata dal modello:

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

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 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 $E[\underline{y}|\underline{X}]$, 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 V
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 sottocluster 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. Generalmente con una delle due misure si può calcolare anche l'altra se la scala è lineare; sono previste tuttavia anche scale non lineari, ottenibili tramite formule come $d' = \frac{d}{d+1}$.

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

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

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.

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 $n \to \infty$ 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:

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

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.

Misura la percentuale di match in due osservazioni, se la caratteristica è presente:

$$J(A, B) = \frac{\#\text{ matching attributes}}{\#\text{ at least in one}}$$

2.4 *Cosine similarity.*

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

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

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.

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

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 ko 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 Metodi gerarchici.

I *metodi gerarchici* sfruttano la matrice di distanza per dividere in modo agglomerativo o divisivo le osservazioni (costruendo un *dendogramma*). La prima famiglia di algoritmi inizia ponendo ogni osservazione in un cluster distinto e unendo a ogni iterazione i due cluster più vicini, secondo un criterio; al contrario gli algoritmi divisivi, poco usati per l'elevato fabbisogno di risorse, collocano tutte le osservazioni in un unico cluster per poi dividere le osservazioni a ogni iterazione. Ponendo $\underline{d} = \{d(a, b)\ \forall a \in A, b \in B\}$ (dove A e B sono due cluster arbitrari), i criteri adottati per verificare l'inclusione o l'esclusione delle osservazioni sono:

- distanza minima: $d(A, B) = min\{\underline{d}\}$
- distanza massima: $d(A, B = max\{\underline{d}\}$
- distanza media: $d(A, B) = \overline{d}$

I cluster possono dunque essere identificati in base al numero o alla distanza tra di loro; tuttavia rispetto ad altri algoritmi di clustering questi metodi richiedono molte risorse.

3.2 *Density Based.*

L'algoritmo DB-Scan individua cluster di dimensione e forma variabile in base alla densità delle osservazioni nello spazio: i cluster sono definiti come zone altamente dense circondate da zone prive di osservazioni. Si definisce una distanza massima Eps entro cui l'algoritmo verifica la quantità di osservazioni (i *vicini*) per calcolare la densità: in base al valore (superiore o meno ad una certa soglia $MinPts$) si definiscono:

- *core points*: le osservazioni con un numero di vicini $\geq MinPts$;
- *border points*: osservazioni con un numero di vicini $< MinPts$ ma che rientrano nel vicinato di almeno un *core point*;
- *noise points*: sono *border points* che non rientrano nel vicinato di nessun *core point*.

A questo punto sono eliminati i *noise points* e formati i cluster raggruppando *core points* vicini tra di loro (si considerano *vicini* due *core points* se uno è nel vicinato dell'altro); i *border points* infine sono assegnati al cluster più vicino.

3.3 *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.

3.4 *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 ia, 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.

3.5 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:

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

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.

3.6 *Kohonen MAPS* e *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.

5 Validazione.

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 diffe-

renti 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:

(a) x e y appartengono allo stesso cluster e alla stessa categoria a priori;

(b) x e y appartengono allo stesso cluster ma a diverse categorie a priori;

(c) x e y appartengono a diversi cluster ma alla stessa categoria a priori;

(d) 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 $J \in [0; 1]$:

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

l'indice di Fowlkes and Mallow $FM \in [0; 1]$

$$FM = \sqrt{\frac{a}{a+b} \cdot \frac{a}{a+c}}$$

e il Γ statistic $\Gamma \in [-1; +1]$

$$\Gamma = \frac{M \cdot a - (a+b)(a+c)}{\sqrt{(a+b)(a+c) \cdot (M-a-b)(M-a-c)}}$$

5.2 Indici interni.

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

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

La separazione invece è valutata tra coppie di cluster:

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

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.

$$\text{cohesion}(C_i) = \sum_{x \in C_i} \text{similarity}(x, c_i)$$

$$\text{separation}(C_i) = \text{similarity}(c_i, \underline{x})$$

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 separazione prototype-based.

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 associate 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 VI
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 $A \cap B = \varnothing$.

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 $\mathcal{O}(NMw)$.

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 sottoitemset 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.

$$X \to Y$$
$$X' \to Y'$$

se:

$$X \subseteq X'$$
$$Y \subseteq Y'$$
$$s(X \to Y) = s(X' \to Y')$$
$$c(X \to Y) = c(X' \to Y')$$

2 Valutazione.

La formulazione di regole non deve essere troppo semplice e deve interessare gli esperti di dominio. Inoltre esistono misure oggettive, indipendenti dal dominio, per calcolare la qualità delle regole formulate: si costruisce una matrice di confusione alla pari della classificazione. Si calcolano quindi le frequenze marginali della tabella per verificare che la regola aggiunga informazione: la confidenza è calcolata in relazione al supporto del conseguente. Infatti l'antecedente è già verificato dall'algoritmo di formulazione, ma solamente regole con un conseguente frequente possono aggiungere informazione. Si calcola quindi il *fattore di interesse*:

$$Lift = \frac{c(A \to B)}{s(B)}$$

$$I(A,B) = \frac{s(A,B)}{s(A) \cdot s(B)} = N \frac{f_{1.1}}{f_{1.\cdot} \cdot f_{\cdot.1}}$$

che assume il valore 1 in caso di indipendenza tra i due item.

Si calcola anche il coefficiente di correlazione:

$$\Phi = \frac{f_{1.1} \cdot f_{0.0} - f_{0.1} \cdot f_{1.0}}{\sqrt{f_{1.\cdot} \cdot f_{\cdot.1} \cdot f_{0.\cdot} \cdot f_{\cdot.0}}}$$

o l'IS Measure:

$$IS(A,B) = \sqrt{I(A,B) \cdot s(A,B)} = \frac{S(A,B)}{\sqrt{s(A) \cdot s(B)}}$$

che è una buona misura per verificare l'associazione di parole all'interno di testi; è semplificabile come $\sqrt{s(A) \cdot s(B)}$ in caso di indipendenza tra le due variabili.

Una misura si dice *simmetrica* se scambiando antecedente e conseguente il suo valore non cambia, *asimmetrica* altrimenti. Esistono comunque una quarantina di misure diverse per calcolare il valore di una regola, e la letteratura suggerisce anche delle particolari misure per ogni dominio. Comunque alcuni attributi sono più adatti a misurare la prestazione in caso di regole simmetriche o asimmetriche.

Non tutte le misure però sono (facilmente) utilizzabili se la regola comprende più di due elementi. Infatti, si possono formulare regole anche con più di un antecedente: il concetto di associazione può essere valutato anche in modo congiunto con altre variabili. Il *paradosso di Simpson* afferma che gli effetti di un algoritmo appaiono o scompaiono in base al raggruppamento effettuato sulle osservazioni. Si deve quindi verificare che gli effetti non siano particolarmente congiunti ad una variabile quando è effettuata la stratificazione del campione, per evitare la formulazione di *pattern spuri*.