Dispensa di Machine Learning

Federico Luzzi

1 Lezione 1

1.1 Introduzione

Gli ambiti più importanti nei quali vengono applicate tecniche di machine learning nella vita reale sono:

- Finanza
- Sanità
- Agricoltura
- e-commerce
- Social
- Chatbot
- Sensoristica (come i veicoli a guida autonoma)

Vista la grande mole di dati la necessità è capire come trattare i dati. **L'obiettivo del Machine Learning è sviluppare metodologia per dare valore ai dati in funzione di una particolare domanda che ci stiamo facendo.**

Tipicamente si divide in tre macro categorie.

- Apprendimento supervisionato o predittivo: qualcuno ha gia catalogato ad esempio delle immagini o dei dati e noi prendendo questi modelli dovremmo essere in grado di predire.
- Apprendimento descrittivo: ci sono delle funzioni obiettivo che vanno ottimizzate. Non usiamo etichette della singola istanza ma in qualche modo sappiamo dove arrivare.
- Apprendimento rinforzato: E' quello più utile in questa epoca: funziona sui premi.

In questo corso ci concentreremo sui primi due tipi di apprendimento.
L'apprendimento supervisionato si divide a sua volta:

- Classificazione: quando sono quantità discrete da dividere
- Regressione: con una data lettura cerco di prevedere dei dati.

L'apprendimento non supervisionato si divide a sua volta:

- Clustering: Vuol dire mettere ordine nelle istanze che ci vengono presentate.
- Associativa: Scopre pattern che descrivono bene caratteristiche associate ad un certo fenomeno.

Per alcuni compiti la correlazione va benissimo, in alcuni casi però addirittura ci danneggia. Se provassi a vedere la correlazione tra il numero di omicidi in america e il numero di fondi investiti sulla ricerca scientifica vedrei che statisticamente sono strettamente correlate. Questo è ovviamente un no-sense.

Paradosso di Simpson: Se uso solo i dati senza modello no c'è alcun modo di scoprire la verità

1.2 Data Types

Il primo passo fondamentale è sicuramente quello di prendere confidenza coi dati. E' fondamentale capire la natura dei dati che abbiamo a disposizione (**dataset**), c'è un fenomeno che si chiama **churn**, quando non siamo soddisfatti di un servizio ci affidiamo al competitor.

Ci possono essere valori missing in un dataset e possono mancare per diversi motivi.

Le colonne sono chiamate **Attributi.**

Le righe sono chiamate **Istanze.**

A volte ci sono anche attributi duplicati che possiamo tranquillamente buttare fuori. Ogni attributo è caratterizzato dal fatto di avere un tipo. Conoscerlo è fondamentale per trattare i dati. Gli attributi si dividono in due grandi gruppi:

- Categorici:
  - Nominali: ad esempio il colore degli occhi.
  - Ordinali: ad esempio possono essere i giudizi.
- Numerici:
  - Intervallo: ammettono operazioni di somma e sottrazione.
  - Ratio: possiamo applicare tutte le operazioni logico/matematiche.

Dall'alto al basso il livello gerarchico sale e le proprietà aumentano.

Si possono anche dividere in attributi *discreti* che possono essere:

- Categorici

- Numerici
- Binari: sono i più particolari da trattare, e hanno una serie di proprietà strane.

Oppure possono essere *Continui.*

1.3 Data exploration

Dobbiamo però anche sapere come esplorare i dati. Per farlo facciamo cose molto elementari. Per farlo si usano tutti gli strumenti a nostra disposizione.

Il concetto di **quantile** è trovare il numero di osservazione che ci indica quanti attributi sono più piccoli di un dato valore. Un quantile molto importante è il quantile di ordine $\frac{1}{2}$ e si chiama **Mediana** che è quello che ha esattamente minori di lui la metà dei dati.

$$mean = \frac{1}{n}\sum_{i=1}^{n} x_i$$

La media non è un buon modo di visualizzare i dati perché dice poco ma quanto meno dice qualcosa. Siccome la media è basata su singole osservazioni si possono vedere le presenze di outliar, ossia di elementi troppo discordanti dalla media e che si presenta poche volte. Sono quindi elementi che è necessario trattare per vedere la provenienza.

Per prevenire questo si usa la **media trimmed** in cui si buttano via il valore più piccolo e il valore più grande. Se si trova un grosso scostamento probabilmente è presente un outlier.

Si può definire anche il range anche se di solito si usa la varianza:

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

Di solito si usa la deviazione standard che ha lo stesso ordine di grandezza dei dati:

$$std = \sqrt{var}$$

Si usa anche il range interquartile (IQR) sempre per ovviare alla presenza di outliar. Se ho a che fare con coppie di attributi allora è naturale parlare di **covarianza** ossia la varianza calcolata su due attributi diversi:

$$cov(X, Y) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})$$

Di solito si fanno scalature perché sennò le covarianze vengono troppo sbagliate. Per ovviare uso la **correlazione di Pearson** che può prendere valori $[-1, 1]$

$$corr(x, y) = \frac{cov(x, y)}{\sqrt{var(x)var(y)}}$$

Possono organizzare i dati in istogrammi in cui posso usare una ampiezza fissa o variabile (bin). A seconda dell'ampiezza che uso posso ottenere due disegni molto diversi.

Un altro modo di rappresentare i dati particolarmente utile è il **grafico Box and Whiskers** applicato solo ad attributi quantitativi.

1.4 Missing replacement

E' un problema enorme di per sè. Le operazioni più elementari sono le seguenti. In alcuni valori degli attributi un valore non è registrato. Ci possono esser tante ragioni: ad esempio un attributo non è sempre stato osservabile (penso all'ambito clinico), o ad esempio un attributo prima non veniva considerato rilevante.

Il primo nmetodo è il **Record removal** che è molto drastico come metodo perché comunque sia vengono eliminati dei valori che sarebbero potuti essere molto importanti.

Il secondo metodo è quello di **imputazione manuale**: è fatta da umani e tramite osservazioni ci si chiede se sia possibile inserirlo, è tremendamente difficile dal punto di vista computazionale.

Il terzo metodo è quello della **global constant**: ossia metto un numero là chiamato place holder con un valore costante, non troppo efficiente.

Il quarto metodo è quello di **rimpiazzarlo con la moda**, anche questo però è fortemente criticabile. Se gli attributi sono continui si fa la stessa cosa ma con la media.

Il quinto metodo è **Conditional mean replacement** ossia bisogna rimpiazzare con la media solo se è presente un altro determinato attributo.

Il sesto metodo è quello del **most probable** ossia di prendere un modello e sostituire il valore.

E' molto difficile dare il confine tra l'esplorazione dei dati e la modellizzazione dei dati.

1.5 Data Preprocessing

Tutti questi processi impattano fortemente tutta l'analisi che farò dopo.

2 Lezione 2

Da fare

3 Lezione 3

3.1 Classification

Entriamo un po' più nella componente di validazione nei modelli di classificazione supervisionata. La cosa più importante è **capire cosa stiamo facendo.**

In questa lezione parleremo di:

- Variabile di classe
- Modellizzazione descrittiva
- Training set/ test set
- Modelli di classificazione
- Matrice di confusione
- Accuratezza

Nel solito dataset vogliamo capire quale sia la particolare combinazione di attributi che abbiamo che determini se un cliente abbandoni il nostro servizio o meno. Bisogna prima vedere di che tipo siano le variabili presenti nel nostro database.

Dobbiamo essere in grado sia di prevederlo che di capire perché qualcuno ha abbandonato. Svolgiamo questo compito con un modello di **classificazione**. Un modello di classificazione è un modello che sfrutta alcuni attributi del dataset per prevedere un valore di un altro attributo. Possiamo vederlo come:

una scatola che ha degli ingressi che *sono gli attributi che usiamo per svolgere il nostro compito* e ha degli output che corrisponde alla *variabile da categorizzare.*

Le variabili in input si chiamano **variabili esplicative o variabili di input.**

Le variabili in output si chiamano **variabili di classe o variabili di output.**

Un **modello di classificazione** è qualcosa che risolve un problema di classificazione:

- Modello descrittivo: serve come strumento di spiegazione per distinguere tra oggetti di classi diverse
- Modello predittivo: predice la classe di un record sconosciuto, può essere visto come una scatola nera che asssegna una label di una classe al record sconosciuto

La tecnica di classificazione o **classificatore** è un approccio sistematico per costruire un modello di classificazione su un dataset.

Prenderemo quindi un **training set** che consiste in dei record in cui tutti gli attributi sono noti. Una volta addestrato (apprendimento e validazione) il nostro modello questo verrà valutato su un insieme di dati chiamato test set.

Il **test set** consiste in dei record le cui label di classe (o valori) sono sconosciute (o presunte tali).

Il modello che ne esce dal training con il training set si basa su un certo algorimo di apprendimento (**learner**), è questo la chiave di tutto.
Io sono sempre interessato a ciò che deve predire più che a quelli che ho già. Il training set è quello principale su cui il modello impara da là passo alla feature selection perché mi accorgo che magari qualche attributo non è necessario nelle predizioni ma fa solo rumore. In particolare è l'algoritmo di apprendimento che dice come il modello impara.

Il modello trainato si chiama **inducer**, in pratica un'istanza del modello di classificazione. All'inducer viene chiesto di predire il valore dell'attributo di classe per il test set.

Bisogna valutare ora le performance del modello inducer. Uno dei modi per analizzare se è venuto bene devo usare la **matrice di confusione.** Nelle righe vi sono i veri valori delle classi, nelle colonne i valori predetti dall'inducer. Il numero di righe è uguale al numero di colonne e corrispondono al numero di classi.

		INDUCER PREDICTION (IP)	
		-1	+1
ACTUAL CLASS (AC)	-1	TN	FP
	+1	FN	TP

Figura 1: modello di matrice di confusione

Definiamo una misura che si chiama accuratezza:

$$accuracy = \frac{\sum_{i=1}^{n} diagonal}{\sum_{i=1}^{n'} elements} = \frac{TN + TP}{TN + TP + FN + FP}$$

La misura complementare che troveremo indicata è quella degli elementi:

$$error = 1 - accuracy$$

3.2 Tecniche di classificazione

Una tecnica di classificazione è un modo sistematico di aggredire un dataset, in particolare possiamo dividerle in 4 macro categorie:

- Euristici: ispezionano il suo vicinato come (Decision Trees, Random Forest, Nearest Neighboor)
- Regression Based: usa la probabilità condizionata parametrica, regressione logica
- Separazione: partiziona lo spazio degli attributi, fa riferimento alle Support Vector Machine e alle Artificial Neural Network
- Probabilistici: usano la formula di Bayes (Naive Bayes ecc...) e si dimostra particolarmente efficiente.

Forniremo una overview di tutte queste, non le tratteremo nel dettaglio

3.3 Euristici

Un **albero di decisione** ha innanzitutto una rappresentazione grafica che ci consente di approcciarci con più propensione. Vi sono 2 elementi: nodi e archi. Il nodo rappresenta un sottoinsieme del dataset, gli archi sono usati per modellare gli output di modelli diversi di dataset.

I suoi ingredienti pricnipali sono:

- Nodo a radice: non ha archi in ingresso ma può averne più di due in uscita.
- Nodi interni: hanno un solo arco in ingresso e alemeno due in uscita.
- Nodi Foglia: sono nodi interni senza archi in uscita.

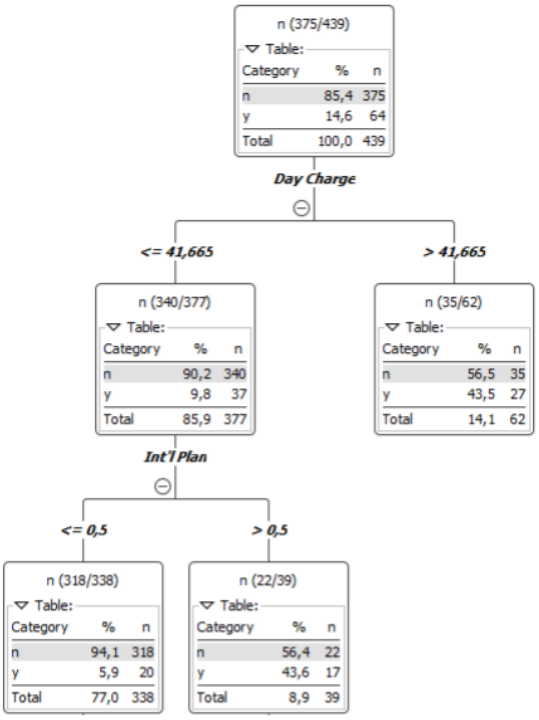

Figura 2: modello decision tree

Cerchiamo di capire come si forma questo albero: ad ogni nodo è associato un determinato schema.

Subito dopo il primo nodo (radice) abbiamo l'indicazione di un attributo: ad esempio posso chiedermi se 'day charge' sia minore di 41 o maggiore e dividere l'albero in due nodi. Se la risposta è vero allora mi muovo in un nodo altrimenti nell'altro.

All'interno del nuovo nodo ho di nuovo la valutazione di una variabile, e proseguo così finchè raggiungo un nodo foglia.

Quando arriviamo ad una foglia devo fornire una risposta, ovvero conto la classe più frequente ed rispondo al chiamante con essa (es. churn = n).

Per la valutazione delle **performance** vengono utilizzati i seguenti indici: entropia, indice di Gini e il classification error.

Inoltre va notato che il **tipo** di attributo utilizzabile può essere: binario, nominale o continuo.

E' un modello che risponde sempre la classe più frequente ed è un modello praticamente inutile quando ho delle classi particolarmente sbilanciate. Sono evidentemente test univariati, ciò che faccio è costruire iper parallelepipedi del nostro dataset.

Il risultato di queste rette fa la differenza sulla capacità di evidenziare dove ci siano elementi di una certa classe, devo quindi imparare a posizionare questi iperpiani: **Voglio partizionare il mio spazio in iperpiani massimizzando l'accuratezza.**

Non c'è alcuna ragione per cui io non possa usare degli splitting multipli e non solo su binari, posso farlo anche su valori nominali.

I segmenti degli iperpiani si chiamano **decision boundary** e sono quelle che passano dalle classi, il confine tra diverse aree della separazione delle classe. È un metodo computazionalmente dispendioso se ho molte variabili da valutare.

3.4 Regressione Logistica Binomiale

Servono per risolvere problemi di regressione binaria a diversi livelli. E' applicabile ad attributi continui e con certe accuratezze anche ad attributi nominali.

Si assume l'attributo di classe $Y = \{0, 1\}$, allora il classificatore a regressione logistica binomiale calcola a posteriore la probabilità che Y assuma il valore di un input esplicativo X. Si cacola come segue:

$$P(Y = 0|X = \bar{x}) = \frac{1}{1 + e^{\bar{x}\bar{w}}}$$

$$P(Y = 1|X = \bar{x}) = \frac{e^{\bar{x}\bar{w}}}{1 + e^{\bar{x}\bar{w}}}$$

dove $\bar{w}$ è chiamato il vettore parametro.

In questo modo vengono portate le probabilità di appartenere ad una certa classe.

4 Lezione 4

Cross-validazione: quando facciamo l'apprendimento, se ci sono parametri per ottimizzare il modello è buona norma provare diverse combinazioni di parametri in fase di apprendimento. (detto informalmente)

Random forest: è un comitato di alberi di decisione. Sostanzialmente usa degli attributi che sono in generale sottoinsiemi degli attributi, ogni albero può avere un sottoinsieme differente di attributi. Ogni albero usa attributi in modo randomico. Ognumo apprende a modo suo e il random forest in base a dei parametri (regione dello spazio) decide a quale albero dare retta.

4.1 Support Vector Machines

Metodi di classificazione con separazione. Lo scopo è separare o "apprendere" classi che vogliamo classificare. In uno spazio bidimensionale possiamo tracciare una retta (se ho due sole classi) per cui definisco l'area di appartenenza di una o dell'altra classe. La retta in questione è definita dalla seguente equazione:

$\bar{w} \cdot \bar{x} + b = w_1 x_1 + w_2 x_2 + b = 0$

per orientare la retta utiliziamo il vettore $\bar{w} = [w_1, w_2]$ oppure b. ($\bar{w}$ fa ruotare, b fa traslare)

se la retta esiste allora l'insieme delle istanze è linearmente separabile.

Problema: vi sono più (anche infinite) rette utilizzabili per effettuare la separazione, quale devo scegliere? Perchè non basta trovare una retta che funziona, ma la retta migliore per quando si dovrà valutare dati nuovi. Si crea una sorta di area grigia nlla quale non so dire esattamente quale delle due classi categorizzare.

inserisci immagine slide

devo sostanzialmente trovare una retta che **massimizza il margine di errore**. L'optimal linear decision boundary.

inserisci altra immagine slide (dove vedo il margine di errore)

la retta B_1 è nettamente preferita rispetto alla retta B_2.

Naturalmente per n attributi bisogna trovare l'iperpiano ottimale per dividere un determinato insieme di istanze. Matematicamente parlando si cerca di massimizzare il margine $\delta = 1/|w|^2$, la retta ha la seguente impostazione: $\bar{w} \cdot \bar{x} + b = 0$, le rette ai confini del margine sono fissate a $\bar{w} \cdot \bar{x} + b = 1$ e $\bar{w} \cdot \bar{x} + b = -1$.

$h(\bar{x}) =$

L'argomento della retta è in 2 dimensioni, però dopo aver applicato la funzione h(x) quella retta diventa un piano che va a -1 da un lato e a 1 dall'altro.

$$min \frac{1}{2} \bar{w} \cdot \bar{w}^T$$

Per trovare la retta devo minimizzare l'inverso del margine δ

bisogna imporvi dei vincoli per ogni attributo, se $\bar{w}$ e $\bar{x}$ sono *concordi in segno* (positivo o negativo) allora classifica perfettamente perchè il risultato è $>= 1$. Garantiscono che tutti i casi del dataset siano classificati correttamente e tra tutti i casi che classificano correttamente scelgo quello che massimizza il margine.

Non risolveremo questa formula di ottimizzazione in modo diretto, ma la sua formulazione **duale**. In ogni caso questa formulazione funziona bene se il problema è linearmente separabile.

Nei casi **non** linearmente separabili, non esiste **mai** una retta in grado di separare correttamente le classi. In questi casi la formulazione precedente non ammette soluzione, perché per alcuni dati i vincoli non ammettono soluzione.

Si introducono allora le **Linear Soft-margin**:

min

il chi deve essere non negativo (variabile di slack), se utilizzo questo parametro di una certa quantità per ammettere un errore allora esiste almeno una retta che risolve il problema di ottimizzazione. Ho sostanzialmente *rilassato* il problema di ottimizzazione, in particolare i vincoli. Graficamente parlando è come traslare degli elementi di una classe diversa dalla regione di appartenenza verso la regione della classe di quell'elemento.

NB: i vettori di supporto sono quelle osservazioni che sono sul bordo del margine

Altra soluzione fattibile è utilizzare una funzione **non lineare**. Si va a cercare una trasformazione che porti dallo spazio originale in uno spazio delle features in cui posso applicare una separazione lineare. Sono in grado di separare il nuovo dataset nello spazio delle features. In questo modo posso sfruttare tutta la metodologia precedente ma in uno spazio "controllato".

la traslazione avviene attraverso una funzione ϕ: $\bar{w} \cdot \phi(\bar{x}) + b = 0$. Il modello diventa:

min ...

per l'apprendimento utilizziamo sostanzialmente delle funzioni kernel: $K(\bar{u}, \bar{v}) = \phi(\bar{u}) \cdot \phi(\bar{v})$. Queste funzioni kernel sono delle funzioni di similarità calcolate nello spazio attributi originale di x.

4.2 Reti di neuroni artificiali

Oggi si utilizzano molto per il deep learning.

Un modello **MLP Multi-layer perceptron** consiste in neuroni artificiali che comunicano in modo unidimensionale dall'input X alla variabile di classe (volendo ci possono essere più neuroni di ouput).

Dati 3 neuroni di parametri continui, vengono inseriti in un neurone 4 che calcola una funzione. Ogni input nel neurone ha associato in peso w. Il neurone calcola una combinazione lineare tra gli input ed il peso dato ad esso.

$z_4 = w_{1,4} \cdot x_1 + w_{2,4} \cdot x_2 + w_{3,4} \cdot x_3$

$z_4 - \theta_4$ threshold di soglia applicato alla combinazione lineare

$f(z_4 - \theta_4)$ funzione di attivazione (o trasferimento) che restituisce il valore che lui trasferirà ad un altro neurone o strato con cui comunica. Storicamente le funzioni applicazione sono la tangente iperbolica e la funzione logistica (intervallo tra -1 e 1 e tra 0 e 1). Oggi vengono utilizzate delle funzioni non derivabili più complesse, come funzioni RELU (Retify Linear Unit) che assumono valore 0 nel semiasse negativo e 1 nel semiasse positivo.

Vi sono 3 tipi di neuroni: di input, di output e nascosti. Quelli di input comunicano con quelli nascosti (i cui valori non sono palesi a noi) e comunicano infine con quello/i di output.

Ogni neurone di input è connesso con ogni neuroni di strato nascosto ed ogni nodo di strato nascosto comunica con il nodo di output (rete fully-connected). Ogni arco ha associato un peso e ogni nodo ha un valore di soglia θ_j e una funzione di attivazione.

Le scelte architetturali fanno la differenza nel risultato della computazione. Doppia scelta: quanti neuoni usiamo e quanti ne utilizziamo nello strato nascosto. Determina l'architettura della nostra rete. Posso pensare di aggiungere un **livello** di neuroni nascosti. Non vi sono teorie che dicono il numero di nodi e livelli di strati nascosti, si procede in modo empirico e si valuta il modello che funziona meglio. Non vi sono vincoli rispetto a comunicare saltando livelli. Non si può però comunicare all'indietro, infatti queste reti sono chiamate **feed-forward neural network** (possono possedere fino a centinaia di strati nascosti).

Per migliorare l'apprendimento si può propagare le valutazioni fatte all'output per tutti i nodi risalendo fino all'input. Modificando i pesi degli archi. Questa cosa funziona bene e ha senso con le RELU.

Il problema MLP. non ha soluzione oggi, pertanto si procede in modo empirico, non è ancora possibile arrivare ad una soluzione ottima, non si riesce a capire se si ha raggiunto il massimo/minimo globale ma si cerca quella che da risultati accettabili e migliori di altri.

NB: classificare non significa saper approssimare!! Le SVM sono così, non adatti a modellare la probabilità.

5 Lezione 5

Classificatori probabilistici calcolano la probabilità condizionata e cercano di capire il valore della classe attributo da altre variabili. Si basa sul **teorema di Bayes**.

$$P(Y|\bar{X}) = \frac{P(\bar{X}|Y) \cdot P(Y)}{P(\bar{X})}$$

dove:

- $P(Y)$ è la probabilità della classe attributo
- $P(\bar{X}|Y)$ la verosimiglianza di un vettore di attributi data la classe attributo
- $P(\bar{X})$ probabilità dell'evidenza (nel senso di certezza)
- $P(Y|\bar{X})$ probabilità a posteriori della classe attributo dato il vettore di attributi esplicativi

5.1 Naive bayes

Supponiamo Y attributo di classe binario {-1,+1}, $\bar{X}$ è attributo esplicativo binario $\{male, female\}$

Sostanzialmente una volta che comprendo le probabilità condizionate tra attributo di classe e attributi esplicativi, applico la formula di bayes per inferire la classe più probabile.

Quando il numero di variabili esplicative cresce (n variabili), le devo binarizzare quindi avrò 2^n parametri. Che raggiungendo $n = 30$ si arriva a più di un miliardo di parametri, valore assolutamente inaccettabile.

Bisogna allora fare un'assunzione: dati X, Y, Z. Diremo che X è indipendente condizionatamente da Y dato Z, se e sono se la probabilità di X è indipendente dal valore dell'attributo Y una volta che Z sia noto.

$\forall i, j, k P(X = x_i|Y = y_j, Z = z_k) = P(X = x_i|Z = z_k)$

Se assumo questa espressione per le variabili utilizzate si riduce enormemente il numero di parametri da computare, ci permette di andare da 2^n a $2 \cdot n$, in quanto $P(X_1, ..., X_n|Y) = \prod_{i=1}^{n} P(X_i|Y)$

Il record di variabili viene etichettato con il valore di classe che massimizza la probabilità a posteriori. La probabilità computata a posteriori è:

$$P(Y = y_k|X_1, ..., X_n) = \frac{P(Y = y_k) \cdot \prod_{i=1}^{n} P(X_i|Y = y_k)}{\sum_j P(Y = y_j) \cdot \prod_{i=1}^{n} P(X_i|Y = y_j)}$$

Naive Bayes fa parte dei modelli **grafico-probabilistici**. Un intera famiglia di modelli di questo tipo: reti bayesiane dinamiche, ecc... non andremo nel dettaglio.

A fianco del risultato output comunque è bene fornire la probabilità con il quale si è ottenuto il risultato, in modo da dare una *misura di affidabilità*.

il Naive bayes può essere applicato a attributi numerici quali intervalli e ratio. Ogni attributo numerico è associato a una densità di probabilità condizionale di classe normale.

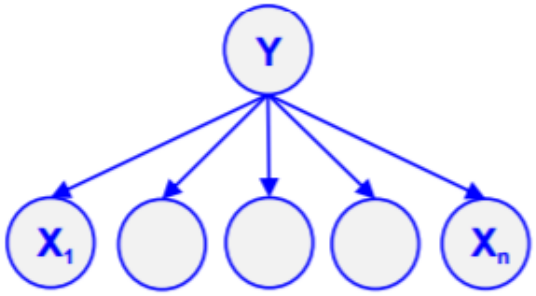

Figura 3: rete naive bayes

Solitamente si combinano attributi categorici (nominali e ordinali) con attributi numerici.

Naive bayes si comporta generalmente bene ma richiede una enorme premessa per essere applicato correttamente (**indipendenza condizionata**). Quindi è stata creata una versione più flessibile mantenendo la stessa capacità computazionale.

5.2 Reti bayesiane

Una rete Naive bayes è generalizzata dalla **rete bayesiana** che è meno forzata dall'assunzione di indipendenza condizionata.

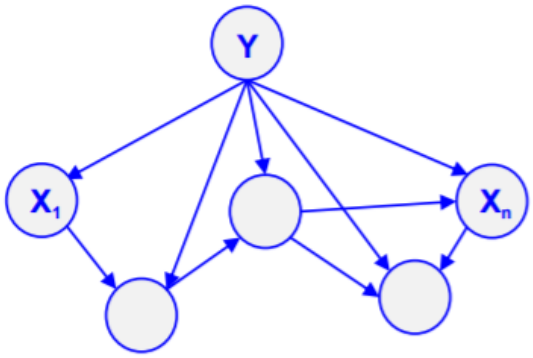

Figura 4: rete bayesiana generalizzata

In queste reti vi è sempre il nodo radice Y che corrisponde all'attributo di classe, ma anche gli attributi esplicativi possono puntare ad altri attributi esplicativi. Inoltre va notato che nel grafo **non** vi possono essere cicli in quanto un nodo successivo non può essere causa di un nodo precedente. Per ogni nodo va specificata una tabella di probabilità condizionata rispetto al valore dei suoi

genitori. In sostanza prendo in considerazione tutte le possibili configurazioni dei genitori e per ognuna do un risultato figlio diverso.

Questo /'e un modello particolare di rete bayesiana che viene utilizzata per la classificazione. La cosa bella di questo modello è che anche con valori null si potr/'a comunque fare inferenza perch/'e nativamente ha questa caratteristica.

Se io ti offro di valutare un certo attributo perchè sapendo il valre di quel attributo so che è condizionato da quell'altro che ho già calcolato.

Nelle reti neurali **non** è possibile far computare un modello senza che io abbia tutti gli input.

5.3 Tree-augmented Naive Bayes

Tree-augmented Naive Bayes oltre ad avere il nodo genitore Y può permettersi di avere un altro nodo genitore (sempre). Se io addestro la rete senza nodo di classe allora ho un albero, dopo inserendovi il nodo Y fa inferenza.

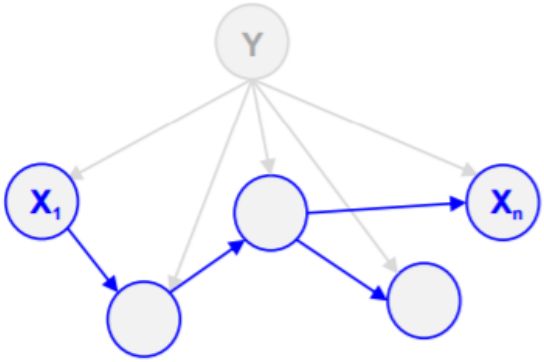

Figura 5: tree-augmented Naive bayes

È molto potente per la feature selection, ovvero per la ricerca delle feature più significative.

Ci sono altre modifiche possibili al Tree-augmente naive bayes, presenti il letteratura.

5.4 Summary

Per capire come *nativamente* si differenziano i modelli di classificazione vedi tabella successiva.

	Explanatory Attribute				Class Attribute	
	Categorical			Numeric	Categorical	
	Nominal	*Ordinal*	*Binary*		*Nominal*	*Binary*
Decision Tree	X	X	X	X		X
Logistic Regression				X		X
SVM		X	X	X		X
MLP		X	X	X		X
NB	X	X	X	X	X	X

Figura 6: comparazione tra i principali modelli

6 Lezione 6

Modelli di separazione nativi tendono a non essere buoni per la stima di probabilità. Modelli di classificazione sono molteplici, bisogna saper valutare quali siano i migliori e quali i peggiori.

La stima dell'accuratezza non è sufficiente per comprendere la confidenza di un modello.

Bisogna rendersi conto del rischio di **overfitting** magnificare alcuni risultati, e **overfitting** ovvero non abbastanza accurati.

Vi sono due diversi tipi di errori:

- training error: numero di record del trainign set mal classificato
- : generalization error: errori non visti precedentemente nei record (test set)

Sempre avere attenzione sulla qualità dei dati!

Se un modello di classificazione fitta il training set troppo bene potrebbe avere un generalization error povero rispetto ad un modello con alto training error, ciò non va bene perchè inficia sulle performace del test set: **Model Overfitting**. Il punto è che il modello deve funzionare bene sui dati di test non per forza in quelli di training. Nella pratica: non devo adattare troppo il mio modello sui dati di training.

Noi puntiamo ad un modello che "sbaglia" sia nel training set che nel test set in modo simile. Questa situazione significa che ho un modello che non ha overfittato.

Il fenomeno contrario è l'**overfitting**. La complessità del modello scelto non è sufficiente per i dati che sto studiando. Accade quando performance del training e test sono simili e basse. Non abbiamo utilizzato tutta la flessibilità che potevamo usare nella definizione del modello.

La soluzione ottimale la si evince da quello che succede nei dati di training e quello che succede nei dati di test. (leggere articolo Baias variance dilemma 2002)

È impossibile avere una misurazione certa sugli errori del modello, si può avere una stima.

6.1 Performance di un modello di classificazione

Problema molto importante è comprendere la catena logica con cui un classificatore impari, perchè per prendere una decisione un individuo deve capire come ha fatto l'algoritmo. In particolare nei contesti delicati quali la guida autonoma in cui la macchina deve sapere come comportarsi in situazioni delicate, ecc.

In una analisi di classificazione vogliamo sviluppare un modello che ci dia una migliore classificazione possibile. Viene valutata in termini di:

- accuracy
- speed
- robustezza
- scalabilità
- interpretabilità (tema più dibattuto)

6.1.1 Accuratezza

importante misura perchè misura il modello in base a se da predizioini affidabili o meno, ci permette di selezionare l'istanza di modello che dà le migliori performance ad un nuovo record.

D_T training set con t records

D_{TS} test set con v records

formula dell'unione ecc

Viene definita una funzione di comparazione tra i valori correttamente rilevati. Viene poi specificata una funzione di accuratezza e il suo complemento, l'errore.

6.1.2 Velocità

Tempo di classificazione e spazio di memoria occupato sono caratteristiche che vanno considerate per il modello. Oggi però si da meno importanza a queste caratteristiche in quanto devono vi sono diverse tecnologie e tecniche ottimizzare i tempi.

Se l'algoritmo impega tempistiche poco accettabili, può essere fatto un campionamento del dataset originale.

Robustezza di un modello sono quelli in cui il modello non rispetta:

- outliers - possono influenzare significativamente il modello
- misssing data - problema centrale (i dati di base sono sporchi e soggetti a deperimento)
- variazione tra training set e test set - si parte dal presupposto che i dati di training siano simili e utili per quelli di test, se questa affermazione decade non è più robusto l'algoritmo

Scalabilità è la capacità di apprendere da enormi quantità di dati. Es. le reti bayesiane scalano molto male.

Interpretabilità è un problema enorme, perchè noi umani non ragioniamo in termini quantitativi, ma qualitativi. Una spiegazione può essere la più dettagliata e precisa possibile ma se non intercetta il linguaggio e il contesto in cui vive il soggetto che voglio raggiungere, non raggiungo il mio obiettivo di far interpretare il mio lavoro.

6.1.3 Holdout

Significa lasciare fuori una porzione del dataset da dedicare al test set e il resto al training. Vi è la best practice 2/3 per training e 1/3 per test. Se si hanno molti dati a disposizione non è necessario seguire questa proporzione si può aumentare sul training.

La stima dipende dalla particolare scelta di divisione tra training e test set. Vi sono anche altre tecniche:

Iterated Holdout consiste di ripetere iterativamente R volte il metodo di holdout per cui apprendo e stimo. Ad ogni iterazione valuto il livello di accuratezza e in base ad esse decido il modello più preciso.

È chiaramente un metodo più robusto però come viene suddiviso il dataset fa la differenza nello studio del metodo migliore. Non ho la certezza di aver usato dei dati ottimi per la stima fatta.

Cross-validation Con questa tecnica divido il fold il dataset. Vengono generati diversi fold di training e viene escluso nella divisione il foglio di

dati dei test. Il test generato è un solo insieme e viene usato per testare tutti i modelli usati precedentemente. (NB: considero sempre solo dati non duplicati). Si provano tutte le possibili combinazioni tra insiemi di training e insieme di test e si prende la stima con la migliiore accuratezza. Il K scelto in letteratura sono: 3,5,10 e non sono per forza le migliori scelte.

LOOCV (Leave One Out Cross Validation): numero di fogli uguale al numero delle osservazioni che vado a considerare, forma più estrema di cross-validation

Se ci troviamo in una situazione in cui una classe è sovrarappresentata, partizionare è molto più problemeatico. In questo caso utilizzo un campionamento stratificato ovvero quello che mantiene le proporzioni all'interno di ogni foglio di dati, non sempre è possibile.

Noi in generale ci fidiamo di più del k-folds cross-validation in quanto rispetto all'holdout ho la certezza che i record non siano ripetuti.

7 Lezione 7 - Comparing Classifiers

7.1 Intervallo di confidenza

L'accuratezza o errore non é sufficiente a definire il migliore classificatore.

Supponiamo di avere:

- Inducer A accuracy = 0.85 con 30 record di test
- Inducer B accuracy = 0.75 con 5000 recordi di test

Qual è il migliore?

Perchè anche se l'inducer A ha un'accuratezza maggiore, è stato testato su un numero di record molto minore rispetto al B. Bisogna testare l'intervallo di confidenza per A e B.

È possibile spiegare la differenza delle accuratezza dei due classificatori? Bisogna svolgere unt est di ipotesi per le deviazioni osservate.

Consideriamo il problema di predire il valore di una classe attributo da un test record come esperimento **binomiale**. Dato un test set D_N contenente N record, consideriamo:

- X: numero di record correttamente predetti dall'inducer (non lo conosciuamo)
- p: vera accuratezza dell'inducer (probabilità di successo della distribuzione binomiale)

X distribuita secondo una binomiale con $media = N \cdot p$ e $varianza = N \cdot p \cdot (1-p)$.

Oggetto del nostro studio è l'**accuratezza empirica**:

$$acc = \frac{X}{N}$$

che è distribuita secondo una binomiale con $\mu = p$ e $\sigma^2 = \frac{p \cdot (1-p)}{N}$

Se la dimensione del test set è sufficientemente grande, è buona norma approssimare una distribuzione binomiale, in una **normale**. Così facendo l'intervallo di confidenza per l'**accuratezza empirica** è:

$$P(-Z_{1-\alpha/2} < \frac{acc - p}{\sqrt{p \cdot (1 - p/N)}} < Z_{1-\alpha/2}) = 1 - \alpha$$

NB: se prendessi diversi test set indipendenti l'uno dall'altro è ovvio che vengono diversi intervalli di confidenza. Però in questi casi fissato α posso stabilire statisticamente quali intervalli sono più significativi di altri. Più riduco α più aumenta β ovvero il caso di errore in cui non rifiuto l'ipotesi nulla quando invece dovrei rifiutarla.

(non chiede la formula gigante!!!)

Supponiamo il caso in cui ho 2 modelli e vengono valutati per forza su due test set diversi (per tanti motivi può succedere):

- M_1 valtutato su test set D_1 contenente n_1 record con tasso di errore e_1
- M_2 valtutato su test set D_2 contenente n_2 record con tasso di errore e_2

I due test set devono essere assunti indipendenti. Il nostro obiettivo è testare se la **differenza** tra i due errori è statisticamente significativa.

La differenza osservata è: $d = e_1 - e_2$ e sono distribuiti secondo una **normale** con $\mu = d_t$ e $\sigma^2 = \sigma_d^2$. la varianza di d può essere stimanata come segue:

$$\sigma^2 = ...$$

L'intervallo di confidenza per la differenza d_t è:

$$(d - z_{1-\alpha/2 \cdot \bar{\sigma}}, d + z_{1-\alpha/2 \cdot \bar{\sigma}})$$

Se l'intervallo contiene 0 allora concludiamo che la differenza osservata non è statisticamente significativa ad un livello α.

Se il limite superiore della confidenza è negativo allora il modello M_1 è migliore del modello M_2 a un livello $\alpha/2$

Se il limite superiore della confidenza è positivo allora il modello M_2 è migliore del modello M_1 a un livello $\alpha/2$

Buona norma fare il test di ipotesi solo se prima mi faccio la domanda se sono o meno differenti, non continuare a condurre test a casaccio o a tappeto perchè altrimenti si rischia di raggiungere degli assurdi. Bisogno farsi la domanda e poi eseguire tutta la procedura sulla coppia di modelli. Vi è un **problema dei confronti multipli** devo adattarli tutti allo stesso livello di α altrimenti non sono confrontabili le differenze.

Nel caso in cui ho lo stesso test set posso svolgere un test più **potente** ovvero posso valutare il caso di errore di secondo tipo: affermo che la differenza tra i due classificatori non è significativa quando in realtà lo è (β).

Si confrontano M_1 ed M_2 usando il k-fold cross validation.

Ho il dataset D partizionato in K subset disgiunti con circa lo stesso numero di record. Vengono costriuti i modelli k-volte in cui ogni volta cambio la partizione di test set. Ottengo:

- M_{1k} inducer per il modello M_1 ottenuto da k iterazioni con e_{1k} errore
- M_{2k} inducer per il modello M_2 ottenuto da k iterazioni con e_{2k} errore

La differenza tra gli errori durante le k iterazioni sono: $d_k = e_{1k} - e_{2k}$, per K sufficientemente grande, d_k è distribuito normalmente con: $\mu = d_t^{cv}$ e $\sigma = \sigma^{cv}$ dove la varianza osservata è stimata secondo la seguente formula:

$$\hat{\sigma}^2_{d^{cv}} = \frac{\sum_{k=1}^{K}(d_k - \bar{d})^2}{K \cdot (K-1)}$$

con

$$\bar{d} = \frac{1}{K}\sum_{k=1}^{K} d_k$$

7.2 Class Imbalance

Consideriamo di nuovo il dataset dei Churner. In particolare sappiamo che il 14.5% sono dei churner. Ricorda un modello che risponde sempre la stessa cosa è un **non modello**. Il modello più inutile è quello che risponde sempre come un **ZeroR Rule** ovvero risponde sempre con la classe più frequente.

Bisogna studiare la classe più rara (meno frequente), essa viene denotata come **classe positiva**, mentre quella più frequente è detta **classe negativa**.

Consideriamo la **matrice di confusione**, allora possiamo calcolare:

- specificità (tasso veri negativi) $TNR = \frac{TN}{TN+FP}$
- sensitività (tasso veri positivi) $TPR = \frac{TP}{TP+FN}$

- tasso falsi positivi $FPR = \frac{FP}{TN+FP}$
- tasso falsi negativi $TNR = \frac{FN}{TP+FN}$

Viene definito un indice chiamato **precisione**: $p = \frac{TP}{TP+FP}$, ovvero il punto di vista del classificatore. Valuto quanti di quelli positivi ha valutato come effettivamente positivi.

Viene anche definito il **recall** (richiamo): $r = \frac{TP}{TP+FN}$, in questo caso ho il punto di vista della realtà. Sotto ho quelli effettivamente positivi, sopra ho quelli che il classificatore ritiene positivi. È la capacità del classificatore di richiamare i risultati positivi (quanti dei positivi è riuscito ad aspirare?).

Vi è un legame molto stretto tra questi due indici, posso avere una recall molto alta ma allora probabilmente avrò una precisione bassa. Oppure potrò puntare ad avere una precisione alta ma in quei casi probabilmente avrò una recall bassa.

NB queste due misure per convenzione sono calcolate sulla classe positiva (quella più frequente) non è però detto che non vadano utilizzate su quella negativa. Inoltre non è sempre detto che queste quantità siano sempre calcolabili (vedi es. slide).

Siccome queste due misure sono legate (pensa alla coperta corta). Si utilizza una misura che le riassume chiamata F_1 measure:

$$F_1 = \frac{2 \cdot r \cdot p}{r + p}$$

F_1 è una misura media armonica tra recall e precision. Un alto valore di F_1 implica alti valori di recall e precision.

Generalizzazione:

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

- F_β con $\beta = 0$ è la Precisione
- F_β con $\beta = \infty$ è la Precisione

8 Lezione 8

8.1 Matrice di costo

Se pensiamo al nostro problema del churner, l'azienda ha interesse ad evitare quello che si prevede, ovvero che il cliente se ne vada, in sostanza cerco di prevedere chi vuole andarsene e cerco di evitare che questa previsione

sia effettiva. L'azienda può spendere un tot budget per invertire questo fenomeno, devo fare il meglio per identificare i possibili churner e attuare SOLO su di loro le politiche di dissuasione (se lo facessi per tutti i clienti non avrebbe senso per quanto riguarda i costi).

Bisogna convincere il proprio interlocutore che il modello da me sviluppato sia migliore rispetto al modello che storicamente hanno utilizzato. Con l'accouracy è facile. Bisogna associarci la **matrice di costo** che stabilisce in base a falsi positivi e negativi quanto costo (in soldi) sostiene l'azienda in base alla situazione.

Costo: $Cost = C_{--} \cdot TN + C_{-+} \cdot FP + C_{+-} \cdot FN + C_{++} \cdot TP$

NB se matrice di costo è simmetrica allora il costo corrisponde all'accuratezza.

		PREDICTED CLASS	
		-1	+1
ACTUAL CLASS	-1	TN	FP
	+1	FN	TP

		PREDICTED CLASS	
		-1	+1
ACTUAL CLASS	-1	p	q
	+1	q	p

N = TP + FN + FP + TN

ACCURACY = (TP+TN)/N

COST = p · (TP+TN) + q · (FN+FP)
= p · (TP+TN) + q · (N–TP–TN)
= q · N – (q–p) · (TP+TN)
= N · [q – (q-p) · ACCURACY]

Figura 7: legame tra matrice di confusione (sn) e matrice di costo (dx)

8.2 Cumulative Gains

È importante notare però che ci sono situazioni nelle quali perdere un cliente è più grave che perderne un'altro. Devo capire se sto utilizzando la matrice dei costi vera oppure se è solo rappresentativa. Se essa non è certa posso ragionare in un altro modo.

Supponiamo che suddividiamo la nostra popolazione in campioni casuali. Vengono valutati la zero rule (campionamento casuale) il modello storico dell'azienda e il nostro modello di classificazione. Si verifica che in termini

percentuali il nostro modello fa 60% e il campionamento casuale fa 37.5%. Il **List Factor** = rapporto tra i due modelli.

NB: considera che il nostro classificatore è bravo ad identificare i casi semplici ma rischia di sbagliare di molto in quelli complessi. Man mano che lo forzo a rispondere lui risponderà in modo sempre più random

Per comprendere il *livello di profittabilità* può essere calcolate una volta che si conoscono i costi coinvolti. Nel nostro caso vogliamo:

- alta proporzione di record positivi
- alto numero di elementi del set

Se ordinassi gli output con probabilità di riconoscimento corretto in ordine decrescente. Posso calcolare il List Factor sui primi risultati per comprendere fino a dove ha buone performance. Man mano aumento gli elementi di risultato per comprendere la sua efficacia se considero elemento con probabilità minore. Quanto ha senso considerare output con bassa probabilità per valutare il mio modello?

Viene quindi visualizzata la cosiddetta curva del **cumulative gains** ovvero la curva che evidenzia questo fenomeno.

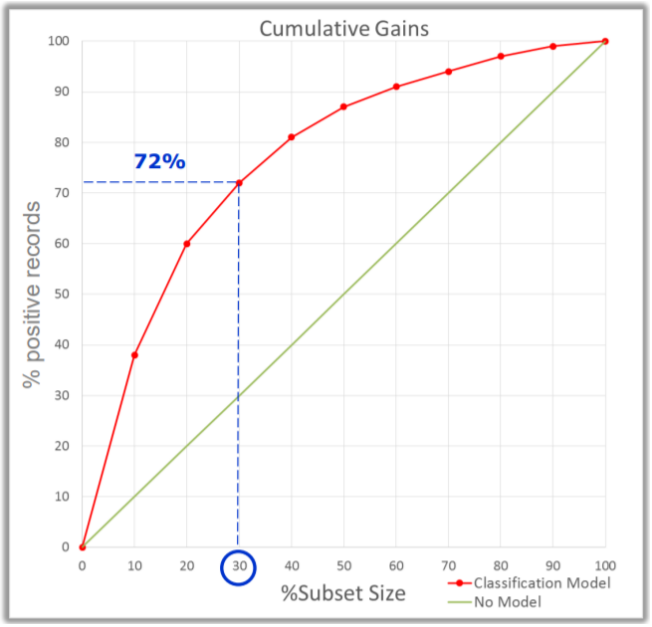

Figura 8: cumulative gains evidenziando un punto di alto valore aggiunto

NB: la retta verde calcola la cumulative gain per un modello che risponde in maniera casuale, sull'asse x è la recall, i positivi riconosciuti correttamente (ricordati ripetendo il test non per un caso ad ogni percentuale)

8.3 Lift Chart

La curva di lift chart è derivata direttamente dalla mappatura della cumulative gains.

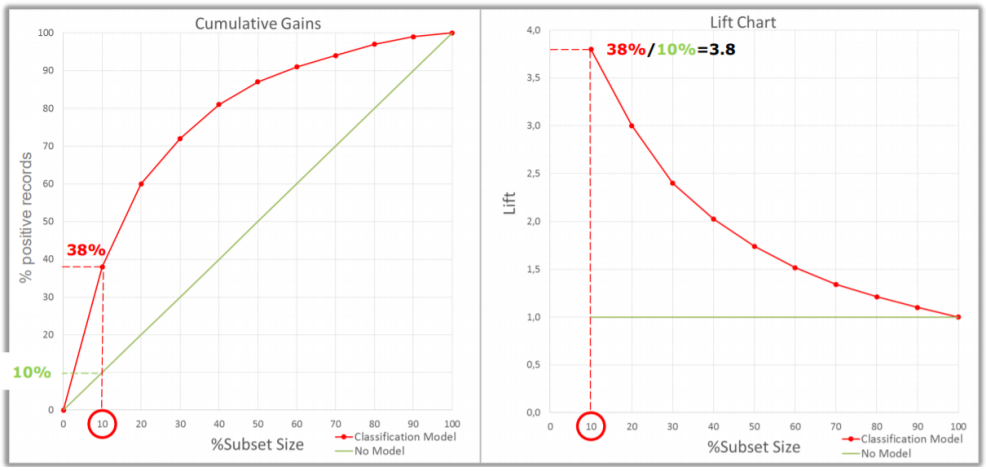

Figura 9: da cumulative gain a lift chart

Queste valutazioni ci fanno comprendere quando il nostro classificatore perde efficacia. Naturalmente più il campione è piccolo meglio funziona, bisogna comprendere però il punto di massima accuratezza in rapporto alla percentuale di istanze.

8.4 Curva ROC

Assomiglia molto alla cumulative gains ma sull'asse delle x ho la percentuale di falsi positivi sopportabili, e sulle y i veri positivi corrisposti. È chiamata **ROC** e sta per Receiver Operating Characteristic curve. Serve a confrontare diversi classificatori per cercare di comprendere dove un classificatore è più o meno efficace.

Un modello è preferibile ad un altro se è disposto a sopportare un certo numero di falsi positivi e valuto quanto veri positivi ci sono.

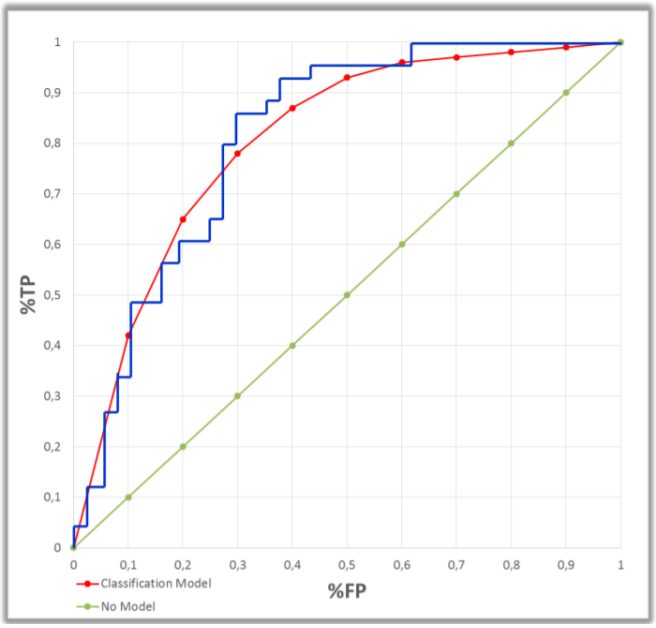

Figura 10: curva ROC legata a dataset variabili (più reale)

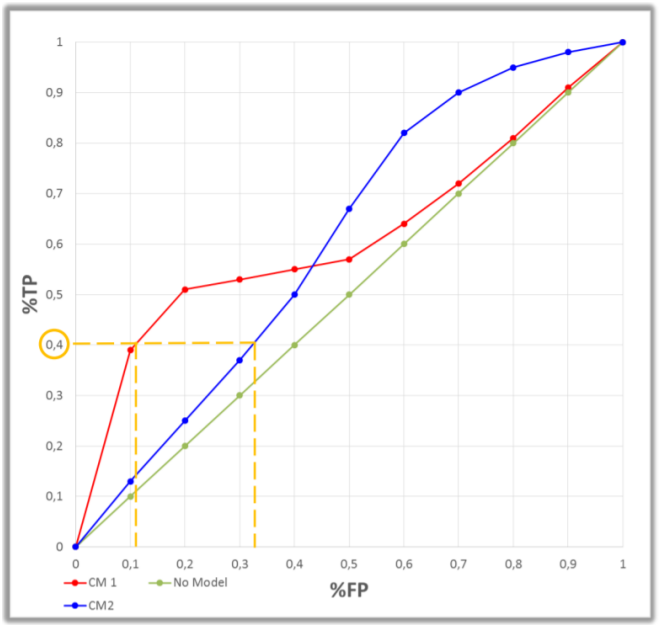

Figura 11: confronto ROC da due modelli

9 Lezione 9 - Feature Selection

La feature selection è un compito importantissimo ed è legato già alla risoluzione del problema. Analizzare le relazioni tra le variabili in modo preliminare potrebbe già aiutare di molto nella ricerca della soluzione migliore al problema.

Quasi sempre non ha senso utilizzare tutti gli attributi per generare il modello che si vuole sviluppare. Quindi si procede con la selezione delle feature.

- attributi **ridondanti** ovvero quelli dei quali l'informazione è già contenuta in altri attributi. La rilevanza va sempre valutata in base ai dati a disposizione
- attributi **irrilevanti** contengono informazioni non utili per risolvere il problema di data mining.

Vi sono diversi approcci per rilevare questi attributi:

- **Brute-force**: provo tutti i modelli per ogni combinazione di parametri usati nel modello di Classificazione. Ma la computazione di questi casi è troppo grossa, vi sono: $\sum_{n=1}^{10}(10n)$ possibili combinazioni
- **Embedded**: attributi scelto in base alla capacità di fare split della classificazione (pensa agli alberi di decisione)
- **Filter**: attributi selezionati pima del classificatore in base a quelli che si ritengono più rilevanti e meno rilevanti (analisi di funzione obiettivo)
- **Wrapper**: gli attributi che si scelgono in base al modello di classificazione scelto, quelli che sono rilevanti per un modello non lo sono per un altro (dipende dall'ipotesi che ho fatto)

NB non va bene fare feature selection su tutti i dati e poi trainare solo su un sottoinsieme!! altrimenti si hanno risultati non comparabili
Filter nel caso di attributi *uni-variati*:

1. si scegle una misura di associazione tra gli attributi candidato e quello di classe
2. si ordinano gli attributi in base alla misura di associazione
3. si selezionano le prime R posizioni come attributi di input per il classificatore

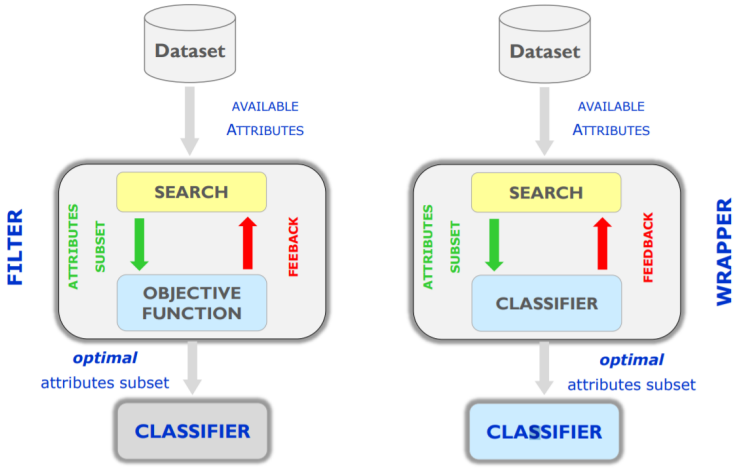

Figura 12: Filter - Wrapper, due differenti approcci per feature selection

solitamente si identificano gli attributi irrilevanti, però non funziona bene nella ricerca di attributi ridondanti.

multi-variati:

- identificano insieme attributi rilevanti e irrilevanti
- un buon sottoinsieme di attributi deve contenere attributi fortemente associati con l'attributo di classe ma essere incorrelati tra di loro

	Advantages	Disadvantages
Uni-Variate	speed scalability independent on the classifier	ignore that attributes can be dependent ignore interactions with the classifier
Multi-Variate	model dependency between attributes independent on the classifier computational cost compares favorably to wrapper	slower than Uni-Variate techniques less scalable than Uni-Variate Techniques ignore interactions with the classifier

Figura 13: vantaggi e svantaggi tra tecniche uni-variate e multi-variate

Vantaggi pe la feature selection sono:

- riduzione del costo di collezione di dati
- riduzione dei tempi di inferenza del classificatore relativo all'attributo di classe
- più alta interpretabilità

- aumento dell'accuratezza

Motivazioni:

- evitare l'overfitting
- sviluppo di un miglior cost-effective classifier
- migliorare la comprensione del processo di generazione dati

Flowchart da seguire:

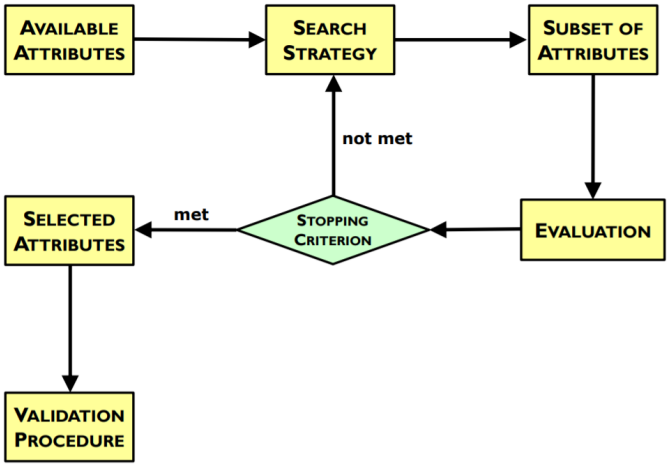

Figura 14: vantaggi e svantaggi tra tecniche uni-variate e multi-variate

La creazione di nuove feature dai dati:

- feature extraction: utilizzati per classificazione di immagini/segnali
- mappare dati in un uovo spazio: vedere gli attributi in modo normalizzato o secondo un'altra scala più facilmente trattabile (SVM)
- feature construction: generare feature nuove più comode per un certo modello di classificatore rispetto a quelle originali attrivarso trasformazioni, es. logaritmi di somme ecc.. (non è detto che i dati originali siano i più utili per la classificazione)

10 Lezione 10 - Classificazione NON binaria

Nei problemi reali la classificazione avviene su più valori non solo binari.

L'idea è quella di trasformare un problema multi-classe in tanti attributi di classe binari. **Multi-classe** esattamente una classe si realizza. **Multi-etichetta** più di una classe può verificarsi. Vi possono essere anche problemi non di classe ma di **ranking** in cui i valori assunti dalla variabile di classe sono ordinati.

10.1 One-Vs-All

In questa modalità si verifica se il set corrente verifica o meno ciascuna delle caratteristiche da valutare.

Ricorda che non sempre binarizzare è necessario, in particolare per quanto riguarda i naive bayes non serve, invece per un decition tree sì.

Nel *One-vs-all* creo un numero di classificatori diverso in base al numero di modalità dell'attributo di classe. Inoltre bisogna normalizzare poi gli output dei classificatori. Nel modello *multi-label* bisogna prendere i valori così come li restituisce il cassificatore e fissare un threshold sopra il quale considero signficiativo il risultato (potenzialmente più di una classe supera il threshold).

10.2 Train/validation/test e feature selection

Una rete neurale è dimostrato che permette architetturarla per approssimare qualsiasi problema. Non è però esente da overfitting anzi visto che non è sempre chiaro il funzionamento è più difficile valutarle.

Se volessimo scegliere i parametri (pesi e soglie) e un **iperparametro** λ detto di *regolarizzazione*. Voglio sceglere qella allocazione di parametri per ridurre il piò possibile l'errore quadratico del modello, quindi l'iperparametro.

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

La prima parte è il fitting della funzione, la seconda parte è la flessibilità alla quale posso accedere. Potenzialmente io posso imparare il valore ottimale di λ dai dati.

Per questo suddivido il dataset in train e test. Se utilizzazzi i dati di training per imparare il migliore lambda allora andrei in overfitting di lambda. Quindi suddivido in questo modo il dataset:

- train dataset viene utilizzato per trainare il modello (sui parametri w)
- validation dataset che viene utilizzato per trainare su lambra
- test dataset viene utilizzato per testare definitivamente il modello

NB se utilizzo il test set per ottimizzare il parametro lambda allora ho bruciato il dataset e devo rifare, obiettivo è sempre quelo di riconoscere dati mai visti.

11 Clustering

Problemi di clustering sono una serie di problemi nei quali non si sa bene come agire in quanto non vi è un supervisore.

La **cluster analysis** ha due scopi:

- **comprensione**: classi o gruppi di oggetti condividono certe carattersitiche. Gioca un ruolo importante come le persone analizzano e descrivono il mondo.
- **utilità**: capacità di risassumere determinate carateristiche, l'obiettivo è trovare le proprietà più rappresentative dei cluster; voglio trovare il **prototipo** per ogni classe.

Gli oggetti all'interno di un gruppo devono essere simili gli uni con gli altri, allo stesso tempo diversi (o incorrelati) con gli oggetti di altri gruppi.

La più grande *similarità* entro un gruppo deve corrispondere ad una grande *differenza* tra i gruppi.

Problema: come faccio a stabilire quando degli oggetti siano simili? Non vi è un metodo per capirlo, tendenzialmente si utilizza una soluzione intermedia rispetto alle altre.

La definizione di cluster è **intrinsicamente imprecisa** e una migliore definizione dipende dalla natura dei dati e dai risultati desiderati.

11.0.1 Tipi di clustering

Partizionale vi è una divisione del set di oggetti dati in sottonisime e nei quali ogi oggetto è in un solo sottoinsieme

Gerarchica Posso aggregare dei cluster in uno più grande in cui ho diversi livelli di imprecisione. Considero un certo sottoinsieme simile ad un altro allora li metto insieme.

Vi sono inoltre:

- esclusivi dei quali non includono altri
- overlapping nei quali ho delle osservazioni condivise da più di un cluster

- fuzzy permette che un oggetto appartenga a ciascun cluster con un peso membro tra 0 e 1.

Clustering **completo** ogni osservazione viene assegnata ad un cluster

Clustering **parziale** non tutte le osservazioni sono assegnate ad un cluster in quanto magari sono outlier per cui se la includessi mi farebbe peggiorare in modo significativo il cluster considerato (altrimenti Garbage cluster).

11.0.2 Differenti nozioni di cluster

I cluster naturali sono cluser che si dicano esistano per davvero, è molto difficile che ciò accada.

Well separated Cluster ogni oggetto nel claster è più vicino a ogni altro oggetto nello stesso claster rispetto alla vicinanza ad altri cluster. Avere dei cluster così ben formati permettono separazioni molto nette (quasi mai succede).

Prototpe-based cluster dato un cluster, ogni oggetto è vicino al prototipo che lo definisce il cluster rispetto al prototipo di un altro cluster. Il prototipo è solitamente il *centroide* del cluster, es. la media di tutti gli oggetti nel cluser. Che corrisponde all'individuo meglio rappresentato dal cluster (anche se esso non esiste).

Density-based cluster un claster in una regione densa di oggetti che sono circostritti da una regione di bassa densità. Usati quando i cluster sono irregolari o intermittenti

Graph-based cluster se i dati sono rappresentati come grafi, i nodi sono oggetti e il collegamenti connettono gli oggettti. Allora in cluster è una componente connessa. La connessione può anche essere pesata e scelta in base ad una certa soglia.

Quelli connessi a grafo sono molto utilizzati in quanto c'è un sacco di ricerca dietro già fatta e molto robusta.

11.0.3 Metodologia

Dal dataset, estrarre le feature o selezionarne di esterne. Applicare un algoritmo di clustering, per poterlo fare devo specificare una misura di affinità. Successivamente bisogna capire come impostare l'ottimizzazione di una misura di merito scelta. Dopo tutto ciò bisogna ricriticare tutto ciò che si è fatto e valutare il funzionamento e le eventuali performance, entrare nel merito e valutare se i gruppi siano sensati. Si tratta di investigazione sul clustering. Successivamente si cerca un'interpretazione dei risultati chiedendo all'esperto di dominio cosa ne pensa. Saranno i cluster non compretamente palesi all'esperto i più interessanti da valutare.

12 Lezione 11 - Proximity

La proximity è uno strumento fondamentale per valutare il funzionamento di un algoritmo di clustering.

La cluster analysis affonda le sue radici in cosa sia **simile** e cosa sia **dissimile**. La similarità è un conetto relativo, nel senso che posso considerare uno smartphone e un telefono simili se li considero come strumenti di comunicazione però non per quanto riguarda la forma.

La **similarità** è un valore alto per paia di oggetti che si assomigliano e solitamente si usa un valore compreso tra 0 e 1.

La **dissimilarità** è una misura numerica di gradi per i quai due oggetti sono differenti. In generale si cerca di ricodurla in un intervallo $[0, 1]$ però anche di $[0, \text{inf}]$.

La **prossimità** ulilizza riferimenti a similarità e dissimilarità.

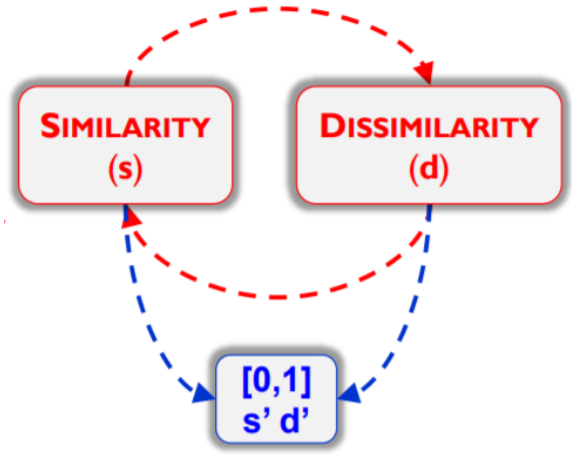

Figura 15: vantaggi e svantaggi tra tecniche uni-variate e multi-variate

Se voglio trasformare una misura di similarità in $[0, \text{inf}]$ allora ho bisogno di una trasformazione **non lineare** e la relazione tra i valori non avrà la stessa scala. Verranno compressi dei valori in valori distorti. (*in alcuni casi distorcere i valori è quello che desidero*).

La trasformazione tra similarità e dissimilarità e sostanzialmente lineare, bisogna scegliere un metodo per metterli in relazione.

Comunque, la **prossimità** tra due record è una funzione di prossimità compresa tra i corrispondenti attributi dei due record.

Per quanto riguarda gli attributi binari la dissimilarità esclude la similarità con valori 0 e 1.

Se ho attributi nominali ordinali assegno un valore intero agli stessi in base alla scala utilizzata e calcolo la dissimilarità come rapporto tra la differenza dei due record e la scala totale di valori di utilizzo. Naturalmente va notato che sto utilizzando una scala linerare! (forte assunzione)

Qualsiasi scala di valori si andrà a scegliere si porterà dietro delle assunzioni.

12.1 Misure della distanza

Le distanze sono dissimilarità con certe proprietà, quando tutti gli attributi sono numerici.

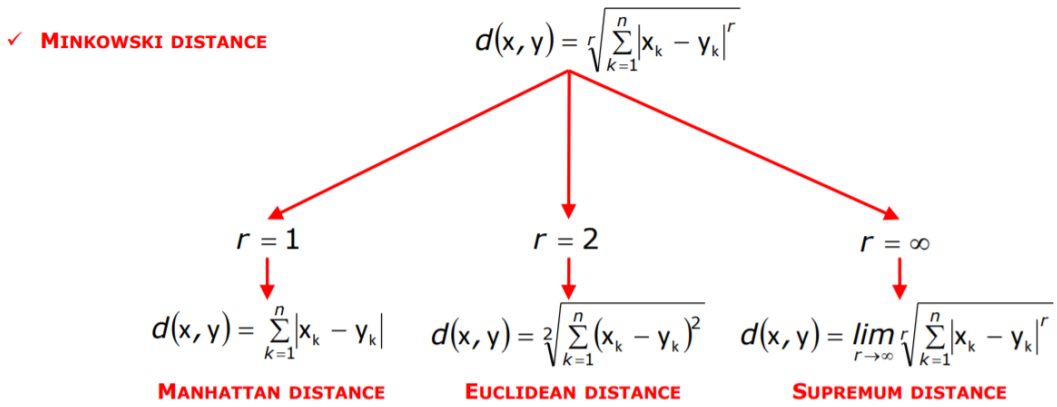

Figura 16: vantaggi e svantaggi tra tecniche uni-variate e multi-variate

La misura deve essere

- non negativa
- simmetrica
- disuguaglianza triangolare

La similarità solitamente non rispetta la disuguaglianza triangolare ma è simmetrica e non negativa.

Misure di coefficienti

Simple matching coefficient:

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

Questa misura ci piace quando gli 0 e gli 1 sono gli stessi. Si dice che gli 0 sono molto meno nobili degli 1. È una misura equilibrata.

Jaccard Coefficient

$$J(x,y) = \frac{\#ofmatchingpresences}{\#attibutesexcept00matches}$$

Questa misura distorta rispetto gli 1 mi permette però di focalizzare la mia attenzione sugli 1.

Una versione di questo indice è la Extended Jaccard Coefficient, viene utilizzato nell'analisi di testo in linguaggio naturale (Es. post su twitter) analizza vettori di attributi.

$$EJ(x,y) = \frac{x \cdot y}{||x||^2 + ||y||^2 - x \cdot y}$$

Similarità al coseno, utilizzata quando gli attributi hanno natura numerica (utilizzata in information retrival).

Correlazione quella di Pearson ma non legato a variabili aleatorie, bensì quelle con i vettori.

Problemi:

- come trattare attibuti su scale di ampiezza diverse e correlati
- come calcolare la prossimit'' a tra record composti da attributi di tipo diverso
- come trattare la prossimità quando assegnavo una diversa rilevanza a diversi attributi, non tutti contribuiscono allo stesso modo

nel primo caso bisogna normalizzare i valori in modo che rispettino la stessa scala e poi applicarci misure di distanza come la euclidea, quando invece tutti gli attributi sono correlati e la distribuzione approssimata dalla gaussiana allora uso la distanza di Mahalanobis

nel terzo caso la similarità diventa:

formula con i pesi della similarità

Indicazioni generali:

- dati densi e continui: distanza come quella euclidea buona rappresentazione
- dati sparsi, binari asimmetrici: misuri della distanza che ignorano i match 00 come cosine, Jaccard e Extended Jaccard

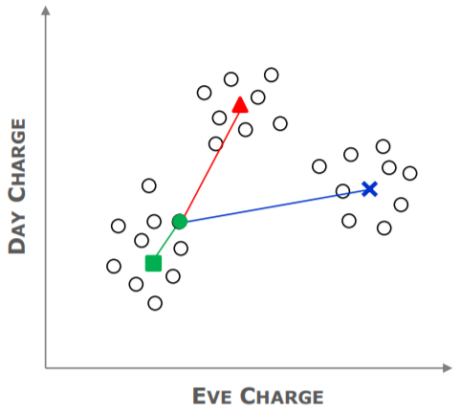

Figura 17: esempio di prototype-based clustering

12.2 Clustering Algorithms

Prototype-Based Custering un cluster è un sottoinsieme di oggetti come vicini (similarità o dissimilarità) rispetto ad un oggetto prototipo.

il quadratino, triangolino e x sono i prototipi centroidi dei cluster. Mi focalizzo su un'osservazione e essa viene assegnata al cluster del centroide prototipo più vicino.

Caratteristiche:

- oggetti appartengono ad un unico cluster
- oggetti sono condizione di appartenere a più di un gruppo
- un cluster è modellato con una interpretazione probabilistica
- i cluster sono vicolati ad avere relazioni fisse (vicinato)

12.3 K-medie

Il prototipo in questo caso prende il nome di **centroide**, di solito è indentificato dal vettore media dei valori degli attributi delle osservazioni di quel determinato cluster (quindi dei record). Solitamente applicato a oggetti in uno spazio continuo n-dimensionale.

Bisogna comunque dichiarare il numero di cluster da identificare (K cluster) e fa la differenza questo parametro!!

12.3.1 Algoritmo

1. Selezione K oggetti come centrodi

2. **Reapeat** (ciclo)
3. Da K cluster, assegnamo ogni record al centroide più vicino
4. Ricalcola il centroide di ogni cluster
5. **Until** i centroidi non cambiano

13 Lezione 12 - Clustering Algorithm

K-mean: vi sono diverse misure di sitanza che popssono essere applciate [errata corrige: massimizzare la somma di coseno (slide 3 part II)]

Problemi sono:

- la scelta dei iniziale centroidi è fondamentale e influenza in modo pesante le performance dell'algoritmo in generale. Vi sono alternative quali il clustering gerarchico.
- punti a favore del K-medie è che occupa il minimo spazio ed è un algoritmo abbastanza rapido in quanto è lineare per il numero di istanze considerate.
- può accadere che un cluster sia vuoto per scelta sbagliata di centroidi iniziale (magari randomica).
- problema di Outlier: crea grossi problemi quando calcolo la media delle osservazioni di un cluster (è però efficiente nella ricerca di outlier in quanto li indentificherà come sigleton)

Limiti sono:

- difficoltà a ricercare cluster di non forma sferica
- difficoltà a trovare cluster con dimensioni diverse
- difficoltà a indentificare cluster di diversa densità (vicinanza dei punti)

Per ovviare ad alcuni di questi problemi vi sono tecniche per clusterizzare in cluster più numerosi ed alla fine unire secondo degli algoritmi specifici nel numero di cluster richiesto, es minimizzazione delle distanze dei centroidi.

Quindi K-medie è:

- semplice e si applica a tanti tipi di dati

- è abbastanza efficiente anche se vengono effettuate più applicazioni successive dello stesso
- vi sono varianti più efficienti e meno problmatiche (es. bisecting k-means)
- non adatto a tutti i tipi di dat e non può gstire cluster non sferici, con size e densità diverse
- problemi con dati outliers
- è strettamente legato al concetto di centroide. Vi sono delle varianti che utilizzano il medoide ed è più efficiente

NB: più cluster vengono utilizzati più aumenta la complessità computazionale.

13.1 Fuzzy C-means

Se gli elementi sono distribuiti in gruppi ben separati il clustering è semplice, i problemi nascono dove le istanze sono molto vicine. Inoltre consideriamo il fatto di osservazione sulla frontiera delle distanze dei cluster.

Per ovviare al problema ogni osservazione viene considerata come appartenente ad ogni cluster ma con una misura diversa per ognuno! (es. utilizzando la misura della media)

w_{ij} è il **peso** con cui l'i-esimo oggetto appartiee al j-esimo cluster.

La **pseudo-partizione Fuzzy** è definita assegnando prima l'isieme di tutti i pesi w_{ij} con $w_{ij} \geq 0$ ed essi devono essere verificati dai seguenti vincoli:

$$\sum_{j=1}^{K} w_{ij} = 1$$

con $i = 1, ..., m$

$$0 < \sum_{i=1}^{m} w_{ij} < m$$

con $j = 1, ..., K$

Ogni oggetto deve essere assegnato ad un certo grado a tutti i cluseter. Non sono permessi cluster vuoti e ogni oggetto può essere assegnato esclusivamente ad un singolo cluster.

Il Fuzzy C-means produce cluseter che danno un'indicazione del grado in cui ogni oggetto appartiene ad ogni cluster, ha gli stessi punti di forza e debolezza del K-means ma più pesante computazionalmente.

13.2 Modelli a mistura

In questi modelli vediamo le osservazioni da una mistura di diverse distribuzioni di problabilità. In sostanza possiamo vedere le misture come distribuite secondo delle normali con uguale varianza ma medie diverse (curve di livello) [proprietà rilassabile].

Supponiamo spazio di 3 componenti, processo di generazione:

1. Seleziono uno delle 3 componenti
2. Estraggo un campione dalla componente selezionata
3. ripeto 1 e 2 m volte per ottenere il dataset

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

probabilità dell'oggetto x dove w_j probabilità della j-esima componente, $p(\bar{x}|\theta_j)$ probabilità di estrarre x dalla j-esima component, θ_j parametri associati a j.

Il processo sostanzialmente è quello di partire dai dati e ridurrli nelle misture significative. (simile a come avviene la nostra capacità di isolare il rumore uditivo). Vi è un algoritmo utilizzato per la risoluzione che è l'**Expectation Maximization**. Si tratta di una classe di algoritmi che sono diversi in base alla loro applicazione.

Svantaggi:

- Apprendimento è lento
- non è pratico per i modelli con un gran numero di componenti
- non funziona bene quanto i cluster conengono poche osservazioni
- non funziona bene quando gli oggetti sono co-lineari
- più generico di k-means e fuzzy c-means perchè usa distribuzioni di vario tipo
- disciplina l'approccio di eliminazione di complessità

13.3 Mappe di Kohonen o SOMs

Una struttura feedforward per algoritmi di clustering. Detta anche mappa atorganizzante. Vi è una mappatura dei neuroni che mappano in uno spazio bidimensionale. Non hanno una sola direzione ma sono bi-direzionali, possono spostare i pesi sia da input ad output che viceversa.

Ho uno spazio di input continuo e voglio mapparlo in uno spazio discreto di output. Vorrei che mappando un'osservazione in un neurone posso poi tornare indietro tramite un altro collegamento.

Vi è un'informazione topografica in tutto ciò. Ed è una cosa unica rispetto agli altri algoritmi di clustering considerati. Durante la fase di apprendimento sfruttiamo la topologia considerando ogni oggetto aggiornato al centroide più vicino secondo essa. Considerando un neurone assegnato ad una osservazione, allora anche i neuroni ad esso adiacenti (vicinato - definito in vari modi) dovrebbero risentirne di questo assegnamento, in questo modo vengono trainati i neuroni e i vicini degli stessi.

Riassumendo:

1. inizializzazione, seleziono i centroidi assocaiti a ciascun neurone
2. competizione tra diversi neuroni per ottenere l'istanza. Ogni neurona fa un'offerta diversa per vincere quell'oggetto e la proposta è proporzionata in base a quanto un neurone (in base al suo centroide) si considera "vicino" all'istanza considerata, viene poi proclamato il vincitore e gli viene assegnata l'istanza (in base a una misura di performance)
3. collaborazione una volta che un neurone vince distribuisce il suo vantaggi ai vicini (processo di premi-punizioni) aggiustando i centroidi
4. aggiornamento del centroide del vincitore e dei vicini calcolandol attraverso l'istanza appena considerata

Punti di forza Interpretabilità dei cluster
Limiti:

- bisogna scegliere un gran numero di parametri, funzione di vicinato ecc e varia molto nelle performance
- non sempre un raggruppamento identifica un singolo cluster naturale (solitamente dopo viene applcato un k-medie sui centroidi trovati)
- difficile paragonare i risultati
- la convergenza non è garantita, è un'euristica

14 Lezione - Clustering Algorithm

14.1 Clustering Gerarchico

- **agglomerativo**: si parte da tutti gli oggetti come cluster individuali e ad ogni step unisce le coppie di cluster più vicine.
- **divisivo**: parte da un cluster unico e passo passo divide il cluster in soli cluster singleton di oggetti individuali rimanenti. Abbiamo bisogno di decidere quale cluster splittare ad ogni passo e come splittare

Clustering gerarchico agglomerativo sono i più comuni. La soluzione che si ottiene è il **dendrogramma** del clustering effettuato. È computazionalmente molto pesante.

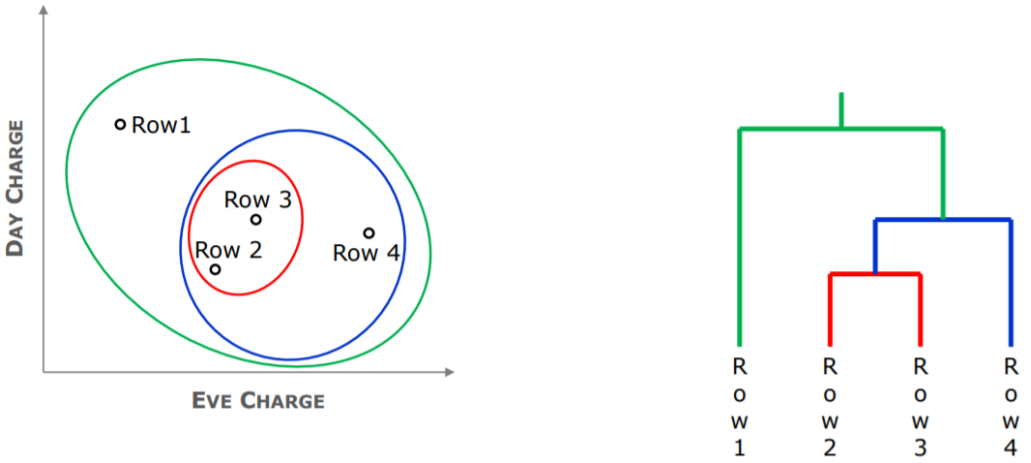

Figura 18: da clustering gerarchico a dendrogramma

Si parte dai singoli elementi e pian piano vengono generati cluster sempre più inclusivi fino ad ottenere un unico cluster.

Passaggi:

1. Calcolare la matrice di prossimità
2. **Ripeti**
3. Unione di due cluster vicini
4. Aggiorna la matrice di prossimità (o delle distanze) che riflette la prossimità tra il nuovo cluster e il cluster originale

5. **Finchè** rimane un solo cluster

La prossimità tra cluster viene calcolata con:

- *Min or Single Linkage*: minore distanza tra tutte le possibili coppie di elementi presenti nei due cluster (soluzione più ottimista)
- *Max or Complete Linkage*: maggiore distanza tra tutte le possibili coppie di elementi presenti nei due cluster (soluzione più robusta o pessimista)
- *Group Avarage or Avarage Linkage*: media distanza tra tutte le possibili coppie di elementi presenti nei due cluster (soluzione media - politically correct cit)
- *metodo di Ward*: assume i cluster come rappresentati dai centroidi, misura la prossimità tra due cluster come somme di scarti quadratici che risulta dalla fusione di due cluster

Caratteristiche:

- Non risolve problemi di ottimizzazione globale
- può gestire cluster con dimensioni differenti, gli elementi possono anche essere pesati o meno
- le decisioni di unione sono definitive: non si può tornare indietro rispetto alla decisione (approccio greedy)

Difetti:

- computazionalmente molto pesante e richiede molto spazio
- decisioni di unione di cluster sono subottimali quindi creano rumore

14.2 Density-Based Clustering

Densità di regioni di oggetti che sono circondati da regioni con bassa densità di oggetti.

DBSCAN è un semplice ma efficace metodo density-based.

Vi sono diversi metodi per definire la *densità*, descriviamo il center-based approach: la densità è il numero di oggetti all'interno di uno specifico raggio (**Eps**) di quell'oggetto.
Ci sono 3 tipi di punti:

- **core point**: stanno all'interno della regione ad elevata densità. È core pont se il numero di punti nel vicinato attorno ad esso eccede un certo *threshold* MinPts, impostato dall'utente
- **border point** (wanna be core point): un punto che include nel suo vicinato un core point, quindi lui si considera un vicino di core point
- **noise point** (osservazione residuale no core o border): non è ne un core point ne un border point

Passaggi

1. Etichettare tutti i core, border, noise point
2. Si eliminano i noise point
3. collegano i core point che sono all'interno dei rispettivi Eps
4. si fa ogni gruppo di core point connessi in cluster separati
5. si assegna ciascun border point in un cluster associato ai core point più prossimo

La scelta dei parametri *MinPts* e *Eps* è fondamentale.

DBSCAN vs K-medie:

- DBSCAN and K-Means assign objects to a single cluster, but K-Means assigns all objects while DBSCAN can discard noise objects
- DBSCAN can handle clusters of different sizes and shapes and it is not strongly affected by noise or outliers. K-means has difficulties with non-globular clusters and clusters of different sizes. Both algorithms perform poorly when clusters have widely differing densities
- K-Means can only be used for data that has a well defined centroid, such as mean or median. DBSCAN requires that its definition of density, which is based on the traditional Euclidean notion of density be meaningful for the data
- K-Means can be applied to sparse, high-dimensional data, such as document data. DBSCAN typically performs poorly for such data because the traditional Euclidean definition of density does not work well for high-dimensional data

- DBSCAN makes no assumption about the distribution of the data. The basic K-means is equivalent to a statistical clustering approach (Mixture Model) that assumes all clusters come from spherical Gaussian distributions with different means but the same covariance matrix
- DBSCAN and K-Means both look for clusters using all attributes, that is, they do not look for clusters that may involve only a sub-set of the attributes
- K-Means can find clusters that are not well separated, even if they overlap, but DBSCAN merges clusters that overlap
- K-Means has complexity O(m) while DBSCAN is O(m2) (except for low dimensional Euclidean data.)
- DBSCAN produces the same set of clusters from one run to another while K-Means, which is typically used with random initialization of centroids, does not
- DBSCAN automatically determines the number of clusters, for K-Means the number of clusters needs to be specified as a parameter

Vi sono altri algoritmi di density-based:

- Grid-based clustering
- subspace clustering
- kernel density function

14.3 Graph-based Clustering Algorithm

Già gli algoritmi a rappresentazione gerarchica possono essere visti come graph-based.

Sparsificare: tagliare, rimuovere certi archi che non superano la verifica di una certa condizione (Es. tutti gli archi che non superano una certa soglia, oppure un concetto di vicinato).

Definire una misura di similarità tra due oggetti basati sul numero di nearest neighbors.

Definire oggetti core e costruire cluster attoro ad essi. Necessita di concetti di densità e prossimità.

Utilizza l'informazione nel grafo di prossimità per fornire una maggiore valutazione di specialità in cui due cluster dovrebbero essere uniti.

Grafo di prossimità stabilisce dei pesi tra i vari nodi che è il costo da percorrere tra un nodo ed un altro, è rappresentazione della matrice delle distanze.

I legami di similarità tra nodi vengono suddivisi in *sufficientemente vicini* (superano il threshold) e *non vicini* quindi con vicinanza minore rispetto al threshold. Vengono quindi rimossi gli archi che non superano il threshold, questo significa *sparsificare* la matrice. Viene così generato il grafo di prossimità sparsificato.

Definiamo quindi il **K-Nearest Neighbor**
se fisso k = 3 scelgo per ogni riga della matrice delle prossimità i primi 3 valori degli archi per come vicinato.

2 modi di procedere: threshold o k neighbors.

La sparsificazione riduce anche di molto il numero di elementi del dataset. Il clustering lavora molto meglio in quanto si lavora in un concetto di vicinanza. Algoritmi su grafi partizionati possono essere usati.

14.4 Altri algoritmi

Non sempre purtroppo la sparsificazione del grafo porta ad un grafo sconnesso. L'idea è di ciclare la sparsificazione finchè non otterrò dei grafi partizionati ben definiti.

Minimum spanning tree o albero di supporto a costo minimo:

- non ha cicli
- contiene tutti i nodi dell'albero
- ha il minimo costo totale dei pesi associati ai suoi alberi di tutti i possibili spanning tree

MST divisive hierarchical clustering Algorithm

1. Calcola il MST per grafo di dissimilarità
2. **Ripeti**
3. Crea un nuovo cluster per rompere il collegamento corrispondente alla più grande dissimilarità
4. **Finchè** rimangono solo cluster singoli

Algoritmo **Opossum**:

1. Caclola il grafo di similarità sparso

2. partiziona il grafo in k componenti distinte (cluster) usando METIS

Altro algoritmo **Camaleon**.

Quando si visualizza il dendrogramma, si possono visualizzare le distanze e solitamente ha una forma a "gomito" la soluzione ottimale è solitamente quella che precede la salita finale del comito. Un buon dendrogramma ha grandi salti prima di aggregare insieme (salti grandi significa che se raggruppo genero un errore importante quindi è positivo per la valutazione del clustering).

15 Lezione - Clustering Evaluation

L'efficienza di un algritmo di clustering viene valutato tramite:

- similarità
- esclusività dei clustering o misure fuzzy di sovrapposizione
- completo o parziale clustering

Innanzitutto scelgo alcuni algoritmi candidati da voler implementare in base alla tipologia di dati che ho. L'utilità di utilizzarne diversi e poterli confrontare. Bisogna impostare un'insieme di esperimenti equi per poterli valutare correttamente (stessi dati, chiare differenze di parametri ecc). Il problema è che è molto complesso valutare i cluster visto che non so esattamente cosa sto cercando.

I problemi grossi nella valutazione sono:

- determinare la tendenza dei cluster (che non siano fatti a caso)
- determinare il numero corretto di cluster (qualunque cosa significhi)

abbiamo 3 diversi tipi di indici:

- **Esterni o supervisionati**: misure che estendono i cluster scoperti con alcune strutture esterne (sapendo circa la verità sul problema)
- **Interni o non supervisionati**: misure di similarità di cluster in particolare le strutture in base a informazioni esterne, possono essere:
  - Misure di coesione: determina la connessione tra elementi di un cluster
  - misure di separazione: quanto sono ben distiniti i cluster
- **Relativi**: compara differenti clustering

15.1 Esterni o supervisionati

$P = \{P_1, ..., P_R\}$ partizione di un dataset di m (oggetti record) in R categorie, possono essere partizionati con algoritmi in k cluster $C = \{C_1, ..., C_K\}$.

Casi:

1. (**a**) x e y appartengono allo stesso cluster di C e alla stessa categoria di P
2. (**b**) x e y appartengono allo stesso cluster di C e a *differenti* categorie di P
3. (**c**) x e y appartengono a *differenti* cluster di C e alla stessa categoria di P
4. (**d**) x e y appartengono a *differenti* cluster di C e a *differenti* categorie di P

Numero totale di coppie che si possono formare:

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

Rand:

$$R = \frac{a+d}{M}$$

con $R \in [0, 1]$

Jaccard:

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

con $J \in [0, 1]$

Fowlkes and Mallows:

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

Γ statistics

15.2 Interni o non supervisionati

Molte misure di validità del partizionamento dei cluster si basano sulla **coesione** e sulla **separazione**. Noi di base vogliamo un valore di coesione alto ed un valore di separazione basso.

Ci si basa sul concetto di grafo (graph-based), la coesione di un cluster può essre definita come a somma dei pesi degli archi inn un grafo di prossimità che connette i punti all'interno del cluster.

$$cohesion(C_i) = \sum_{x,u \int C_i} proximity(x,y) = \sum_{xy \in C_i} similarity(x,y)$$

La similarità è quindi inversamente proporzionale alla dissimilarità. La separazione è valutata secondo la somma dell prossimità di nodi che non appartengono allo stesso cluster.

Per i cluster prototype-based la coesione è valutata rispetto al centroide che ne rappresenta il prototipo; la separazione è misurata sulla prossimità del centroide o prototipo del cluster. Altro metodo per la separazione è la valutazione rispetto al centroide generato dai due cluster.

La validità è valutata in questo modo: $overallvalidity = \sum_{i=1}^{K} w_i \cdot validity(C_i)$, dobbiamo decidere però il peso da applicarvi (W_i).

Importante capire che alle volte separare un cluster potrebbe migliorare le misure di coesione, quando ho valori di coesione bassi.

Se invece ho valori di coesione buoni ma valori di separazione alti allora provando ad unirli si raggiungono cluster più coesi.

La valutazione che sta all'interno del cluster contribuisce maggiormente informazione alla coesione e separazione, invece per le osservazioni più esterne queste valutazioni possono non essere così utili. Pertanto è stato definito l'**indice di Silhouette**. questo indice combina coesione e separazione per l'i-esimo oggetto e si definisce in questo modo:

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

dove a_i è la media della distanza tra l'oggetto i-esimo e gli altri oggetti nel cluster, b_i la minima distanza media dell'oggetto i-esimo a tutti gli altri oggetti di ogni custer diverso dal cluster di partenza dell'oggetto.

(es. se $a_i = 0$ allora $s_i = 1$ perchè si trova nel cluster migliore. se ha invece valore < 0 significa che se avessi assegnato ad un altro cluster avrei avuto un indice di silhouette più alto)

L'avarage Silhouette coefficient di un cluster da un valore di silhouette medio

Per il clustering gerarchico viene definito il Cophenetic Correlation Coefficient: Si considera la matrice di prossimità e la matrice cophenetica. Cophenac matrix è la matrice che tiene conto delle distanze tra coppie di osservazioni raggruppate all'interno di uno stesso cluster. Il valore varia da $[-1, 1]$. Più il valore è vicino a 1 meglio l'algoritmo gerarchico fitta.

15.3 Paradigma di validità

Rispondere alla domanda se il nostro dataset tende ad essere clusterizzabile.

Vengono considerati criteri interni ed esterni legati a metodi statistici e test di ipotesi.

Se non ci fosse nessuna struttura dei dati (ipotesi NULLA).

- Random Position Hypothesis in tutte le location di m osservazioni in specifiche regioni n-dimensionali sono equamente buone → per ratio data
- Random Graph Hyothesis la matrice di prossimità ha rank equamente buoni → per prossimità ordinali tra coppie di osservazioni
- Random Level Hypothesis tutte le permutazioni delle label di m osservazioni sono equamente buone → per tutti i tipi di dati

A questo punto valuto in base alla distribuzione sotto l'ipotesi nulla e ci calcolo l'indice. Se il valore dell'indice è all'interno del quantile fissato (α) posso affermare che ci sia struttura nel dataset, se invece l'ipotesi è rigettata allora si sa che il clustering sviluppato non ha struttura pertanto può essere mostrata al proprio interlocutore l'algoritmo sviluppato.

15.4 Selezione del numero di cluster

A senso porsi questa domanda se si è prima rilevato che vi è struttura di clustering.

I criteri **relativi** è un uso particolare delle misure di clsustering per comprendere il numero ottimale di cluster. Si proietta lo spazio delle osservazioni in uno spazio di dimensione minore e da questo si utilizzano delle tecniche di valutazione.

Gli indici principali sono:

- Calinski and Harabasz: seleziono il numero di k che ottimizza il numero di cluster
- Dunn: simile a quello sopra
- Devies-Bouldin: minimizzo K per l'ottimo numero di cluster

Misure miste di probabilità model-based clustering sono:

- Akaike Information Criterion (AIC)

- Minimum Description Length (MDL)
- Bayesian Information Criterion (BIC)

Per l'algoritmo di clustering richiede un input di numero di clsuter K dagli utenti una sequenza di strutture di clustering ottenute eseguendo l'algoritmo motlre volte dove K spazia da K_{min} a K_{max}.

16 Lezione - Analisi di associazione

Con questa componente ritorniamo al concetto di causalità di variabili: dati certi valori di attributi cosa posso dire del valore di un altro attributo?

Le regole associative ci permettono di prendere delle scelte molto operative in diversi ambiti.

Problema chiave è quello del posizionamento di prodotti in un negozio. Si sa che all'acquisto di un certo prodotto si tende a acquistare altri prodotti connessi, quindi si cercano queste associazioni basandosi sul carrello della spesa dei clienti (market basket).

Obiettivo: identificare quali siano gli **item associati** per poter prendere delle decisioni.

Generazione di **Regole associative** o insiemi di **Item frequenti**.

es. {Beer} $\rightarrow$ {swiss cheese} ovvero {antecedente} $\rightarrow$ {conseguente}
non è una causalità ma un'associazione!!

Noi organizziamo il nostro dataset in modo da indicare se l'item considerato in un certo basket sia presente o meno, da lì possiamo fare le nostre valutazioni.

Consideriamo $I = \{i_1, i_2, ..., i_d\}$ il set di tutti gli item nel market basket, e $T = \{t_1, t_2, ..., t_N\}$ sia l'insieme di tutte le transazioni.

Una collezione di zero o più item è chiamata **Itemset**. Se un itemset contiene k item è detto **K-Itemset**.

La **transaction width** è il numero di item presenti in una transazione. Una transazione t_j conteien l itemset X, se X è un sottoinsiemen di t_j.

Per comprendere la frequenza degli item in una transazione si considera il **Support count**. Un itemset viene valutato anche dal numero di transazioni che lo contengono, conteggiato nel *SupportCount*:

$$\sigma(X) = |\{t_i | X t_i, t_i \in T\}$$

SBAGLIATA

Una **regola di associazione** viene rappresentata come:
$X \rightarrow Y$

dove X e Y sono itemset disgiunti ($X \cap Y = \emptyset$), una regola viene valutata in termini di supporto e confidenza.

Support determina quanto spesso una regola sia applicabile dato un data set:

$$s\{x \to Y\} = \frac{\sigma(X \cup Y)}{N}$$

Confidence determina quanto frequente Y è presente in una transazione che contiene X (si assume che l'universo sia rappresentato partendo da X):

$$c\{x \to Y\} = \frac{\sigma(X \cup Y)}{\sigma(X)}$$

Perchè utiliziamo il supporto:

- potrebbe esserci un'associazione casuale se troppo basso
- potrebbe non valere la pena seguire associazioni che si applicano in modo poco significativo dal punto di vista dei profitti

Supporto utilizzato per eliminare regole non desiderate e condivide interessanti proprietà che possono essere sfruttate per la ricerca di regole associative efficaci.

La confidenza è molto importante perchè un alta confidenza significa che Y sarà molto presente in transazioni con C e si stima la probabilità condizionata di Y dato X.

Association Rule Mining Problem:

dato un set di transazioni T, trovare le regole con support $\geq$ minsup e confidence $\geq$ minconf.

Approccio a forza bruta di association rule non è molto praticabile in quanto i tempi di computazione aumentano in modo esponenziale: $R = 3^d - 2^{d+1} + 1$.

Una strategia comunemente adottata in molti algoritmi è decomporre il problema in 2 grandi supertask: VEDI CHRISTIAN

16.1 Generazione di frequent itemset e generazione regole

Un itemset è frequente se supera la soglia fissata di minima frequenza per essere considerato tale. **Candidate Itemset**: ovvero l'insieme di tutti gli itemset che possiamo formare. Avranno un numero di item diverso.

Se procediamo con un approccio forza bruta per trovare gli **itemset frequenti** bisogna determinare il support count per ogni itemset candidato.

I confronti da effettuare sono nell'ordine di O(NMw), dove N = numero di transazioni, M = numero di itemset candidati e w = massima lunghezza delle transazioni.

È troppo come numeri confronti contando che molti dei quali sono inutili o poco significativi.

Vi sono due approcci per ridurre il costo computazionale della generazione di itemset frequenti:

- ridurre il numero di candidati itemset. Il principio Apriori è un metodo per eliminare alcuni candidati itemset senza contare il support count. Principio: se un itemset è frequente, allora tutti i suoi sottoinsiemi sono frequenti
- riduce il numero di confronti anzichè controllare tutte le possibili combinazioni, lo si fa con strutture dati avanzate

Il principio apriori si basa sul fatto che se una regola ha una frequenza bassa allora tutte le regole che prevedono come sottoinsieme la stessa non supereranno quella frequenza pertanto è inutile considerarle. Posso quindi fare **pruning** dell'albero per queste soluzioni, viene chiamato **support-based pruning**.

16.2 Algoritmo apriori

Genera due operazioni:

- Candidate Generation: genera nuvi candidati k-itemset basati su (k-1)-itemset frequenti calcolati nella precedente iterazione
- Candidate Pruning: questa operazione elimina alcuni candidati k-itemset usando la strategia del support-based pruning

La complessità computazionale soffre di 4 limiti:

- Support threshold: la soglia di supporto se troppo bassa non taglio molto l'albero, però non deve essere neanche troppo elevata.
- Numero di item (dimensionalità): se il numero di item cresce, più spazio ci sarà bisogno in memoria per registrare il support count degli item.
- numero di transazioni: e molto grande bisogna scorrere ogni volta tutta la lista
- avarage transaction width: quanto è mediamente densa la lista di item. La massima taglia di itemset frequenti tende ad aumentare la width delle transazioni media,

16.3 Rule Generation

Ogni k-itemset frequente, Y può generare al limite $2^k - 2$ regole di associazione.

Una regola di associazione può essere estratta partizionando itemset Y in 2 sottoinsiemi non vuoti: $X \to Y - X$ che soddisfa il threshold di confidenza.

Tutte le regole generate da itemset frequenti sono esse stesse frequenti.

In pratica, il numero di itemset frequenti prodotti da transazioni possono essre molto grandi. Per questo si ragiona in due rappresentazioni:

- Maximal Frequent Itemset
- closed Frequent Itemset

Maximal Frequent Itemset: verifica il vincolo della frequenza e definisce la frontiera dove lui non gode più di questa proprietà, massima frontiera in cui mi posso spingere. Lo è se lui è frequente e se tutte le sue estensioni non sono frequenti.
Fornisce una rappresentazione compatta dell'insieme di itemset che cerchiamo, la più piccola espressione in cui gli itemset sono derivabili. Il maximal frequent itemset è applicabile solo se si utilizzano algoritmi efficienti che cerca tutti e solo gli itemset massimali.

Per costruzione però non ci dice quanto è la sua confidenza rispetto ai suoi sottoinsiemi.

Closed Frequent Itemset: un itemset X è chiuso se nessun immediato sottinisme ha esattamente lo stesso support count di X. In ogni caso il suo spporto deve essere maggiore o uguale a minsup.

Importante qunado vi sono gruppi di prodotti venduti a blocco ignorando gli altri.

Questo tipo di itemset sono utili per rimuovere regole di associazione ridondanti.

Una regola di associazione $X \to Y$ è ridondante se esiste $X' \to Y'$ che rispetta certe proprietà. (vedi Christian)

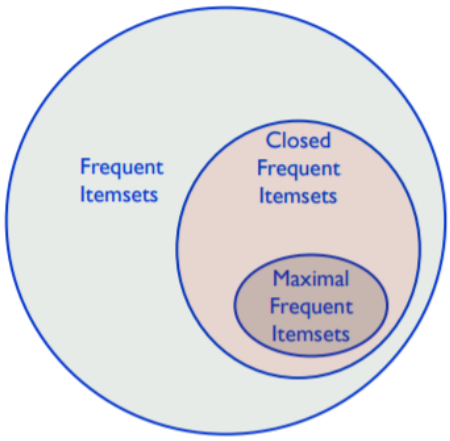

Figura 19: gerarchia dei frequent itemset