Appunti del corso di Machine Learning

Leonardo Alchieri

2019/2020

Indice

1 Informazioni generali

Il corso è organizzato in video lezioni, in cui spiega dei contenuti specifici. Sono video registrati con *unità didattiche*, con lezione di metodologia e hands-on.

1.1 Esame

L'esame è basato su un progetto, che dobbiamo svolgere. Vi è poi un esame il laboratorio, ovvero con i computer, ed è basato su un insieme di domande e quiz su cui ci si può allenare online. La votazione è organizzata come:

$$\texttt{grade} = 70\% \cdot \texttt{project} + 30\% \cdot \texttt{lab\_exam}$$

I dataset su cui si deve fare il progetto sono quelli di **Kaggle**[1] e deve essere fatto in un team dalle 2 alle 5 persone. Non siamo limitati a usare il materiale del corso: siamo invitati a usare tutte le risorse a disposizione.

Dobbiamo consegnare un **workflow KNIME** e un **techical report** di meno di 8 pagine.[2]

Il progetto presenta una serie di voci in base a cui viene valutato (si trovano tutte le informazioni sulla pagine del corso).

Si sconsiglia di sviluppare il progetto con *trial and error*: il progetto deve essere completo con quello indicato sul corso.

Per lavorare in gruppo, consiglia l'uso della piattaforma **SLACK**, e usarlo come unico contenitori del progetto.

L'esame scritto è dato da una risposta aperta e da 6 quiz. Gli argomenti su cui saranno fatte domande sono quelle indicate come un `*`, e sarà della durata di 40min.

Consegna progetto Per poter partecipare all'esame scritto in Laboratorio bisogna aver consegnato il progetto. Sebbene i due siano legati alla stessa sessione d'esame, si può consegnare il progetto anche prima.

Challenge Per la classifica, viene eseguita una *short list*, ovvero i migliori gruppi andranno a fare una presentazione.

1.2 Organizzazione del corso

Le lezione del corso sono già tutte disponibili online: vi è già tutto sulla pagine del corso. Il professore consiglia di guardare le lezioni del giorno, segnarsi le domande e poi venire in classe per chiarire tutti gli aspetti. Se ci sarà tempo, verranno ospitati dei seminari e laboratori tematici.

[1] Vedi online la lista di dataset da analizzare.
[2] Siamo invitati a produrre il materiale in inglese.

2 Introduzione al Machine Learning

Nella società moderna, una vasta quantità di ambiti, non strettamente più legati all'informatica, è in grado di produrre una grande quantità di dati: sia che esso sia *e-commerce*, *social networks* o anche finanza, sanità e agricoltura.

L'obiettivo di tecniche note come **Machine Learning** è quello di sviluppare strumenti e metodologie in grado di ricavare valore aggiunto da essi, soprattutto sotto forma di risultati non ottenibili da tecniche analitiche più tradizionali, come per esempio il *Data Mining*.

Nell'ambito di queste nuove tecniche, si possono distinguere 3 grosse macrocategorie:

- Modellizzazione predittiva o *supervised learning*, in cui si hanno dei dati che sono stati preclassificati.

  Si cerca di sviluppare un modello che, messo davanti a un nuovo dato, riesca a classificarlo correttamente.

- Machine Learning descrittivo o *un-supervised learning*.[3]

- *Reinforcement learning*, ovvero basato su un *premio-punizione* meccanismo.

Ci concentriamo sui primi due tipi di apprendimento mostrati, ovvero supervisionato e non-supervisionato. In particolar modo, si possono ulteriormente categorizzare.

- Supervised learning
  - Problemi di classificazione, ovvero si cerca in target discreto.
  - Problemi di regressione, ovvero si cerca un target continuo.
- Unsupervised learning
  - Analisi di cluster, ovvero mettere ordine tra le istanze fornite.
  - Analisi associativa, ovvero apprendimento delle regole.
    L'idea è quella di capire come oggetti vengano associati tra di loro.

2.1 Paradosso di Simpson

Se usiamo solamente i dati, senza alcun riferimento a quello che vi è a priori, si potrebbe commettere errori, ovvero trovare dei legami dove non ce ne sono[4] oppure dove non sono corretti. Dipende quindi molto dal tipo di analisi che ci si trova davanti.

Ci sono infatti diversi livelli di informazione che si possono identificare; in ordine di informazione, possono essere classificati come:

- Vedere
- Fare
- Immaginare

[3] Noto anche con *auto-supervised learning*.
[4] `https://www.tylervigen.com/spurious-correlations`

3 Data

3.1 Data types

Il primo passo per analizzare i dati è quello di identificare che cosa si sta manipolando e che tipo sono. Nei dati che si ha a disposizione ci possono essere dei dati "strani": per esempio, un `?` indica che il dato non è disponibile, per diversi motivi, e.g. non pervenuto, dati corrotti, errore umano, etc.

In un **dataset**, le colonne sono note come **attributi** e le righe come **istanze**, **casi** o **osservazioni**. È importante conoscere il tipo di variabile usata per in attributo, in maniera tale da evitare errori. Tra i diversi tipi:

- Nominali, su cui si possono solo fare operazioni di di uguaglianza, $=$ o $\neq$.
- Ordinali, su cui si può fare $=$, $\neq$, $<$, $>$.
- Intervallari, come sopra ma con somma e differenza.
- Razionali, su cui si possono fare tutte le operazioni matematiche.

Si possono ovviamente distinguere gli attributi anche in altra maniera:

- Discreti
  - Categorici
  - Numerici
  - Binari
- Continui

3.2 Esplorazione

Dopo aver identificato il tipo dei dati, si possono andare a fare delle statistiche semplici, e.g. media, deviazione standard etc. Il riassunto statistico sui dati ovviamente dipende a seconda degli attributi.

In un dataset possono esserci degli *outliers*, ovvero dei dati che sono molto differenti da tutti gli altri, e che quindi potrebbero modificare il calcolo di alcuni parametri importanti. Spesso si calcola una **media trimmed**, ovvero dove vengono esclusi i valori agli estremi; una sua versione più complessa è nota come **media winsorize**.

Un importante indicatori statistico è anche il **range**, calcolato spesso in aggiunta alla media, e indica quanto i dati siano dispersi. Anche in questo caso, se ci sono molti outliers, il range potrebbe fornire indicazioni fuorvianti. Per evidenziare bene dove si trova la maggior parte dei dati, si calcola il **range interquantile**, ovvero la differenza tra il primo e il terzo quartile.

Si calcola spesso la **varianza** dei dati. Si può inoltre calcolare la **mean absolute mediation**. Viene citata anche la **matrice di varianza covarianza**.

La **correlazione lineare** mostra quanto dei dati siano legati in maniera lineare tra di loro:

$$corr(X, Y) = \frac{cov(X, Y)}{\sqrt{var(x) \cdot var(Y)}}$$

I dati possono essere rappresentati graficamente, con diverse figure:

- Istogrammi
- Boxplot, che identifica facilmente gli *outliers*, oltre che la dispersione dei dati.

3.3 Missing replacement

Come detto, vi possono essere dei dati mancanti all'interno di un *dataset*; diverse tecniche consentono di trattare questi ultimi. Le ragioni per cui questi manchino possono essere diverse, come per esempio l'attributo non è stato misurabile, incomprensioni su come dovesse essere trattato, problemi tecnici, etc.

Si elencano alcune tecniche per trattare dati mancanti:

- *Record removal*, ovvero buttare un dato che ha un dato mancante. Questa tecnica è molto valida per dati con pochi attributi, ma diventa pessima se ci sono molti dati.
- *Imputazione umana*, ovvero si inserisce un dato a mano dove ci sia qualcosa di mancante. Questa tecnica è molto efficace, ma è molto lenta.
- *Costante globale*, che però risulta solamente un placeholder.
- *Sostituzione con moda*: si rimpiazza il dato mancante con la moda. Un'osservazione potrebbe essere una moda, ma non per forza molto probabile.
- *Sostituzione con la media*: simile al caso con la moda, ma nel caso in cui ci siano dati continui.
- *Sostituzione con media condizionale*: si sostituisce con la media condizionale tra dei dati specifici.
- *Più probabile*, ovvero si inserisce il dato più probabile

I dati che si hanno a disposizione possono variare molto a seconda del modello che si va a utilizzare, e questo cambia anche il modo in cui si sostituiscono dei dati mancanti.

L'operazione che si esegue per i dati mancanti impatta pesantemente quello che si va a fare dopo; spesso tecniche di modellazione considerano la presenza di dati mancanti.

3.4 Preprocessing - Parte 1

Ho spiegato come funzione KNIME.

Domande Una parte fondamentale dell'analisi dati è quella di prendere coscienza dei dati che si hanno a disposizione, oltre che i problemi che potrebbero nascere da essi. Spesso il tipo di dati che si ha davanti determina il tipo di domande che ci si possono porre.

Spesso si lavora con dei **dati osservazionali**, ovvero con i quali non possiamo interagire col sistema, i quali possono rispondere solamente delle specifiche domande. Nel nostro caso, tratteremo solamente modelli per estrarre informazione da *dati osservazionali*.

Preprocessing Il **preprocessing** è noto anche come *massaggiare i dati*: si tratta di applicare alcune modificazioni ai dati, dettate sia dal tipo di domanda che si deve formulare, sia dal tempo/spazio computazionale a disposizione. È fondamentale che il controllo sia da parte dell'analista, e non essere passivi di fronte a dati e software: prima avviene la domanda a cui si vuole rispondere.

Da questo parte il nostro "viaggio" per trovare delle risposte attendibili. I tipi standard di preprocessing sono:

- Aggregazione
- Campionamento
- Riduzione della dimensionalità
- Selezione di un sottoinsieme di variabili

Si ricordi che la frase *più roba c'è meglio è* non è vera: ci si deve considerare con quello solamente necessario per l'obiettivo che ci si pone.

Aggregazione Aggregare dei dati significa collassare alcuni dati insieme trasformandoli in un nuovo valore, secondo una qualche funzione. I vantaggi sono diversi:

- Riduzione del dataset, e quindi del tempo computazionale necessario.
- Cambiare la visione sui dati, ovvero passare da una visione locale a una più globale.
- Si riduce la *varianza*, spesso comodo per i vari algoritmi che si vanno ad utilizzare. Lo svantaggio è che vanno perse alcune particolarità specifiche di alcune situazioni.

Campionamento Spesso l'uso di tutta la popolazione di dati che si ha a disposizione non è sempre fattibile, magari a causa di limitazioni computazionali. Si usa quindi un *campionamento statistico*, ovvero si selezionano dei dati secondo delle tecniche specifiche.

Un campione è rappresentativo se ha le stesse proprietà e caratteristiche di tutta la popolazione: siamo sempre noi a indicare quale siano le caratteristiche che si cerca di preservare. Durante un campionamento si dovrebbe controllare se si sta procedendo in maniera corretta.

Tra i diversi tipi di campionamento, ci sono:

- Campionamento casuale semplice
  - Senza reintroduzione
  - Con reintroduzione.

    Se i dati a disposizione sono pochi, questo può fare una grande differenza nel campionamento.
- Campionamento stratificato.

  Si va deliberatamente a selezionare un campione che abbia delle particolari caratteristiche.

- Campionamento proporzionale.

  Non si fa altro che selezionare un campione che abbia la stessa proporzione che tutta la popolazione per una caratteristica.

Un campione più grande aumenta ovviamente la probabilità che il campione sia rappresentativo della popolazione: questo però non sempre può essere fatto. Un campione troppo piccolo può portare la perdita di alcune particolari caratteristiche.

3.5 Preprocessing - part 2

Riduzione dimensionale Ridurre il numero di attributi può essere utile per l'analisi dati; oltre che vantaggi computazionali, alcuni algoritmi possono produrre risultati erronei. Si ha anche un vantaggio di rappresentazione e visualizzazione dei dati: spesso si portano i dati in 2 dimensioni per riuscire a mostrarli bene.

Le tecniche di riduzione dimensionale usano degli algoritmi per passare da tante dimensioni a meno; si vanno a proiettare i dati da tante dimensioni su poche dimensioni.

Una delle principali metodologie in questo ambito è nota come **analisi delle componenti principali**: si tratta della ricerca e costruzione di nuovi attributi a partire da quelli presenti all'interno dei dati. In particolare, questo calcolo viene eseguito cercando la direzione di massima variazione (ovvero dove è massima la derivata). I nuovi attributi trovati come descritto qui saranno:

- una combinazione lineare degli attributi originali
- sono ortogonali tra di loro
- catturano la direzione di massima variazione dei dati

Questo tipo di algoritmo richiedere di specificare il numero di componenti principali che si vogliono analizzare o la percentuale di variazione che si vuole.

Binarizzazione Una utile trasformazione è la **binarizzazione**, ovvero trasformare degli attributi in stringhe di 0 e 1. In un dataset si ha che l'uso di variabili binarie ha senso solo nel momento in cui interessa avere informazione specifica solo su una caratteristica.

Discretizzazione Questa tecnica consiste nella trasformazione di variabili o funzioni continue in discrete. In particolare, essa può essere eseguita considerando o meno la presenza di altri attributi. Nel primo caso si parla di **discretizzazione supervisionata**, nel secondo di **non superviosionata**.

Tipicamente, per poter eseguire questo metodo di devono isolare i dati in intervallo, ovvero classi/categorie, sulla variabile (dimensione) di interesse. Nel caso che essa sia **non superviosionata**, gli intervalli discreti sul continuo sono presi di ugual grandezza, ove possibile. Viceversa, in presenza di altri attributi, ovvero quando si ha discretizzazione **supervisionata**, si va a calcolare un *livello di purezza*[5] dei possibili intervalli, con riferimento alle variabili esterne.

[5]Spesso questo è associato all'**entropia**, sebbene non sia obbligatoriamente l'unico parametro su cui eseguire il conto.

Nei casi in cui gli attributi categorici posseggono troppi valori: in questi casi si aggregano, in maniera differente:

- Se gli attributi sono ordinali, vanno bene le tecniche precedenti
- Se però sono nominali, non vanno bene: si possono raggruppare in sottocategorie, ovvero si creano dei nuovi attributi su cui si andrà ad eseguire l'analisi. Si possono altrimenti usare delle tecniche di ottimizzazione algoritmica.

Trasformazione di variabili È spesso conveniente modificare le variabili a disposizione per rendere più visualizzabili e/o più facili da trattare. Alcune tecniche sono:

- Applicazione di una semplice funzione.
- Standardizzazione.

4 Classificazione

Dopo aver preso visione dei dati che si hanno a disposizione, ovvero dopo aver fatto tutto quello descritto precedentemente.

4.1 Introduzione

È molto importante capire che cosa si sta facendo. A volte, si cerca di **prevedere** o **identificare** alcune caratteristiche all'interno del dataset con cui si lavora.

Modello di classificazione Si può vedere un **modello di classificazione** come una *black box* con uno o più ingressi e una o più uscite (dal numero di input e output che si usano varia solamente la complessità del modello), in grado di prevedere un certo evento/caratteristica.

La variabili che si forniscono in input sono note come **variabili esplicative** o di **input**: sono quelle che entrano nel modello e tramite le cui esso produce un certo *output* desiderato.

Un **modello di classificazione** è utile per due finalità:

- Modellizzazione descrittiva: si cerca di distinguere tra oggetti di differenti classi, e.g. scoprire se un cliente uscirà o meno dal suo piano tariffario.
- Modellizzazione predittiva: scoprire delle caratteristiche a partire da dei parametri non conosciuti, e.g. capire se un nuovo cliente deciderà di cambiare, e quando.

Test data La tecnica di classificazione è un approccio sistematico in cui, dato un insieme di dati noto come **training set** in cui è sempre osservata la **variabile di classe** (il nostro desiderato *output*), si costruisce, ovvero *apprendere* e *validare*, il nostro modello di classificazione. Questo verrà poi criticato e misurato su un nuovo insieme di dati, noti come **dati di test**.

Generalizzazione Se il modello funziona bene sui dati con cui è stato costruito, non ottengo niente: se invece funziona bene su dei dati che non ha mai visto, significa che è in grado di **generalizzare**.

Apprendimento Quando si sottopongono i dati ti addestramento, i *training data*, al modello, esso **apprende** alcuni particolari attributi. Una parte essenziale di questo procedimento è anche la **feature selection**, ovvero andiamo a selezionare solamente quei parametri che servono per l'apprendimento.

Inducer Una volta che il modello è stato addestrato e viene usato per prevedere delle nuove istanze, è noto come **inducer**.

Confusion Matrix Nel caso di classificazione binaria, la **matrice di confusione** è uno dei modi più efficaci per vedere come funzioni il modello. Nel caso binario, quando un modello predice può o sbagliare o dare un risultato giusto,

per entrambe le variabili a disposizione. Il numero di classificazioni corrette può essere messo in una matrice, come segue:

$$\begin{matrix} \texttt{true negative} & \texttt{false positive} \\ \texttt{false negative} & \texttt{true positive} \end{matrix}$$

La matrice ideale ovviamente non dovrebbe presentare alcun valore *falso*, nel qual caso significa che è perfetto.

Performance metric Si può definire la misura di **accuratezza**, definita come:

$$\texttt{accuracy} = \frac{\texttt{true negative} + \texttt{true positive}}{\texttt{number of predictions}}$$

ovvero mostra esattamente quanti dati sono stati classificati correttamente tra tutti i dati usati come addestramento.

Complementare all'accuratezza vi è l'**errore**, ovvero:

$$\texttt{error} = 1 - \texttt{accuracy}$$

4.2 Classification Techniques - Part 1

Classificatore I modelli di classificazione disponibili si possono suddividere in 4 categorie:

- Modelli **euristici**, come *decision trees*, *random forest*, *nearest neighbour*.
- Modelli su **regressioni**, come la *regressione logistica*..
- Modelli di **separazione**, come *support vector machines* e *artificial neural networks*.
- Modelli **probabilistici**, che si basano la statistica Bayesiana, come *naïve Bayes*, *tree-augmented naïve Bayes*.

Andiamo a fornire una overview generale su questi modelli.

Decision Trees Un albero di decisione consiste un nodo alla radice, il *capostipite*, che non presenta **archi** in ingresso e più archi in uscita. I nodi *interni* hanno solamente un arco in ingresso e due o più archi in uscita. Quando un nodo non presenta archi in uscita è noto come *nodo foglia*, o *terminale*.

Ad ogni nodo è associato un determinato schema. In emanazione di un nodo, si conduce un test, ovvero una domanda specifica, e.g. se una caratteristica è maggiore o minore di un certo valore.[6]

Quello che esegue con dei test univariati, che fungono solamente da soglie, si sta partizionando lo spazio degli eventi fino a quando non si riescono a separare le due caratteristiche. A seconda di dove si esegue la partizione, si ha ovviamente l'abilità di separare le due classi: la parte di **apprendimento** è la parte in cui si modificano le partizioni al fine di ottenere una corretta classificazione.

Per decidere dove andare a fare gli *splitting* si usa l'**entropia** o il **Gini index**, anche se si può usare anche l'errore sulla classificazione.

[6] Vedi schema sulle slide: si capisce molto bene che cosa succede.

Ovviamente questa tecnica funziona sia per splitting binari sia per splitting multipli, oltre che sul tipo di attributo, che esso sia binario, nominale o continuo.

I segmenti nello spazio lungo i quali di discrimina sono noti come **confini di decisione**.

Si noti come lo splitting non deve per forza essere univariato, ma anche bivariato o altri, e.g. separare per la somma. In questi casi si ottengono degli **alberi obliqui**.

Regressione Logistica Uno dei principali modelli usati per risolvere problemi di classificazione binaria, si basa sull'applicazione di una funzione ad un argomento che è la combinazione lineare degli attributi. Dato l'attributo output Y e l'attributo input X si calcola la probabilità a posteriori che la classe sia nulla, con la condizione che il vettore degli attributi assuma un particolar vettore.

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

dove w è il **vettore dei parametri**, appreso tramite i dati di training. Ovviamente vale anche:

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

4.3 Classification Techniques - Part 2

Support Vector Machine Le SVM appartengono alla categoria dei metodi di separazione.

Si consideri il dataset $D = \{(x_1, y_1) \dots (x_m, y_m)\}$, dove $x \in \mathbb{R}$ e $y \in \{-1, 1\}$. Ovvero si hanno dei dati a cui sono associati dei *labels*, e vogliamo usare un algoritmo per riuscire a separare, nello spazio dato, le varie categorie. Nel caso mostrato, si può scegliere una retta che separi lo spazio: essa dipenderà da un insieme di parametri, a seconda della modifica dei quali viene modificata la retta.

Se esiste una possibilità di separare i dati, ovvero nel caso in cui i dati siano *linearmente separabili*, esisterà una configurazione di parametri in grado di identificare la retta migliore.

Possono però capitare che vi siano più rette che rispondo alla stessa domanda, ovvero che riescono a discriminare tra le categorie proposte. Ovviamente, le diverse rette "corrette" corrispondo a una combinazione di parametri differente; in particolare, al fine della classificazione queste risultano uguali tra di loro.

In generale, si può trovare una molteplicità di rette che siano in grado di eseguire la classificazione. Ci si domanda quindi se queste rette siano tutte egualmenti buone, e, soprattutto, quale tra queste sia da scegliere. Siccome il compito non è solamente quello di separare i dati che si stanno usando per apprendere, è anche quello di fornire delle classificazioni robuste su dei nuovi dati: sebbene esse siano tutte uguali tra loro per il *training set*, non è detto che lo stesso valga per il *testing set*.

Data una retta, si può definire il livello di errore che essa può commettere quando sceglie tra le categorie fornite. Ovvero, si determina quanto il margine entro in cui il classificatore possa sbagliare. Se il criterio risulta la robustezza, si preferisce una retta con un margine di errore maggiore: si cerca di allontanare il più possibile le istanze delle due categorie.

Da questi concetti, è stata ricavata una formulazione matematica, in modo tale da trovare l'iperpiano ottimale. In termini matematici, si seleziona la retta che *massimizza* il margine, ovvero la distanza che intercorre tra due rette parallele alla retta scelta equidistanti; la semidistanza tra la retta scelta e le separatrici è la distanza tra la retta e il punto di una delle due classi più vicino.

Una **Support Vector Machine** calcola la funzione:

$$h(x) = \begin{cases} +1 & \text{IF } \underline{w} \cdot \underline{x} + \underline{b} \geq 0 \\ -1 & \text{OTHERWISE} \end{cases}$$

dove la retta $\underline{w} \cdot \underline{x} + \underline{b} = 0$ è la retta che divide il piano. L'argomento della SVM è sull'asse reale, ma la funzione h giace su due piani paralleli al piano in cui giacciono i punti.

Si cerca di minimizzare il reciproco della norma di $\underline{w}$, che è la retta separatrice:

$$\min_{\underline{w},\underline{b}} \frac{1}{2} \underline{w} \cdot \underline{w}^{\intercal}$$

con vincoli, per ogni dati i-esimo del *training set*:

$$y_i(\underline{w} \cdot \underline{x}_i + \underline{b}) \geq 1, \quad \forall i = 1, \ldots, m$$

Si noti, che la definizione della distanza tra le due rette separatrici, ovvero il margine, è

$$\frac{1}{||\underline{w}||}$$

ovvero si cerca di massimizzare la metà di questa quantità, come descritto sopra.

Noi non andiamo a risolvere questo problema di ottimizzazione, essendo un problema quadratico, ma la sua **formulazione duale**.

In alcuni casi, inoltre, non esiste mai una allocazione di una retta che riesca a distinguere i dati tra di loro: ovvero i dati non sono linearmente separabili. Ovvero, il problema formulato non ammette soluzione, in quanto i vincoli non saranno validi per tutti i punti dell'iperpiano. Quello che si può eseguire è usare delle **linear soft margin**, che riformulano il problema aggiungendo un *termine di penalizzazione* ξ:

$$\min_{\underline{w},\underline{b},\xi} \frac{1}{2} \underline{w} \cdot \underline{w}^{\intercal} + \Delta \sum_{i=1}^{m} \xi_i$$

s.t.

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

con $\xi_i \geq 0$.

L'altra soluzione si basa sull'uso di **funzioni non lineari**. Come idea, ci si mantiene la stessa struttura matematica ma cercando una *trasformazione* ϕ degli attributi che porti dallo spazio originale a uno nuovo in cui vi sia separazione lineare. In pratica, si usano delle **funzioni kernel** definite come:

$$K(\underline{u}, \underline{v}) = \phi(\underline{u}) \cdot \phi(\underline{v})$$

Esistono delle funzioni kernel a seconda del problema che ci si trova davanti, tutte classificate in letteratura.

4.4 Classification Techniques - Part 3

Percettrone Un modello **multi layer parceptron** è fatto da molti neuroni artificiali organizzati in maniera unidirezionale, che partono da un dato di input per arrivare a un output. L'elemento base di un modello MLP è fatto da neuroni, a cui sono associati dei parametri)*pesi*), associato a una singola connessione. Ogni neurone calcola una combinazione lineare dei parametri di input e produce un risultato.

A ogni neurone è associato a una funzione di **attivazione**, che prende il risultato della combinazione lineare e produce un output.

Ogni singolo neurone quindi calcola un valore

$$y_j = f\left(\sum_{i=1}^{n} w_{i,j} \cdot x_i - \theta_j\right)$$

dove f è la funzione di attivazione, $w_{i,j}$ il parametro di input dal i-esimo neurone.

Storicamente, sono state usate come funzioni di attivazione sono del tipo *tengenti iperboliche* o *logistiche*; a oggi, molte delle reti **deep** usano delle funzioni non derivabili, come le *ReLu*, acronimo di *Rectified Linear Unit*.

Tutto quello che si osserva è l'insieme dei valori di input e il valore che assume la variabile di classe Y. In particolare, tra l'input e l'output della rete vi sono una serie di **neuroni nascosti**, ovvero i cui risultati non sono esplicitamente visibili. Si identifica un network in cui sono collegati tra loro tutti i neuroni come **fully connected network**.

Quello che si può scegliere sono le *funzioni di attivazioni*, il numero di neuroni e il numero di strati nascosti. Le reti profonde, in particolare quelle *dense*, possono prevedere fino a centinaia di strati nascosti. In particolare, la funzione di **abckpropagation** calcola la derivata delle funzioni di attivazione, le confronta con l'output del network e modifica il parametri di conseguenza.

Il problema delle reti deep con funzioni di attivazioni tradizionali, la funzione di propagazione, dopo una serie di iterazioni, andava ad azzerare il gradiente della funzione (o viceversa esplodeva). L'invenzione delle *ReLu* non portano a questo problema, ed è sempre possibile calcolare dove la funzione aumenta o diminuisce.

Il problema di ottimizzazione che si va a risolvere è *non-convesso*, ovvero non si sa ancora risolvere, ovvero si possono solo calcolare massimi o minimi. Oggi, si procede a tentativi per risolvere questo problema: si sbaglia molto rapidamente, e si correggono gli errori fino a quando non si giunge a una qualche convergenza.

4.5 Classification Techniques - Part 4

Naïve Bayes

Classificatori Probabilistici Esso fa parte dei **classificatori probabilistici**: essi calcolano la probabilità condizionata, una volta che si ha avuto accesso ai valori che assumono le variabili attributo. Ovvero, secondo il **teorema di Bayes**:

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

dove $P(Y)$ è la probabilità a priori della classe Y; $P(\underline{X}, Y)$ è la **verosomiglianza**, ovvero ipotizziamo di conoscere Y e cerchiamo $\underline{X}$; mentre $P(\underline{X})$ è la probabilità osservata dell'evento.[7]

Il fenomeno di apprendimento consiste nel calcolare tutte le probabilità condizionate, e determinare quella più probabile; si esegue quindi questo processo in maniera ripetitiva. Si noti come, per eseguire un semplice confronto, non è richiesta la conoscenza di $P(\underline{X})$.

Indichiamo

$$\theta_{ki} = P(\underline{X} = x_k | Y = y_i)$$

con $k = 2^n,\ n \in \mathbb{N}$ e $y_i \in \{-1, 1\}$. In particolare, n indica il numero di parametri che sarebbero da stimare: si può vedere come, in presenza di n già sopra il 10, sarebbe necessario un gigantesco quantitativo di dati.

Per fortuna, si può fare un'ipotesi di **indipendenza tra gli attributi quando è nota la variabile di classe**; ovvero si calcola l'indipendenza condizionale. La relazione che vale in questo caso è:

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

Assumendo questo per tutti gli attributi, condizionatamente alla variabile di classe, fa si che valga:

$$P(X_1, \ldots, X_n | Y) = \prod_{i=1}^{n} P(X_i | Y) \tag{1}$$

Poiché indipendenti, le configurazioni possono essere calcolate localmente: si devono calcolare solamente $k = 2n$ parametri. Il modello **naïve Bayes** è infatti in grado di calcolare classificazioni con moltissimi attributi.

Come funzione NB In particolare, il classificatore **Naïve Bayes** calcola:

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

Si cerca quindi quale tra queste probabilità sia massima, ovvero

$$\arg\max_{y_k} P(Y = y_k | X_1, \ldots, X_n)$$

Questo tipo di modello è noto come **modelli grafico-probabilistici**, basati sull'uso di reti Bayesiane.

Si noti inoltre come la probabilità fornisce anche una misura di quanto sia attendibile il calcolo che si sta eseguendo: distinguere su un 51% è molto meno sicuro rispetto a un 99%.

Dati per NB Il **Naïve Bayes** può essere fatto sia su attributi categorici, sia su attributi numerici. In particolari, nel secondo caso si dovrebbe andare a discretizzare i dati; nel caso in cui questo non fosse possibile/preferibile, si può utilizzare una **densità di probabilità per la classe condizionale**, e.g. una distribuzione normale.

[7] Vedi slide di probabilità per chiarimenti

Rete Bayesiane Una generalizzazione di un classificatore *naïve* è una **rete Bayesiana**. Essi si basano su una serie di nodi che interagiscono tra di loro per ottenere un risultato. Ogni nodo prende in considerazione tutte le possibili configurazioni dei genitori. Si noti come queste reti *non ammettono cicli*.

Per risolvere classificazioni, si usa dei modelli particolari: si calcola la probabilità condizionata di Y date tutte le configurazioni; queste probabilità sono facilitate da alcuni particolari algoritmi.

Il bello delle reti Bayesiane è che riescono a funzionare perfettamente anche in mancanza di dati mancanti: si possono comunque calcolare i valori della probabilità anche senza conoscere i valori di input.

Dentro le reti Bayesiane vi è "scritto" quale coppie di variabili sono indipendenti rispetto a quale altro insieme di variabili.

Tree Augmented Naïve Bayes Esistono diversi tipi di *reti Bayesiane*: una di queste è un **tree augmented naïve Bayes**. Ovviamente si potrebbero imparare strutture diverse, ognuna per le classi fornite: il suo vantaggio è che riesce a riconoscere attributi. Quando si è noto lo stato di X_i, conoscere un suo discendente non fa cambiare la probabilità di Y.

Il vantaggio del modello è che suggerisce come ingegnerizzarsi: si identificano facilmente tutte quelle cose che non servono.

Esistono poi diverse variazioni di questi tipi di algoritmi, e.g. network con nodi nascosti.

4.6 Regressione Lineare

Modelli di stima Si hanno $n+1$, di cui n sono attributi candidati e 1 è l'attributo target. La realizzazione delle variabili indipendenti n è x_i e quella target y_i. Ciascuna di queste possiede m dimensioni. Si ipotizza una funzione $f : \mathbb{R}^n \to \mathbb{R}$ tale che:

$$Y = f(X_1, \ldots, X_n) + \varepsilon$$

dove Y e $X_1, \ldots, X_n$ sono le variabili aleatori a cui sono associati i risultati m esimi. In particolare, si ha un errore stocastico ε.

Regressione Lineare Il modello più elementare è la regressione lineare, ovvero del tipo:

$$f(X_1, \ldots, X_n) = \sum_{i=1}^{n} w_i X_i + b$$

dove b identifica l'intercetta e w_i sono i coefficienti angolari per le n variabili aleatorie.

Regressione Lineare Semplice Nel caso $n = 1$ si ha il modello più semplice di tutti, molto utile da un punto di vista teorico (ma non tanto pratico). Sarà del tipo:

$$f(X) = wX + b + \varepsilon$$

Si vanno a scegliere i parametri w e b tale che:

$$\min_{w,b} SSE = \min_{w,b} \sum_{i=1}^{m} \varepsilon_i^2 = \sum_{i=1}^{m} [y_i - wx_i - b]^2$$

Regressione Multivariata Nel caso si abbiano più variabili, si hanno modelli di regressione multivariati, del tipo:

$$Y = w_1 X_1 + \ldots + w_n X_n + b + \varepsilon$$

dove non sono presenti termini di interazione (detti a *una via*, *due vie* etc.).

In generale, si possono avere dei modelli anche più complessi: dipende da noi il grado di complessità del modello lineare semplice. Maggiore è il grado che si usa, si può entrare in *overfitting*.

Si definiscono **gradi di libertà** il numero di parametri che si deve andare a stimare.

Ipotesi dei residui Una volta costruito il modello, si deve verificare che le ipotesi eseguite sul modello siano vere. Queste coincidono sulle ipotesi fatte su ε, che deve soddisfare:

- Errore casuale
- Indipendenti e identicamente distribuiti.
- Distribuito normalmente
- Omoschedasticità.

L'obiettivo è ottenere $E(\varepsilon_i|x_i) = 0$ e $Var(\varepsilon_i|x_i) = \sigma^2$. Il modello di regressione è tanto più accurato tanto è meglio la deviazione standard.

4.7 Train/validation/test and features selection

Regolarizzazione Alcuni modelli di machine learning permettono degli *iperparametri*, e.g. reti neurali. Si ricordi che in una NN, si ha:

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

Il parametro λ si deve scegliere: viene spesso ottimizzato su un'altra porzione di dati, che non fa riferimento al dataset di addestramento. Può comunque capitare che si vada in *overfitting* del parametro λ: per ovviare questo, si usa un **validation set** per addestrare l'iperparametro.

5 Performance

Trattiamo del problema di usare quello che si ha a disposizione per apprendere, e capire come migliorare l'addestramento e la classificazione.

5.1 Performance Evaluation - Part 1

Accuratezza Stimare l'**accuratezza** di un modello di classificazione non è sufficiente ed efficace: si usa come stima quella calcolata su un altro batch di dati.

Overfitting & Underfitting Se si "spinge troppo" si rischia di creare un modello che entra in *overfitting*, oppure "troppo conservativi", ovvero *underfitting*.

Errore L'errore sui dati di apprendimento non è la migliore delle stime che si possono usare; l'**errore di generalizzazione** non si può mai conoscere, e la sua migliore stima è data dall'errore sui dati di testing. Un buon modello deve infatti riuscire a lavorare bene con dei dati che non ha mai visto, ovvero con dati come quelli del *testing set*.

Si noti che, in alcune istanze, ci potrebbero essere dei duplicati di dati di apprendimento anche nel set di testing: in tal caso il risultato sarebbe erroneo.

Un buon modello di classificazione dovrebbe avere sia un basso errore di training and di apprendimento.

Overfitting Quando un modello ottiene un ottimo risultato sul *training set* ma sbaglia su quello di *testing*, si dice che è in **overfitting**. In questo caso, significa che il modello non riesce a generalizzare: quello che si cerca è un modello che funzioni allo stesso modo qualsiasi dati gli siano forniti, ovvero in maniera analoga su *training* e *testing set*.

Underfitting Una posizione nella quale uso solamente i dati di *testing* per confrontare il modello, si va a creare un modello troppo semplice che fa male sui dati di apprendimento. Se si fa male sia sui dati di *test* e *training*, significa che la **complessità** del modello non è sufficiente per il problema mostrato.

5.2 Performance Evaluation - Part 2

Come risultato, in un classificazione si vuole sviluppare e/o selezionare un modello che garantisca la migliore prestazione possibile su dei nuovi dati. I modelli di classificazione sono confrontati in base a:

- Accuratezza
- Velocità di esecuzione
- Robustezza, ovvero poco sensibile alla presenza di rumore nei dati
- Scalabilità, ovvero la quantità di dati che riesce a usare
- Interpretabilità

Accuratezza L'accuratezza misura la capacità del modello di fornire misure attendibili del modello di fornire risultati corretti. Consente di selezionare qual è l'istanza che è in grado di garantire la migliore prestazione, indipendentemente dal compito che si deve svolgere.

Definendo con D_T in *training set* e D_{TS} il *testing set*. Ovviamente, è importante che $D_T \cap D_{TS} = \emptyset$. Un buon indicatore dell'accuratezza è quello che si ricava coi dati di test. Indicando con y_i il valore della classe associato all'istanza $x_i \in D_{TS}$, si calcola la **funzione di perdita** (*loss function*):

$$L(y_i, f(x_i)) = \begin{cases} 0 & y_i = f(x_i) \\ 1 & y_i \neq f(x_i) \end{cases}$$

e si calcola l'**accuratezza** come:

$$acc[D_{TS}] = 1 - \frac{1}{\nu} \sum_{i=1}^{\nu} L(y_i, f(x_i))$$

dove ν è il numero di dati nel *testing set*. Ovviamente, l'uso di questa funzione di perdita **non considera come si sbaglia**. Inoltre, l'errore risulta completamente simmetrico: sbagliare in un modo o nell'altro è indifferente.

Si definisce l'**errore** come:

$$err[D_{TS}] = 1 - acc[D_{TS}]$$

Velocità La prima accezione del termine velocità è quella del tempo necessario per l'apprendimento del modello. Insieme allo spazio di memoria, è un vincolo molto meno importante, grazie ad avanzamenti nella tecnologia.

Ci sono comunque alcuni algoritmi che, per funzionare, devono mantenere in memoria tutto il dataset, ovvero non possono usare istanze di *caching*. Nel caso questo non sia possibile, si esegue un **campionamento**, eseguito possibilmente in maniera tale da produrre risultati ottimali.

Robustezza Un algoritmo è tanto più robusto quanto è resistente alla presenza di **outliers**, **missing data** o la **variazione dei dati** in generale. Bisogna prendere coscienza del fatto che spesso i dati sono sporchi e anch'essi soggetti a deperimento.

Scalabilità È la capacità e propensione ad apprendere da una grande quantità di dati, ovvero come cambia all'aumento del numero di dimensioni (categorie).

Interpretabilità Detta anche **trustworthyness**, è una classificazione complessa. Noi umani apprendiamo le cose in maniera tutto che quantitativa, e quindi è difficile capire esattamente che cosa fa un algoritmo. A seconda dell'interlocutore che si ha davanti, si deve riuscire a spiegare come funziona la tecnica in maniera diversa, e la spiegazione più corretta non è detto che sia la più efficace.

Holdout Consiste nel tenere da parte una porzione del dataset $D = D_T \cup D_{TS}$, ovvero in una parte di *training* e una di *testing*. Consiste nel limitare la quantità di dati che si usano per l'addestramento, al fine di vedere come funzioni su dei dati nuovi. Si esegue quindi la stima sull'accuratezza tramite i dati di test; si tenga conto che a scelte diverse, si potrebbero ottenere dei risultati diversi.

Iterated holdout Altre tecniche che si possono usare sono note come **interated holdout**, nella quale ci si "immunizza" dalla sfortuna delle scelta. In questo caso, si prende il dataset e si ripete il partizionamento più volte: si conducono r fasi di apprendimento, e si confronta tra loro le accuratezze. Ovviamente, il costo computazionale risulta più alto.

Questa tecnica, ovviamente, funziona meglio del semplice *holdout*. Si deve tenere conto che, in presenza di una grande quantità di dati, anche un semplice *holdout* può funzionare bene.

Questo metodo non garantisce che i dati di apprendimento siano sempre bene rappresentativi: si può avere quindi un forte bias. In presenza di outliers, si potrebbe andare "fuori strada".

Cross validation Si tratta di un particolare tipo di **iterated holdout**, in cui si pongono alcuni restrizioni. Si divide il dataset in k sottoinsiemi mutualmente esclusivi, $D_1, \ldots, D_K$. Anche in questo caso si possono scegliere diverse istanze.

Si va quindi a svolgere K fasi di apprendimento; nella k-esima in cui si usa D_k come test e tutti gli altri per l'apprendimento. Ovvero:

$$D_{T,k} = \{D_1, \ldots, D_{k-1}, D_{k+1}, \ldots, D_K\}$$

e

$$D_{TS,k} = D_k$$

L'accuratezza viene stimata come **media delle accuratezze**: i *fogli* che si vanno a formare hanno, approssimativamente, la stessa cardinalità.

$$acc[D] = \frac{1}{K} \sum_{k=1}^{K} acc[D_k]$$

In letteratura, si usano spesso $K = 3, 5, 10$. Questo tipo di tecnica è particolarmente utile in presenza di pochi dati e molte caratteristiche.

La forma più estrema di cross validation si parla di **Leave One Out Cross Validation**, ovvero in cui si ha che ogni sottoinsieme ha un solo elemento.

In presenza di situazioni critiche, ovvero in cui una delle tue classi è sottorappresentata, si possono avere delle partizioni che presentano solo una caratteristica. Per ovviamente questo problema, si usa **campionamento stratificato**, ovvero si garantisce che in ogni *foglio* vi sia una proporzione uguale rispetto a tutto il dataset. Questo tipo di campionamento non è detto essere sempre possibile.

6 Class imbalance problem

6.1 LEZIONE MANCANTE

6.2 Counting the cost - Part 1

Errori diversi A seconda del problema, si possono sempre avere due errori differenti. Per esempio, in un piano binario, con 0 e 1, i due diversi errori (mettere 0 quando è 1 e viceversa) potrebbero avere un costo differente; uno dei due errori può infatti essere molto più grave rispetto all'altro.

Il **costo** deve essere tenuto in considerazione quando si confrontano modelli.

Come calcolare il costo Data una matrice di confusione, si confronta con una **matrice dei costi**. Si moltiplicano quindi le due matrici, e si ottiene il **costo** indotto sul modello.

6.3 Counting the cost - Part 2

Cumulative gains Si confronta il modello rispetto alla scelta casuale. In particolare, le *cmulative gains* sono i risultati positivi a seconda della percentuale del campione totale fornita come training.

Lift Chart Per rappresentare i cumulative gains, si può usare una *lift chart*.

Curva ROC Sull'asse delle x c'è la percentuale di *falsi positivi* disposti a sopportare, mentre sull'asse delle y vi sono i *veri positivi*.

6.4 Non binary classification

Mi manca la lezione precedente

L'attributo di classe, in questi problemi, può assumere una quantità più elevata di valori. A seconda dei casi, si può avere a che fare con **multi classe** (solo una) oppure **multi etichetta** (ci possono essere più risultati per un individuo).

Multi classe Nel caso si abbia multi classi, ci si può ridurre a tante classificazioni binarie del tipo **uno contro tutti**, ovvero si confronta ciascun caso con tutti gli altri (ovvero si pongono delle domande logiche *aut aut*). Si sviluppano quindi tanti classificatori differenti, e si va ogni volta a usare lo specifico modello di classificazione.

Le probabilità non vengono normalizzate, e si deve prendere una scelta progettuale per determinare una soglia per dire se sia presente o meno la classe. In questo caso, più classi potrebbero superare la soglia, sebbene non sia un problema *multi etichetta*.

7 Clustering

7.1 Introduzione

L'obiettivo è quello di raggruppare istanze, in maniera tale che individui nello stesso gruppo siano tra loro simili, e quelli in altri gruppi differenti. Si hanno due macro-finalità: la **comprensione** e l'**utilità**.

Per la comprensione, si cerca di trovare dei gruppi di oggetti che condividano delle caratteristiche comuni. Si possono quindi costruire delle gerarchie.

Da un punto di vista dell'utilità, si cerca di riassumere le principali caratteristiche degli elementi all'interno di un gruppo. Quando si hanno troppe istanze, è più facile illustrare le loro caratteristiche in comune, e in particolare concentrarsi su che cosa li rende simili in un gruppo. L'obiettivo è quello di trovare dei *prototipi*.

Gli oggetti che sono all'interno di un gruppo devono essere simili a quelli dentro li stesso e differenti a quelli fuori. Maggiore è la differenza tra gli elementi che stanno in gruppi diversi, migliore sarà la realizzazione. Il problema principale però è quella di definire che cosa si intenda per "**simile**".

Ci possono essere diversi tipi di *clusterizzazione*:

- A esclusione
- Partizionale
- Completa

Si può partire dalla dicotomia *partizionale/gerarchica*. Nella prima, si ha che l'insieme delle classificazioni vengono divise in *insiemi disgiunti* Nel secondo caso, in quella gerarchica si separa in gruppi, che contengo dei sottogruppi e così via; per trovare i diversi gruppi, si modificano semplicemente i requisiti di accettazione.

Nella dicotomia *completa/parziale*, si hanno da una parte l'associazione forzata a qualche cluster, dall'altro si possono lasciare delle osservazioni *outliers*, ovvero rimangono a se stanti.

Ci sono diversi tipi di cluster con cui si potrebbe avere a che fare.

- Cluster ben separati
- Cluster a prototipi
- Cluster a densità
- Cluster basati su grafi.

Il primo compito quando si cerca di eseguire *clustering* è quello di valutare quale attributi utilizzare: chiarire quale sia la domanda di ricerca, e che cosa si debba misurare. Si deve poi scegliere quale algoritmo di clustering e quale misura di similarità/prossimità che si vuole usare. Quando si è specificato, si deve quindi chiedere quale sia il criterio di merito, e.g. massimizzare il grado di omogeneità nel gruppo.

Il fatto che l'algoritmo abbia formato dei gruppi, non è detto che sia effettivamente vero: si deve eseguire *validazione* del cluster. Si investiga l'ipotesi

fondamentale, e vedere se un'allocazione puramente casuale possa avere una caratteristica significativa.

Una volta garantito che il raggruppamento sia sensato, si possono interpretare i risultati come qualcosa di effettivamente utile.

7.2 Prossimità - Parte 1

Similarità Quando si cerca di caratterizzare in termini formali che cosa significa *essere simile*, è abbastanza difficile applicare una definizione. Dipende tutto da quello che si sta cercando in una valutazione di *similarità*, ovvero quale sia l'obiettivo del paragone.

Valori elevati della caratteristica **similarità** sono quindi ciò su cui ci basiamo per mettere insieme oggetti. Per comodità, so possono usare delle probabilità, ovvero tra 0 e 1. In generale si attribuirà 0 se i due oggetti sono completamente differenti per l'ambito che si cerca e 1 se invece sono esattamente uguali.

Dissimilarità Analogamente, si può definire l'opposto della *similarità* come **dissimilarità**, ovvero quanto due oggetti sono differenti. Anche in questo caso, è comodo usare un valore compreso tra 0 e 1, spesso applicando delle banali trasformazioni lineari.

Problemi per mappare Quando si passa da misure generiche per andare ad avere valori nell'intervallo $[0, 1]$, si possono avere delle complicazioni. In particolare, in alcuni casi una trasformazione non lineare potrebbe essere necessaria: in tal caso, si avrebbero dei cambiamenti nel clustering.

Come esempio, vedere che cosa succede con la trasformazione:

$$d' = \frac{d}{1+d}$$

dove in questo caso d indica la *dissimilarità*. In questo caso, si ha quindi che i valori vengono distorti, ovvero introduzione di un *bias* nell'analisi. Questo non è sempre detto che sia un qualcosa di negativo: vado ad *allontanare* le cose più diverse e *avvicinare* di più le cose uguali. La distorsione può quindi essere introdotta deliberatamente.

Analogamente si possono avere dei "problemi" anche quando si passa da s a d. Anche in questo caso, si possono usare dei differenti tipi di trasformazioni, e.g. $d = 1 - s$ o $d = -s$.

Valutazione di similarità Quando valuto se due oggetti sono prossimi o meno, vado a prendere tutti gli attributi e vedo come l'attributo di un oggetto è simili o dissimile a quelli di un altro.

Iniziamo per semplicità che le nostre variabili siano caratterizzati da un singolo attributo. A seconda dei tipi di dati, si possono applicare diverse misure, in particolare a seconda che essi siano *nominali*, *ordinali* e *cardinali*.

Nominali Nel caso di attributi nomali, si definisce la dissimilarità come:

$$d = \begin{cases} 1 & x = y \\ 0 & otherwise \end{cases}$$

In presenza di più valori, si potrebbe obiettare sulla distanza, e.g. nero e verde sono *più diversi* di marrone e nero. Rimane comunque sempre da identificare l'obiettivo.

Ordinali In presenza di un singolo attributo ordinale, le cose sono leggermente più complesse. La misura che viene suggerita valuta la differenza in valori tra di giudizi pesata sul numero di giudizi:

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

Questo calcolo implica implicitamente la sostituzione dei valori ordinali con dei numeri interi, per i quali si può eseguire il calcolo.

Numerico In caso di attributo numerico, da una dissimilarità del tipo $d = |x - y|$ si può poi ottenere la *similarità* attraverso diversi tipi di trasformazioni, e.g. $s = -d$, $s = e^{-d}$ e altre.

7.3 Prossimità - Parte 2

Distanza di Minkowski Le dissimilarità tra alcune proprietà può essere calcolare tramite la *distanza*, ovvero una funzione del tipo:

$$d(x, y) = \left(\sum_{i=1}^{n} |x_i - y_i|^r \right)^{1/r}$$

Nel caso in cui la distanza verifichi proprietà di non-negatività, simmetria e la disuguaglianza triangolare, la misura è norma. Purtroppo, nel calcolo della dissimilarità, capita spesso che l'ultima proprietà non viene soddisfatta.

In una situazione in cui si calcolano le distanze, si utilizza il **coefficiente di similarità**.

Simple Matching Coefficient Valido in cui gli unici valori possibili sono 0 o 1.

$$SMC = \frac{n_{match}}{n_{tot}}$$

dove n_{match} è il numero di attributi corrispondenti e n_{tot} il numero totale di attributi che combaciano.

Coefficiente di Jaccard Simile a prima, ma in cui ci si focalizza nelle posizioni di realizzazione (1). Si ha:

$$J(x, y) = \frac{n_{match}}{n_{tot} - n_{00}}$$

dove n_{00} è il numero di coppie che presentano entrambi realizzazioni negative.

Coefficiente di Tanimoto Una versione più estesa del coefficiente di Jaccard è la seguente:

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

che vale nel caso di più attributi. In presenza di attributi binari, coincide con il coefficiente di Jaccard. Una misura simile viene data anche dalla misura della similarità del coseno.

7.4 Prossimità - Parte 3

Similarità di Coseno Definito come:

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

Soprattutto usata nell'*informatio retrieval*, assume valori su $[0, 1]$. In particolare, se $cos(x, y) = 1$ significa che i due vettori sono uguali, a meno di un fattore di scala.

Correlazione Classica correlazione di Pearson,

$$corr(x, y) = \frac{cos(x, y)}{stdv(x) \cdot stdv(y)}$$

In particolari casi, in cui magari i valori misurati sono molto rumorosi, si può usare la misura della **correlazione di Spearman**.

Problemi In generale, ci si possono porre delle domande importanti:

- Come trattare attributi su scale diverse e/o sono correlati.
- Come calcolare dissimilarità se ci sono attributi di differente tipo.
- Come calcolare la prossimità in presenza di diversi pesi.

Scale differenti In questi casi, se si calcolasse la distanza con una semplice *distanza Euclidea*, si otterrebbero dei valori dominati dagli attributi con più larga scala. Per ovviare al problema, si può eseguire una normalizzazione.

Attributi correlati In questo caso, la similarità può non essere solamente la distanza tra gli attributi, ma la solo correlazione. Per questo caso, si usa dunque la **distanza di Mahalanobis**:

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

Tipi differenti In questo caso, si calcola per ogni attributi k-esimo la similarità per la sua categoria, usando la specifica misura; eseguito questo, si usa una variabile indicatrice δ_k che è 0 se l'attributo è *asimmetrico*, oppure se c'è un missing value in uno dei due, e 1 altrimenti.

Dunque, la similarità risulta:

$$s(x, y) = \frac{\sum_{k=1}^{n} \delta_k s_x(x, y)}{\sum_{k=1}^{n} \delta_k}$$

Pesi differenti In questo caso, al posto che usare il coefficiente δ_k, si associa a ciascun attributo un valore w_k per il peso, e quindi:

$$s(x,y) = \frac{\sum_{k=1}^{n} w_k \delta_k s_x(x,y)}{\sum_{k=1}^{n} \delta_k}$$

Anche la distanza di Minkowski deve essere di conseguenza modificata:

$$d(x,y) = \left(\sum_{k=1}^{n} w_k |x_k - y_k|^r \right)^{1/r}$$

7.5 Algoritmi di Clustering - Parte 1

Algoritmi su prototipo Nel caso di clustering basato su prototipo, si possono classificare in base alla similarità del prototipo che descrive un certo cluster. Ogni cluster può essere identificato da un centroide; si calcola quindi la distanza, come misura di dissimilarità, tra a un singolo punto e i diversi centroidi.

Gli algoritmi su prototipo si basano su alcune premesse. In particolare, la singola variabile può appartenere o a un cluster oppure a più di uno. Si ipotizza inoltre una modellizzazione dei cluster di tipo probabilistico, e che vi siano delle *relazione fisse* tra i diversi cluster.

K-medie Algoritmo più semplice di clustering, si basa sulla presenza di un **centroide** di un insieme di informazioni. Questo è identificato come la media degli attributi delle osservazioni assegnate a un determinato cluster.

Nell'algoritmo, si inizia a considerare k elementi a caso tra i dati come centroidi iniziali. In realtà esistono molti meccanismi per sceglierli. Una volta scelti, si formano dei k cluster e si assegna ogni record al centroide che è più prossimo. Si ricalcola quindi il centroide per ogni cluster e si ripete. Si continua fino a quando i centroidi non rimangano immutati.

La scelta di k deve essere fatto a mano.

7.6 Algoritmi di Clustering - Parte 2

K-medie Come già detto, questo algoritmo non dipende dalla scelta del tipo di distanza, e si possono quindi usare delle differenti misure, come:

- Manhattan
- Euclidea quadrata
- Coseno

Questo tipo di algoritmo ha però una serie di problemi, che pongono delle difficoltà nella realizzazione:

- Scelta dei centroidi. Purtroppo, la scelta dei centroidi può modificare il risultato finale, cosa poco desiderata. Per ovviare questo problema, si va ad applicare differenti paradigmi per i centroidi. Oppure si può fissare un algoritmo di clustering gerarchico per la scelta dei centroidi.
- Complessità spaziale e temporale. L'algoritmo occupa una quantità modesta di spazio ed è lineare nel tempo, e quindi i tempi risultano modesti.

- Cluster vuoti. In alcuni casi, i centroidi assegnati potrebbero non avere istanze assegnate.
- Outliers. Quando si sintetizza una serie di dati con delle medie, che ovviamente è molto sensibile alla presenza di outlier nei dati. Si noti che però il metodo può essere usato per calcolare la presenza di outliers.
- Difficoltà di identificare dei cluster con forma non-sferica. Infatti, il K-media tende a raggruppare i dati intorno a un centro.
- Difficoltà nel trovare dei cluster che abbiano dimensione differente. In questo caso, la tendenza a fare dei cluster di ugual dimensione, può rimandare risultati erronei.

Quello che si può fare in molte istanze, spesso conviene dividere i cluster in sottocluster e poi riportarsi a una quantità maggiore. In questo modo, l'algoritmo riesce a essere più preciso. Questo approccio può aiutare a trattare diversi casi "problematici".

Forza e debolezza del K-media Questo algoritmo è molto semplice e si applica a diversi tipi di dati. Come detto, scala linearmente nel tempo, ovvero risulta molto efficiente; esistono comunque versioni che ottimizzano, e.g. *bisectin K-mean*. Nel qual caso non abbia senso parlare di centroide, perde di validità e soffre quando si ha grandezza o forma non ideale.

Esistono delle derivazioni del K-media che introducono il concetto del *medioide*, che sostituiscono il concetto di *centroide*. Questa non è una media ma un'osservazione nel dataset, e non è quindi influenzato dalla presenza di outliers.

7.7 Algoritmi di Clustering - Parte 3

Fuzzy C-medie Una generalizzazione del K-medie è noto come **fuzzy C-means**. Questo si basa sul fatto che le variabili possano essere associate a diversi tipi di cluster. Tutte le osservazioni che si trovano sulla "frontiera" tra due cluster vengono assegnate a entrambe, piuttosto che continuare a oscillare tra entrambi i gruppi a cui è assegnato.

Conditions Si indica con w_{ij} il peso della i-esima osservazione al j-esimo cluster. La formulazione matematica introduce il concetto di **pseudo-partizione fuzzy**, fatte in modo tale che ogni osservazione appartenga a tutti i cluster con un grado diverso. In particolare"

$$w_{ij} \geq 0, \ \forall i = 1, \ldots, m \ \wedge \ j = 1, \ldots, K$$

Non tutte le scelte dei valori è valida, e devono valere i due seguenti vincoli:

- Ogni oggetto è dato con un grado di appartenenza

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

- Non esistono dei cluster vuoti:

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

Ovvero tutti gli oggetti non possono esclusivamente assegnati a un solo cluster.

Questi vincoli fanno sì che non esistano cluster "inutili" e che le osservazioni siano tutte di un cluster.

7.8 Algoritmi di Clustering - Parte 4

Modello a mistura La logica di questo modello è quella di vedere i dati come proveniente da un set di differenti distribuzioni di probabilità, tendenzialmente delle Gaussiane multi-variate. Il processo generativo è del tipo:

- Si seleziona a caso una delle componenti.
- Si sceglie un'osservazione dalla distribuzione selezionata.
- Si ripete il processo fino a quando non si giunge a convergenza.

Matematicamente, si ha:

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

dove w_j è la probabilità di scegliere una mistura e $p(\underline{x}|\theta_j)$ rappresenta la probabilità di estrarre x dalla distribuzione j-esima. Si ottengono quindi delle osservazioni $\underline{x}$ messe secondo la mistura. Questa è comunque una concettualizzazione: in realtà non conosciamo quante componenti ci sono che hanno generato i dati e non si conoscono le matrici di *varianza-covarianza* dei cluster.

Nella realtà non sappiamo quanto sono le componenti, come sono fatte le loro curve di livello e come sono distribuite. Spesso si assume a priori che le curve siano delle *Gaussiane*, e si cerca di capire quante siano le componenti del problema. In ambito di analisi dei segnali è noto come **blind source separation**, ovvero quello di isolare l'attenzione da un rumore di sottofondo.

Expectation maximization Il modo per riuscire a trovare quanti modelli ci sono nel problema, si usa una tecnica nota come *Expectation maximization*. in essa, ci si alterna a "mondi alternativi" fino a quando non si trova una convergenza. In generale, l'algoritmo funziona bene e converge dopo poche iterazioni.

Problemi e vantaggi Il problema di questo modello è che è pesante computazionalmente, e risulta poco affidabile in presenza di tanti cluster. Inoltre non funziona molto bene quando i cluster hanno poche osservazioni. Ha inoltre dei problemi quando le osservazioni sono co-lineari.

Rispetto al K-medie sono molto più generali, e in grado di trovare cluster diversi e con forme a piacimento. Sono comunque un modo che si basa su un ipotesi chiara della generazione.

Mappe di Kohonen Note anche come *Self Organizing Maps*, sono state trovate da un neuroscienziato che studiava i movimenti degli arti dei gatti. Ogni singolo attributo è connesso con dei neuroni all'interno di una mappa, interconnessi tra loro.

Da uno spazio di input, continuo e a grandi dimensioni, si cerca di andare in uno spazio di output, spesso a dimensioni minori. Si utilizza una mappa per andare nello spazio di output, composto dai differenti neuroni. Questa tecnica porta ad avere un'organizzazione topologica dei dati, ovvero informazione sulla posizione assoluta e relativa.

Mentre gli altri algoritmi non hanno un'interpretazione topografica di come siano i cluster, e non hanno percezione se due cluster siano più o meno vicini. Nelle SOM, ogni osservazione viene assegnata a un neurone della mappa di output usando il concetto di distanza minima da un centroide.

Dato un neurone, si identificano con *vicinato* tutti i neuroni intorno, che dovrebbe risentire della presenza che una certa osservazione sia stata associata al dato neurone. Durante l'apprendimento, si ha aggiustamento dei parametri per tutti i neuroni del vicinato. Ovviamente può essere definito il vicinato in maniera differente.

Una volta scelti i neuroni, si esegue una *competizione*, a cui viene assegnata una particolare istanza. Successivamente, il neurone vinto aggiorna sia il suo centroide sia quello dei neuroni vicini, nota come *collaborazione*. In ultimo passo, si cambiano quindi i centroidi e si ricomincia.

Pro e Contro SOM Questo dispositivo ha tanti parametri e molto flessibile, motivo per cui è difficile da utilizzare ma può risultare molto potente.

7.9 Algoritmi di Clustering - Parte 5

Clustering gerarchico Si possono avere due diverse tipologie, **agglomerato**, in cui ogni oggetti è un cluster individuale e si procede a ritroso; oppure **divisivo**, in cui si ha un cluster che contiene tutto e si procede a rimpicciolirsi. Il metodo che si riesce ad applicare con una certa frequenza, sebbene pesante computazionalmente, è quella dell'agglomerazione. Il divisivo è troppo pesante computazionalmente e raramente usato.

Il **clustering gerarchico agglomerativo** viene spesso rappresentato attraverso un albero, che ricorda il procedimento che si esegue per produrre *decision trees*. La rappresentazione viene spesso chiamata *deindogramma*.

Per prima cosa, si calcola la *matrice di prossimità*; si entra quindi in `loop` e si eliminano i primi cluster che sono vicini tra di loro. Con i nuovi cluster si calcola quindi una nuova matrice e si ripete il processo, fino a quando non rimane un solo cluster.

Il problema che pone questo algoritmo è il calcolo della **prossimità tra due cluster**. Ci sono diversi metodi:

- *Min or Single linkage*, ovvero tra i due casi più vicini tra di loro.
- *Max or Complete linkage*, ovvero tra le due osservabili più distanti tra quelle possibili.
- *Group average or average linkage*, in cui si calcola il valor medio tra tutte le distanze possibili.

Gli algoritmi gerarchici sono in grado di gestire molto bene cluster che presentano dimensioni differenti. Inoltre, questi algoritmi non specificano delle funzioni oggettive globali e a ogni step si ha la possibilità di scegliere differenti metodi di unione. Uno dei limiti è però che, una volta eseguita un'unione non si può tornare indietro; ovvero si soffre delle scelte fatte precedentemente.

Tra le debolezze, si ha una complessità computazionale e il dendogramma che si ottiene può essere equivoco, ovvero difficile da interpretare. Le scelte che si vanno a fare sono di tipo subottimali, ovvero non si ha una maniera precisa per decidere come unire i diversi cluster.

7.10 Algoritmi di Clustering Parte 6

Clustering basato su densità Gli algoritmi di tipo **DBSCAN** sono caratterizzata dall'avere alta densità all'interno e da essere separati da regioni di bassa densità tra di loro. Ci sono diversi modo per definire il concetto di densità, che quindi va a differenziare.

DBSCAN Se si tiene come concetto per fare la densità quello di centro, si ha appunto algoritmi **DBSCAN**. Ogni dentro circonferenze concentriche poste su un centroide, si va a calcolare la densità come quantità di osservabili all'interno della circonferenza. Si deve però definire il concetto di *raggio* relativo alla misurazione.

I **core point** sono quei punti che stanno all'interno della regione ad elevata densità. Un punto è definito come *core point* solamente se la quantità di punto intorno presenta una certa densità, maggiore di una soglia posta a priori. Anche la distanza su cui si definisce la grandezza della circonferenza centrata sui diversi punti deve essere specificata a priori.

Tutte le osservazioni che non sono definite come *core point* sono dette o **border point**, ovvero molto vicini a quelli *core*, oppure **noise point**, tutti gli altri.

L'algoritmo si basa sui seguenti punti:

1. Si classificano tutti i punti.
2. Si eliminano i *noise point*.
3. Si collegano tra loro tutti i core points.
4. Si assegna ogni border point a uno e un solo border point.

I problemi principali di questo algoritmo sono dovuti al fatto che si devono specificare a mano due parametri. L'addestramento cerca di ottimizzare questi ultimi.

Rispetto al *K-medie*, il *DBSCAN* va a buttare via alcune osservazione, ovvero non è sensibile agli outliers. Inoltre esso è in grado di gestire dei cluster che hanno delle forme differenti, ma entrambi funzionano male se la densità varia sensibilmente nelle varie zone - si ricordi che il parametro del raggio viene definito per tutti. Il *K-medie* può essere applicato a dati sparsi o alti dimensionali, mentre *DBSCAN* funziona abbastanza male, in quanto basato su un concetto di distanza euclidea. Inoltre, con il *K-medie* si devono fare delle ipotesi sulle distribuzioni, cosa che non si fa con quello gerarchico. Entrambi gli algoritmi partono dal presupposto che si sappia come sono fatti i cluster e su quali attributi.

Altri algoritmi In letteratura, ci sono anche algoritmi del tipo **grid-based** oppure algoritmi come **CLOSA**, altamente specializzati, ovvero che fa clusterizzazione sia dalle osservazioni che dagli attributi. Ci sono poi una serie di algoritmi basati su *kernel*, il più importante è quello noto come **DENCLUE**.

Esistono comunque diverse tecniche in letteratura e in continua innovazione, soprattutto specializzati a particolari problemi.

7.11 Algoritmi di Clustering - Parte 7

Algoritmi a grafo Questa classe di algoritmi è molto limitata e mostriamo solo l'idea alla base. Già quelli gerarchici offrono una rappresentazione a grafo dei dati, in quanto mettono insieme a vari livelli le osservazioni e i cluster più vicini tra loro. Uno basato su grafo va a rappresentare il dataset come grafo, in cui i nodi sono le osservazioni; gli archi rappresentano la similarità.

Operazioni Tra le operazioni che si possono fare, vi è la **sparsificazione della prossimità del grafo**, ovvero si "taglia" il grafo in punti in cui le osservazioni sono troppo lontani. Si rimuovono tutti quegli archi che non superano una certa condizione. Questo può essere fatto o sulla distanza o sulla grandezza del vicinato.

È quindi importante definire il concetto di **similarità del vicinato**.

Si possono poi definire dei **nodi centrali**, intorno a cui si vanno a creare i cluster.

L'**uso di informazioni** nella prossimità può essere usato per fare una valutazione più sofisticata nel raggruppamento.

Gli algoritmi sui grafi hanno fatto grandi passi avanti, e ci sono algoritmi molto efficienti per trattarli. Questi sono particolarmente potenti e aiutano a capire la natura del problema. Per esempio, ci sono algoritmi che si basano sul *taglio di costo minimo*, ovvero quello che individua il sottoinsieme di archi che separa due componenti del grafo facendo in modo che il peggioramento sia meno possibile.

7.12 Algoritmi di Clustering - Parte 8

Mostriamo ora alcuni dei più semplici algoritmi su grafi e principi importanti.

Sparsificazione Una volta eseguita la sparsificazione del grafo, non sempre si possono avere delle componenti non connesse. Come idea, si potrebbe provare a ciclare sulla soglia per la sparsificazione, fino ad arrivare alla condizione in cui si hanno i grafi partizionati. Ci si accorgerà che i grafi sono ben definiti secondo il **concetto di click**.

La sparsificazione è molto potente, ma non garantisce che si possa essere efficace su tutto lo spazio.

Minimum Spanning Tree L'albero di supporto a costo minimo è molto semplice da calcolare. Esso deve essere caratterizzato dal non avere cicli e deve contenere tutti nodi del grafo originale. Inoltre, si definisce il costo dell'albero come la somma dei pesi associati ai suoi archi; e quindi si cerca il minimo di questi.

Si noti come ci sono casi in cui si potrebbe avere più alberi con uguale costo.

Questo algoritmo implica il fatto che si debba definire una distanza, o un qualche peso associato agli archi.

OPOSSUM Questo algoritmo partiziona il grafo in k componenti e usa l'algoritmo **METIS**, un tipo specializzato nella realizzazione di algoritmi su grafi. L'algoritmo è veloce e semplice, con molte implementazioni, con la creazione di cluster che hanno circa la stessa numerosità.

CHAMALEON In questo algoritmo, prima di sparsifica fino a ottenere delle componenti separate e si vanno poi ad aggregare le componenti.

7.13 Validità dei cluster - Parte 1

Diversi cluster per uno stesso problema Data la complessità di un cluster, il risultato che si fornisce dipende dalla scelta di algoritmo, attributi. tipologia (esclusivo oppure con sovrapposizioni) e completezza. Una volta esclusi alcuni algoritmi e alcune misure di similarità/distanza, si deve applicare un criterio per confrontare quello che si ha a disposizione.

Valutazione La fase di valutazione tra differenti risultati è molto importante per capire quale sia l'algoritmo migliore per un dato problema. Come prima cosa, si dovrebbe verificare che effettivamente vi sia una tendenza a formare dei cluster da parte dei dati. In seconda misura, si deve controllare quale attributi possano determinare la clusterizzazione. La **natura della domanda** è al centro del problema.

Una volta stabilità la tendenza dei dati di raggrupparsi, ci si deve porre come domanda quale sia il **numero di cluster** che si dovrebbero formare.

Indici di valutazione Per rispondere alle domande precedenti, si può usare una serie di indici, che possono essere raggruppati come:

- Indici esterni o supervisionati, ovvero vanno a vedere come funziona l'algoritmo sapendo a priori quale sia la verità.
- Indici interni o non-supervisionati, tendenzialmente misure di coesione o misure di separazione.
- Indici relativi, che servono per stimare il numero di cluster.

Misure esterne In questo caso, si hanno a disposizione già a priori le categorie dentro cui devono essere raggruppati i dati; ovvero si conosce a priori il risultato desiderato. Data una partizione di m oggetti dentro R categorie, $P = \{P_1, \dots, P_R\}$, e una possibile raggruppamento del tipo

$$C = \{C_1, \dots, C_K\}$$

Ovviamente, si deve capire quanto C e P possano essere simili tra di loro. Dati due dati x e y, può capitare che essi la loro classificazione coincida o meno con quella della partizione.

Il numero totale di coppie che possono essere formate è $M = \frac{m(m-1)}{2}$. Si definisce l'**indice di Rand** come:

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

Altre misure sono l'**indice di Jaccard**:

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

Altri indici sono l'**indice di Fowlkes-Mallows**. `Vedere come sono definiti` a, b, c, d.

7.14 Validità dei cluster - Parte 2

Misure non-supervisionate Se si è chiamati a valutare la classificazione è sensato dire che la validità complessiva sia una media delle validità di ogni coppia, pesata su un qualche riferimento. Ovvero del tipo

$$\texttt{overall validity} = \sum_{i=1} w_i \cdot \texttt{validity}_i$$

Quello che si vuole è avere **coesione grande** e **separazione bassa**.

Graph-based cluster Il concetto di coesione dipende dal tipo di classificazione. In particolare, nel caso di un grafo essa è definita come la somma sulla prossimità tra le coppie di un cluster:

$$\texttt{cohesion}_i = \sum_{x,y \in C_i} \texttt{proximity}(x, y)$$

Per tanto, sia la coesione che la similarità vengono massimizzate quando la dissimilarità è minima.

La **separazione** tra una coppia di cluster a grafo è definita sulla prossimità, ma tra due elementi in cluster diversi:

$$\texttt{separation}_{i,j} = \sum_{x \in C_i \neq C_j \ni y} \texttt{proximity}(x, y)$$

Prototype cluster In questo caso, si deve vedere la relazione tra gli elementi di un cluster e il loro prototipo

$$\texttt{cohesion}_i = \sum_{x \in C_i} \texttt{proximity}(x, c_i)$$

dove c_i indica il centroide/medoide a del cluster i-esimo. Analogamente si avrà:

$$\texttt{separation}_{i,j} = \texttt{proximity}(c_i, c_j)$$

Questo calcolo è quello computazionalmente più efficace.

Si può però calcolare la separazione anche in base a quanto ogni rappresentante dista dal centroide dato da tutti i dati insieme. Ovvero

$$\texttt{separation}_i = \texttt{proximity}(c_i, c)$$

dove c identifica il centroide/medoide di tutti i dati, o *overall prototype*.

Pesi coesione Il tipo di peso che si deve usare per unire le varie coesioni dipende dal tipo di algoritmo usato. Nel caso di approccio a grafo, il peso è calcolato come $1/m_i$, mentre per quelli a prototipo è sempre 1.

Pesi separazione Nel caso della separazione, si usa m_i come peso per quelli a grafo e sempre 1 per quelli a prototipo.

Migliorare la qualità del cluster Nel caso di coesione e separazione, si possono post-processare i cluster che si sono ottenuti, e.g. mi accorgo che unendo due cluster ho dei risultati migliori. È molto importante andare a classificare il contributo che ogni cluster contribuisce alla validità complessiva del processo.

Coefficiente di silhouette In particolare, le osservazioni sul "bordo" di un cluster tendono a essere più disperse; in base a questo, si può definire il **coefficiente di silhouette**, misura riassuntiva a diversi livelli che dice quanto un'osservazione sia sensata in un cluster, quanto sia sensato il cluster e, in analisi massima, quanto sia buona la classificazione. Questo viene calcolato come:

$$s_i = \frac{b_i - a_i}{\max(a_i, b_i)}$$

Dove a_i è il valore medio della distanza dell'osservazione i-esima e tutti gli oggetti all'interno del suo cluster; e b_i misura il minimo delle distanze tra l'osservazione e gli altri cluster. Ciascuna di queste distanze *osservazione-cluster* viene fatta come media tra l'osservazione i-esima e ogni osservazione dentro il dato cluster.

Definita così, è un valore che oscilla tra 1 e -1; in particolare, è 1 quando l'osservazione è associata in maniera perfetta al suo cluster e -1 quando invece è completamente sbagliata.

Si può poi calcolare il **valor medio del coefficiente di silhouette** semplicemente prendendo il valor medio di tutti in un cluster. In maniera analoga si può eseguire anche su tutto il dataset.

Coefficiente Cophonetic Correlation Coefficient Si valuta quanto sia in accordo la matrice delle prossimità con la **matrice cofonetica**, ovvero quella matrice che tiene traccia della distanza tra due osservazione quando vengono messe in uno stesso cluster per la prima volta. Dalla sua definizione, questa misura è molto utile nel caso di *cluster gerarchico*.

Si noti che, nel caso di **cluster average linkage** un valore elevato del coefficiente non porta ad avere una buona validità.

7.15 Validità dei cluster - Parte 3

Ipotesi nulle Per vedere la validità di una classificazione, si può usare un **paradigma di validità**. In prima cosa si eseguono una serie di **ipotesi nulle**, tra cui:

- Random position, applicato alle osservazioni continue.
- Random graph, applicato per dati ordinali.

- Random label, applicato per qualsiasi altro tipo di dato.

In generale, tutte e 3 le ipotesi si basano sul fatto che ci sia qualcosa di distribuito in maniera casuale. Si ricava quindi una **distribuzione di probabilità** e si estraggono dei valori casuali da essa, e.g. tramite **analisi Monte-Carlo** e *bootstrapping*. Si ottiene quindi come sono distribuiti i valori della misura nel caso in cui ci sia casualità. Ovvero si confrontano le misure estratte casualmente con i risultati che si sono ottenuti dai dati.

7.16 Validità dei cluster - Parte 4

Indici relativi Si possono usare alcuni indici per determinare il numero ottimale dei cluster all'interno del dataset. Quello che si potrebbe fare sarebbe proiettare i dati su uno spazio a bassa dimensione, e questo approccio rimane ristretto all'assunzione che lo spazio di lavoro sia sufficiente.

Viceversa si può usare il calcolo di alcuni indici:

- **Indice di Calinski-Harabasz**
- **Indice di Dunn**
- **Indice di Davies-Bouldin**

Nei primi due si cerca il numero K di cluster per cui questo è massimo, mentre per l'ultimo si cerca di minimizzare l'indice.

Indice per Clustering probabilistico

- **Criterio informazionale di Akaike**
- **MDL**
- **Criterio informazionale Bayesiano**

Quello che si esegue è applica l'algoritmo diverse volte per più valori di clustering, ovvero in un range $[K_{min}, K_{max}]$, e calcolare uno dei vari indici. Andare a risolvere questa procedura porta a ottenere una curva, non serve ottimale.

8 Analisi Associativa

8.1 Introduzione - Parte 1

Se si osserva che una determinata variabile assume un certo valore, che cosa si può dire di altre variabili; questo ovviamente senza poter interagire con il sistema. Le **regole di assocazioni** sono molto importanti, in quanto permettono di risolvere importanti domande operative.

L'obiettivo è quello di identificare quali osservazioni siano *associate*. Quello che si fornisce sono un insieme di **regole associative**, dette anche ***frequent item set***. Si creano delle tabelle con associazioni, identificati da binari.

8.2 Introduzione - Parte 2

Si ipotizzi di avere $I = \{i_1, \ldots, i_d\}$ oggetti e $T = \{t_1, \ldots, t_N\}$ numero di transazioni in un certo intervallo temporale. Una collezione di items è identificato come *item set*, uno dei possibili insiemi che si può ottenere unendo alcuni items. Se l'insieme contiene k oggetti si chiama *k-itemset*. Una transazione contiene un itemset se esso è un sottoinsieme della transazione medesima.

Si chiama **support count** il numero di volte che un *itemset* si presenta all'interno di tutte le transazioni. *vedi definizione matematica su slide.*

Regola associativa Essa è un'implicazione del tipo:

$$X \to Y$$

dove X e Y sono due *itemset* disgiunti, ovvero $X \cap Y = \emptyset$.

Si può misurare la forza della regola di associazione, tramite due parametri. Il **supporto** indica quanto spesso una regola è applicabile a un set:

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

Questo da un indice di quanto la regola è presente all'interno del dataset.

La **confidenza** è:

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

Essa misura l'affermazione di chi è la precondizione e chi è la conseguenza. Indica quanto l'elemento Y appare insieme a X in una transazione.

La confidenza sarà maggiore o uguale del supporto, anche se di norma si useranno due soglie differenti per determinare che cosa interessa e, tra queste, quali siano più efficaci.

Una regola con basso supporto può essere legata a pura casualità oppure che è qualcosa di non interessante. Non bisogna comunque interpretare le associazione come delle causalità.

Il problema della Association Rule Mining Dato un insieme di transazioni T si cercano tutte le regole che hanno `support > minsup` e `confidence > minconf`.

Un approccio a forza bruta è del tipo:

$$R = 3^d - 2^{d+1} + 1$$

ovvero computazionalmente molto difficile da calcolare.

Per semplicità, spesso si separa il problema in due parti: prima si cerca di risolvere il primo threshold, per ottenere i *frequent itemsets*, e poi si cerca di trovare quali tra questi sono molto confidenti, detti *strong itemset*.

8.3 Frequent itemset generation and rule generation

Si cerca di risolvere la prima componente delle regole associative, quella in cui si generano tutti e solo gli itemset valutati *frequenti*. Si identificano dei **candidate itemset**, ovvero quegli itemset che potrebbero essere frequenti. Con questa metodologia, si richiedono solamente $o(Mw)$ paragoni, dove M è la grandezza del *candidate itemset*.

Si possono poi ridurre il numero di confronti, usando delle strutture dati invertire, ovvero si scandisce solamente la transaction list. Si usa poi il **principio a priori**, ovvero se un itemset è frequente, lo saranno anche tutti i suoi sottoinsiemi.

Apriori algorithm Questo è stato il primo algoritmo sviluppato per itemset frequenti e rimane comunque subottimale. Ha inoltre trovato un'implementazione molto diffusa. L'algoritmo usa una *generazione di candidati*, ovvero genera k itemset candidati basati su $k-1$ itemset frequenti della generazione precedente. Ovviamente, più si va avanti più sono i passaggi che si devono fare.

Rule generation Tutte le regole che sono generate da un itemset frequente sono esse stesse frequenti. Di fronte a un grande numero di itemset frequenti può essere molto grande. Può essere quindi utile trovare un sovrainsieme rappresentativo di un certo gruppo di itemset, legati tra loro. Si focalizza l'attenzione su degli itemset che godono di particolari proprietà: si parla di **maximal frequent itemset** e **closed frequente itemset**.

Il primo è l'itemset più frequente con la proprietà che nessuno dei suoi superset è frequente. Verifica il vincolo ed è la frontiera oltre la quale non si è più frequenti. Se si estende in qualsiasi direzione, non si gode più della proprietà. Nell'altra direzione, ogni suo subset è frequente per definizione. Esso però è pratico solamente se l'algoritmo è in grado di cercare solo quello, senza andare a generare di nuovo tutto l'albero.

Un **itemset chiuso** è uno in cui tutte le estensione che partono da esso hanno lo stesso *support count*. Essi sono utili per evitare che si applichino delle regole ridondanti. Sarebbe ovviamente meglio non andare a generare questo tipo di regole.

Gli itemset chiusi contengono tutti gli itemset massimali.

9 Seminario Deep Learning

Non sono andato a lezione: vedi slide nella sezione *expert* dell'ELearning.