Algoritmi Avanzati
A.A. 2017–2018, primo semestre
Traccia delle lezioni

Mauro Brunato

Versione 2017-12-12

Caveat lector

Lo scopo principale di questi appunti è quello di ricostruire quanto detto a lezione. Queste note non sono complete, e la loro lettura non permette, da sola, di superare l'esame. Le fonti utili a ricostruire un discorso coerente e completo sono riportate alla pagina web del corso, dov'è disponibile anche la versione più recente di queste note:

`https://disi.unitn.it/~brunato/AA/`

Alcune fonti per approfondimenti sono indicate nelle note a pie' di pagina di questo documento.

Si suggerisce di confrontare la data riportata sul sito web con quella che appare nel frontespizio per verificare la presenza di aggiornamenti.

Changelog

2017-12-12

- Selezione degli attributi
- Clustering
- Domande ed esercizi
- Ultime esercitazioni di laboratorio

2017-11-20

- Alberi di decisione
- Domande ed esercizi sugli alberi di decisione
- Esercitazioni di laboratorio sugli alberi di decisione

2017-10-27

- Correzione di alcuni refusi nella teoria e negli esercizi
- Utilizzo di KNN per problemi di regressione
- Aggiornamento dell'esercitazione di laboratorio sulla regressione logistica

2017-10-18

- Regressione logistica
- Normalizzazione e standardizzazione
- Esercitazione di laboratorio sulla regressione logistica

2017-10-12

- Algoritmi di regressione per problemi di classificazione: impostazione di soglie
- Introduzione alla regressione logistica
- Cross validation
- Esercizi

2017-10-05

- Generalizzazione dei minimi quadrati a funzioni non lineari e al caso polinomiale
- Iniziato il capitolo sulla validazione
- Terza esercitazione

2017-10-01

- Correzione di alcuni errori nelle formule dei minimi quadrati
- Seconda esercitazione

2017-09-25

- Regressione: RMSE e metodo dei minimi quadrati
- Domande di comprensione

2017-09-24

- Formalizzazione del problema del machine learning
- Prima esercitazione

2017-09-16

Versione iniziale:

- Introduzione, scopo del corso

Indice

I Appunti di teoria **1**

1 Introduzione alla Data Science **2**
1.1 La Data Science . . . 2
1.2 Scopo del corso . . . 3
1.2.1 Significatività di un esperimento: il p-value . . . 3
1.2.2 Correlazioni spurie . . . 3
1.2.3 Il paradosso di Simpson . . . 4
1.3 Formalizzazione del problema dell'apprendimento automatico . . . 4
1.3.1 L'addestramento e il modello . . . 4
1.4 Considerazioni sulla dipendenza funzionale . . . 5

2 Algoritmi di regressione **6**
2.1 Valutazione della bontà di un modello di regressione . . . 6
2.2 Modelli lineari: il metodo dei minimi quadrati . . . 6
2.2.1 Il caso scalare ($n = 1$) . . . 7
2.2.2 Il caso vettoriale ($n > 1$) . . . 7
2.2.3 Generalizzazione per funzioni non lineari . . . 8
2.2.4 Problemi di classificazione . . . 9
2.3 Regressione logistica . . . 9
2.3.1 Stima dei coefficienti del modello . . . 10
2.3.2 Utilizzo del modello . . . 11

3 Algoritmi di classificazione **12**
3.1 K-Nearest-Neighbors (KNN) . . . 12
3.1.1 Applicazione di KNN a problemi di regressione . . . 12

4 Valutare la bontà di un modello **14**
4.1 Criteri di valutazione . . . 14
4.1.1 Valutare un modello di regressione . . . 14
4.1.2 Valutare un modello di classificazione . . . 14
4.2 Addestramento e validazione . . . 16
4.2.1 K-fold cross-validation . . . 16
4.2.2 Una soluzione estrema: l'approccio *leave-one-out* . . . 17

5 Pretrattare i dati **18**
5.1 K-nearest-neighbors . . . 18
5.1.1 Input categorici . . . 18
5.2 Normalizzazione e standardizzazione dei dati . . . 18
5.2.1 Normalizzazione . . . 19
5.2.2 Standardizzazione . . . 19
5.2.3 Insiemi di validazione . . . 19

6 Alberi di decisione **20**
6.1 Addestramento di un albero di ricerca . . . 20
6.1.1 Prima misura di imprevedibilità: l'entropia di Shannon . . . 20
6.1.2 Seconda misura di imprevedibilità: l'impurità di Gini . . . 21
6.2 Passo induttivo: riduzione dell'imprevedibilità . . . 21
6.3 Partizione ricorsiva del dataset e condizione di terminazione . . . 22
6.3.1 Variabili in input categoriche: l'algoritmo ID3 . . . 23

7 Selezione degli attributi **24**
7.1 Dati continui: il coefficiente di correlazione . . . 24
7.2 Variabili categoriche: l'informazione mutua . . . 25
7.2.1 Stima dell'informazione mutua in presenza di variabili continue . . . 26
7.2.2 Qualche tranello . . . 27
7.3 Un algoritmo greedy per la selezione degli attributi . . . 27

8 Cenni di apprendimento non supervisionato **28**
8.1 Distanze e somiglianze . . . 28
8.1.1 Dati non numerici . . . 29
8.2 Classificazione degli algoritmi di clustering . . . 29
8.3 Una tecnica bottom-up: clustering agglomerativo gerarchico . . . 29
8.3.1 Distanze fra insiemi . . . 30
8.3.2 L'algoritmo . . . 30
8.3.3 Similitudini e distanze . . . 30
8.3.4 Rappresentazione interna ed esterna . . . 32
8.4 Una tecnica top-down: il clustering K-means . . . 32
8.4.1 L'algoritmo "hard" K-means . . . 33

II Esercitazioni di laboratorio **34**

1 Lettura e utilizzo di un semplice dataset **35**
1.1 Scopo dell'esercitazione . . . 35
1.2 Script . . . 35
1.3 Osservazioni . . . 35

2 K-Nearest Neighbors e minimi quadrati a una dimensione **37**
2.1 Scopo dell'esercitazione . . . 37
2.2 Script . . . 37
2.3 Osservazioni . . . 37

3 Regressione polinomiale e suddivisione del dataset **39**
3.1 Scopo dell'esercitazione . . . 39
3.2 Script . . . 39
3.3 Osservazioni . . . 39

4 Regressione logistica **41**
4.1 Scopo . . . 41
4.2 Procedimento . . . 41
4.2.1 Suddivisione del dataset . . . 41
4.2.2 Normalizzazione . . . 41
4.2.3 Discesa lungo il gradiente . . . 41
4.2.4 Addestramento . . . 42
4.3 Script . . . 42

5 Alberi di decisione **43**
5.1 Scopo . . . 43
5.2 Prima parte: utilizzo di una libreria esterna . . . 43
5.2.1 Codice . . . 43
5.2.2 Risultati . . . 43
5.3 Seconda parte: implementazione Python . . . 45
5.3.1 Codice . . . 45
5.3.2 Risultati . . . 45

6 Curve di prestazione **46**
6.1 Scopo . . . 46
6.2 Procedimento . . . 46
6.3 Script . . . 46
6.4 Discussione . . . 47

7 Clustering **49**
7.1 Scopo . . . 49
7.2 Il codice . . . 49
7.3 Risultati . . . 49

III Domande ed esercizi **53**

A Domande teoriche e spunti di riflessione **54**
A.1 Domande a risposta aperta . . . 54
A.1.1 Introduzione alla Data Science . . . 54
A.1.2 Algoritmi di machine learning . . . 54
A.1.3 Metodologia . . . 55
A.1.4 Selezione degli attributi . . . 55
A.2 Domande a risposta multipla . . . 56

B Esercizi **59**

Parte I

Appunti di teoria

Capitolo 1

Introduzione alla Data Science

1.1 La Data Science

Immagazzinare dati è fin troppo facile, e visti i costi ormai irrisori lo fanno tutti. Il problema è farne uso: i dati grezzi non sono immediatamente comprensibili, bisogna estrarre informazioni (fruibili da persone o da algoritmi) a fini migliorativi.

Questo è lo scopo della Data Science[1]; con sfumature diverse, si parla anche di "Business analytics", "Business Intelligence". Anche le discipline della Statistica e della Ricerca Operativa occupano lo stesso ambito, ma i loro nomi sono spesso associati a tecniche classiche e non sempre la loro definizione include il machine learning, che costituisce invece il nucleo del nostro corso.

In Figura 1.1 troviamo alcune "parole chiave" che definiscono le varie discipline e operazioni che portano dalla raccolta dati iniziale al risultato finale.

Il modus operandi più consueto consiste nell'utilizzare i dati per creare un modello (statistico, matematico, informatico) degli stessi da utilizzare poi per acquisire conoscenza sul sistema in esame, prendere decisioni, modificare processi, eccetera. Spesso, il risultato finale di tale processo sono nuovi dati che possono essere utilizzati per affinare il modello, e così via. Vedremo vari esempi nel resto del corso, che si concentrerà principalmente sulla fase dell'analisi dei dati e del machine learning.

[1] `https://en.wikipedia.org/wiki/Data_science`.

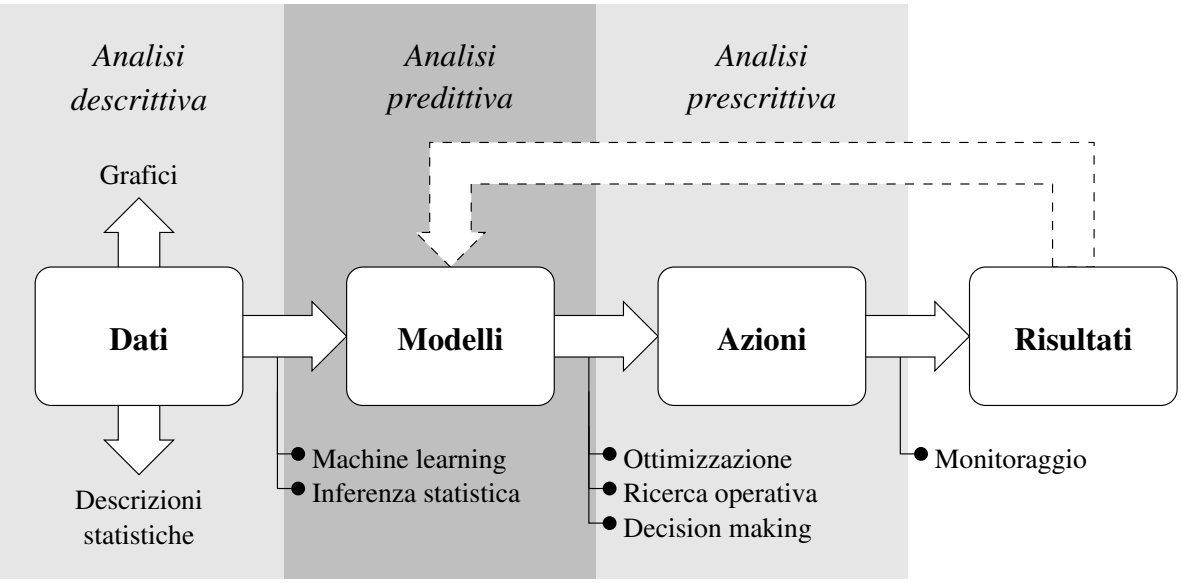


Figura 1.1: Alcune parole chiave

1.2 Scopo del corso

Esistono decine, se non centinaia, di pacchetti software e di librerie in grado di importare dati, costruire modelli, effettuare previsioni. Uno sviluppatore può creare una rete neurale, addestrarla e utilizzarla in un suo programma senza nemmeno sapere come funziona, semplicemente utilizzando le API di una libreria e trattando i diversi algoritmi come "scatole nere". Insomma, non è necessario conoscere un algoritmo di machine learning per poterlo utilizzare, come non è necessario conoscere i protocolli di rete in dettaglio per realizzare un'applicazione web, e si possono ordinare gli elementi di un vettore senza conoscere gli algoritmi di ordinamento.

Nonostante ciò, le tecniche che studieremo in questo corso vanno applicate con estrema attenzione, perché sono soggette a errori estremamente perniciosi, ma non sempre facili da evitare. Il corso sarà in parte dedicato a studiare le metodologie utili a evitare o a minimizzare l'impatto dei trabocchetti più comuni.

Ecco alcuni esempi di trabocchetto tratti dalla statistica classica. Più tardi ne vedremo anche alcuni specifici del machine learning.

1.2.1 Significatività di un esperimento: il p-value

Sappiamo che la Statistica offre strumenti per capire se un certo risultato è significativo o meno. Il principio generale è: più campioni misuriamo, minori sono gli errori che commettiamo nello stimare le grandezze statistiche della popolazione. Un metodo molto usato in ambito sperimentale è il cosiddetto "p-value". Formulata un'ipotesi H, ed effettuato un esperimento consistente nella misurazione di alcuni campioni di una popolazione, diciamo che l'esperimento conferma l'ipotesi se il suo "p-value" è al di sotto di una soglia, detta *significatività*, da noi stabilita a priori. Il p-value esprime la probabilità di ottenere risultati simili a quelli effettivamente ottenuti nel caso in cui H sia falsa[2].

Supponiamo, ad esempio, di sospettare che una certa moneta sia truccata in modo da offrire sempre il lato "testa" in un lancio. La nostra ipotesi H è "Questa moneta è truccata". Il nostro esperimento consiste di $N = 10$ lanci, e otteniamo sempre testa. Il p-value dell'esperimento risponde alla domanda "quale sarebbe la probabilità di ottenere dieci teste se H fosse falsa, cioè se la moneta *non* fosse truccata?" La risposta è ovviamente $p = 2^{-10} < 1/1000$. Con un p-value così basso possiamo concludere che il nostro risultato non è dovuto al caso, e che quindi la nostra ipotesi è confermata.

Cosa succede, però, se ogni italiano esegue questo test su una moneta di propria scelta? Anche se nessuna delle monete fosse truccata, ben 60000 italiani (circa uno su mille) concluderebbero che la loro moneta lo è. Dovremmo togliere dalla circolazione quelle sessantamila monete?

Si consideri che nella maggior parte degli esperimenti scientifici, soprattutto quando il costo degli esperimenti è elevato (si pensi a medicina, genetica, psicologia), la soglia di significatività è molto maggiore di 1/1000 (spesso ci si accontenta di 1/20, cioè del 5%). Se un esperimento consiste nella verifica di 20 diverse ipotesi, e ci si accontenta di una significatività del 5%, è possibile che una delle ipotesi venga confermata erroneamente, spesso per ingenuità, talvolta intenzionalmente (in tal caso, si parla di *p-hacking*[3]).

1.2.2 Correlazioni spurie

Un effetto collaterale dell'abbondanza di dati è la facilità con la quale, in assenza di un'ipotesi precisa da verificare, sia sempre possibile trovare serie di dati in apparente dipendenza l'uno dall'altro nonostante la completa estraneità[4].

[2] `https://en.wikipedia.org/wiki/P-value`.

[3] Un esempio: la cioccolata accelera il dimagrimento (`https://io9.gizmodo.com/i-fooled-millions-into-thinking-chocolate-helps-weight-1707251800`). Più in breve: `https://xkcd.com/882/` — vedere `https://explainxkcd.com/882/` per chiarimenti.

[4] `http://www.tylervigen.com/spurious-correlations`

1.2.3 Il paradosso di Simpson

In alcuni casi, nonostante l'ipotesi da verificare sia precisa e ogni test statistico dica che i risultati sono significativi, può accadere che la conclusione sia errata semplicemente perché le nostre informazioni sono incomplete. Un esempio è il cosiddetto "paradosso di Simpson", nel quale l'assenza di una variabile esplicativa causa la formulazione di un'ipotesi opposta rispetto alla realtà dei fatti[5].

1.3 Formalizzazione del problema dell'apprendimento automatico

Nella prima parte del corso ci occuperemo dell'apprendimento supervisionato (*supervised learning*), in cui si chiede al sistema di prevedere un esito sulla base di alcuni dati in ingresso. Ad esempio:

- date alcune misure fisiche, determinare la specie di un organismo vivente;
- date alcune informazioni su un conto bancario, determinare se un cliente sarà disposto ad accettare un certo investimento.

Assumeremo sempre che le informazioni in ingresso siano esprimibili come una tupla numerica (vettori in $\mathbb{R}^n$), e che l'informazione da prevedere sia invece scalare (un singolo valore).

Formalmente, possiamo descrivere il problema come segue: abbiamo un *dataset* costituito da m "individui" o "campioni":

$$D = \{(\boldsymbol{x}_i, y_i) : i = 1 \ldots, m\}.$$

- le grandezze $\boldsymbol{x}_i$ sono gli "ingressi" del nostro algoritmo, solitamente rappresentati da tuple numeriche. Rappresentano le variabili indipendenti del sistema, quelle che possiamo misurare (o impostare) direttamente. Ad esempio delle dimensioni fisiche, informazioni di censo...
- le y_i sono le "uscite", ovvero i valori che desideriamo imparare a predire sulla base delle $\boldsymbol{x}_i$ appena citate. Possono essere numeriche (ad esempio, una probabilità, un'età) oppure categoriche (la varietà di un fiore, una risposta sì/no).

Sulla base degli esempi forniti in D, il sistema deve fornire una mappa

$$\begin{aligned} f : \mathbb{R}^n &\to C \\ \boldsymbol{x} &\mapsto y \end{aligned}$$

che fornisca una stima (previsione, predizione) del valore di uscita y sulla base del vettore in ingresso $\boldsymbol{x}$.

L'insieme $\mathbb{R}^n$, che rappresenta lo spazio in cui si trovano i dati di ingresso, è anche detto "spazio degli attributi" (o delle "feature"), dove per "attributo" si intende ciascuna delle n dimensioni che lo compongono.

L'insieme di appartenenza dell'uscita y (indicato con C qui sopra) può essere:

- variabile con continuità in un intervallo dei reali; in tal caso si parla di un problema di *regressione* (in inglese "regression" o, più comunemente, "fit");
- un insieme finito di valori, numerici o alfanumerici, detti *classi* o *categorie*; si parla in tal caso di un problema di *classificazione*.

1.3.1 L'addestramento e il modello

La funzione f, detta anche "modello", deve quindi rappresentare per quanto possibile una dipendenza funzionale dei valori $y_1, \ldots, y_m$ dai corrispondenti vettori di attributi $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m$; per essere utile, questa dipendenza deve poter essere *generalizzata*, ovvero applicabile anche a nuove istanze $\boldsymbol{x} \in \mathbb{R}^n$ non presenti fra gli m campioni del dataset D.

[5] Si veda `https://en.wikipedia.org/wiki/Simpson%27s_paradox`, in particolare i primi due esempi.

Non è detto che tale dipendenza funzionale sia sempre molto precisa. Ci si potrebbe aspettare che il modello, ad esempio, fornisca sempre la risposta corretta per i campioni del dataset (cioè che $f(\boldsymbol{x}_i) = y_i$ per ogni $i = 1, \ldots, m$). Questo non è sempre possibile, né desiderabile, per varie ragioni:

- non sempre il vettore delgli attributi $\boldsymbol{x}$ è perfettamente rappresentativo del campione; potrebbero essere stati tralasciati alcuni attributi importanti, quindi il sistema potrebbe essere incerto nell'attribuzione di un valore y;
- spesso i dati sono "rumorosi" (errori di misura o, peggio, di trascrizione), quindi alcuni esempi del dataset D potrebbero essere imprecisi o errati;
- vedremo che un sistema che "impara a memoria" le risposte da dare sui campioni di D non sarà sempre in grado di fornire risposte adeguate su un nuovo esempio; si pensi a uno studente che, invece di imparare i principi generali della materia, si limiti a studiare a memoria lo svolgimento degli esercizi.

1.4 Considerazioni sulla dipendenza funzionale

Nonostante la maggior parte delle tecniche di cui ci occuperemo consista nella determinazione di una dipendenza funzionale, cioè *univoca* e *deterministica*, non è detto che questa possa esistere.

Consideriamo ad esempio il problema di classificazione degli iris (vedere la prima esercitazione di laboratorio), in cui è chiesto di determinare la varietà di un fiore sulla base di alcune misure. È possibile che due fiori i e j di varietà diverse ($y_i \neq y_j$) abbiano le stesse misure fisiche ($\boldsymbol{x}_i = \boldsymbol{x}_j$), e che quindi una dipendenza funzionale non sia nemmeno possibile. Ci troviamo in presenza di un errore *intrinseco*, ineliminabile da qualunque tecnica di machine learning[6]. L'unico modo di eliminare questo genere d'errore è la ricerca di nuovi attributi la cui misura permetta una migliore discriminazione.

Se un dataset lega l'altezza x al peso y di una popolazione umana, è ovvio che due individui i e j possono avere la stessa altezza ($x_i = x_j$) e pesi diversi. In questo caso, la dipendenza sarà meglio rappresentata come

$$y = f(\boldsymbol{x}) + \varepsilon, \tag{1.1}$$

dove ε, che rappresenta l'errore intrinseco, è una variabile casuale con valor medio nullo $E[\varepsilon] = 0$.

[6]Esistono però modelli di dipendenza probabilistica che, a fronte di un input $\boldsymbol{x}$, forniscono una *distribuzione di probabilità* per il corrispondente valore di uscita.

Capitolo 2

Algoritmi di regressione

Consideriamo il dataset D per un problema di regressione con m campioni e n attributi:

$$D = \{(\boldsymbol{x}_i, y_i) : i = 1, \dots, m, \quad \boldsymbol{x}_i \in \mathbb{R}^n, \quad y_i \in \mathbb{R}\}.$$

In seguito utilizzeremo la notazione matriciale, quindi potremmo riassumere il dataset in una matrice $X = (x_{ij}) \in \mathbb{R}^{m \times n}$ le cui righe sono i singoli vettori $\boldsymbol{x}_i \in \mathbb{R}^n$, e in un vettore $\boldsymbol{y} = (y_1, \dots, y_m) \in \mathbb{R}^m$ contenente i valori della variabile dipendente.

2.1 Valutazione della bontà di un modello di regressione

SUpponiamo di aver determinato, sulla base del dataset D, un modello $\hat{f}$. Per ogni individuo $i = 1, \dots, m$ del nostro dataset possiamo calcolare l'errore ε_i commesso dal nostro modello rispetto al valore "vero" y_i:

$$\varepsilon_i = y_i - \hat{f}(\boldsymbol{x}_i).$$

Dato che siamo interessati a ridurre per quanto possibile la magnitudine media dell'errore, una misura molto utilizzata per valutare la bontà della funzione $\hat{f}$ è il cosiddetto "Root Mean Square Error" (RMSE):

$$\mathrm{RMSE}(\hat{f}) = \sqrt{\frac{1}{m}\sum_{i=1}^{m} \varepsilon_i^2} = \sqrt{\frac{1}{m}\sum_{i=1}^{m} \big(y_i - \hat{f}(\boldsymbol{x}_i)\big)^2}.$$

Il problema di cercare la funzione migliore per modellare il dataset non è in generale ben definito, a meno di non limitare la ricerca a una specifica famiglia di funzioni $\mathcal{F}$, solitamente parametrizzata da uno o più valori reali. Il problema di trovare un buon modello può essere descritto come

$$\hat{f} = \arg\min_{f \in \mathcal{F}} \mathrm{RMSE}(f). \tag{2.1}$$

Considerato che la radice quadrata è monotona e un fattore costante non influisce sulla minimizzazione, possiamo riscrivere (2.1) come

$$\hat{f} = \arg\min_{f \in \mathcal{F}} \sum_{i=1}^{m} \big(y_i - f(\boldsymbol{x}_i)\big)^2. \tag{2.2}$$

2.2 Modelli lineari: il metodo dei minimi quadrati

Un caso molto utile, e molto più generalizzabile di quanto appaia a prima vista, è quello delle funzioni lineari. Nella consueta notazione, cerchiamo una funzione della forma

$$f_{\boldsymbol{\beta}}(\boldsymbol{x}) = \boldsymbol{\beta} \cdot \boldsymbol{x} = \sum_{j=1}^{n} \beta_j x_j,$$

dove $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_n) \in \mathbb{R}^n$ è un vettore di coefficienti reali da determinare. In questo caso, l'equazione (2.2) diventa

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta} \in \mathbb{R}^n} \sum_{i=1}^{m} (y_i - \boldsymbol{\beta} \cdot \boldsymbol{x}_i)^2. \tag{2.3}$$

2.2.1 Il caso scalare ($n = 1$)

Se il dataset contiene un solo attributo x, allora il problema si riduce a determinare un singolo valore scalare β:

$$\hat{\beta} = \arg\min_{\beta \in \mathbb{R}} \sum_{i=1}^{m} (y_i - \beta x_i)^2. \tag{2.4}$$

Per trovare il valore di β che minimizza l'RMSE, dunque, ci basta derivare la funzione in (2.4) rispetto a β e uguagliare la derivata a zero:

$$\frac{\mathrm{d}}{\mathrm{d}\beta} \sum_{i=1}^{m} (\beta x_i - y_i)^2 = 0;$$

$$2 \sum_{i=1}^{m} x_i (\beta x_i - y_i) = 0;$$

$$\beta \sum_{i=1}^{m} x_i^2 - \sum_{i=1}^{m} x_i y_i = 0;$$

Quindi il valore ottimale $\hat{\beta}$ è

$$\hat{\beta} = \frac{\sum_{i=1}^{m} x_i y_i}{\sum_{i=1}^{m} x_i^2}. \tag{2.5}$$

Si osservi che, nella notazione matriciale introdotta a inizio capitolo, la matrice X ha una colonna e (2.5) può essere scritta in forma più compatta come

$$\hat{\beta} = \frac{X^T \boldsymbol{y}}{X^T X}. \tag{2.6}$$

2.2.2 Il caso vettoriale ($n > 1$)

Per risolvere il problema con più attributi in input, dobbiamo ritornare alla formulazione (2.3) e annullare simultaneamente tutte le derivate parziali rispetto ai coefficienti β_k, $k = 1, \ldots, n$:

$$\frac{\partial}{\partial \beta_k} \sum_{i=1}^{m} \left(y_i - \sum_{j=1}^{n} \beta_j x_{ij} \right)^2 = 0 \qquad \forall k = 1, \ldots, n;$$

$$\sum_{i=1}^{m} \frac{\partial}{\partial \beta_k} \left(y_i - \sum_{j=1}^{n} \beta_j x_{ij} \right)^2 = 0;$$

$$\sum_{i=1}^{m} 2 \left(y_i - \sum_{j=1}^{n} \beta_j x_{ij} \right) x_{ik} = 0;$$

$$\sum_{i=1}^{m} x_{ki}^T (\boldsymbol{y} - X \cdot \boldsymbol{\beta})_i = 0 \qquad \forall k = 1, \ldots, n.$$

In forma matriciale, le equazioni per $k = 1, \dots, n$ assumono la seguente forma:

$$X^T(\boldsymbol{y} - X\boldsymbol{\beta}) = 0,$$

$$X^T X\boldsymbol{\beta} = X^T\boldsymbol{y},$$

$$\boldsymbol{\beta} = (X^T X)^{-1} X^T \boldsymbol{y}. \tag{2.7}$$

Notare che se $n = 1$ la formula appena ricavata si riduce alla (2.6).

2.2.3 Generalizzazione per funzioni non lineari

La regressione ai minimi quadrati si può utilizzare per trovare i coefficianti di qualunque combinazione lineare di funzioni. Nel caso generale, possiamo cercare i coefficienti di un modello esprimibile nella seguente forma:

$$y \sim \beta_1\varphi_1(\boldsymbol{x}) + \beta_2\varphi_2(\boldsymbol{x}) + \cdots + \beta_\ell\varphi_\ell(\boldsymbol{x}), \tag{2.8}$$

dove $\varphi_1(\cdot), \dots, \varphi_\ell(\cdot)$ sono ℓ "funzioni di base" (sul vettore di attributi $\boldsymbol{x}$) che vogliamo combinare linearmente.

Per fare questo, dalla matrice X del dataset originario otteniamo la matrice trasformata X', con m righe e ℓ colonne, la cui i-esima riga contiene i valori delle ℓ funzioni di base calcolati sulla riga $\boldsymbol{x}_i$ del dataset originario:

$$\boldsymbol{x}'_i = \big(\varphi_1(\boldsymbol{x}_i), \dots, \varphi_\ell(\boldsymbol{x}_i)\big)$$

o, in forma matriciale,

$$X' = \begin{pmatrix} \varphi_1(\boldsymbol{x}_1) & \varphi_2(\boldsymbol{x}_1) & \cdots & \varphi_\ell(\boldsymbol{x}_1) \\ \varphi_1(\boldsymbol{x}_2) & \varphi_2(\boldsymbol{x}_2) & \cdots & \varphi_\ell(\boldsymbol{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \varphi_1(\boldsymbol{x}_m) & \varphi_2(\boldsymbol{x}_m) & \cdots & \varphi_\ell(\boldsymbol{x}_m) \end{pmatrix}$$

In questo modo, il modello (2.8) può essere riscritto come

$$y \sim \beta_1 x'_1 + \beta_2 x'_2 + \cdots + \beta_\ell x'_p$$

e il vettore $\boldsymbol{\beta}$ può essere calcolato applicando la formula (2.7) alla matrice X':

$$\boldsymbol{\beta} = (X'^T X')^{-1} X'^T \boldsymbol{y}.$$

Caso particolare: regressione affine

Ad esempio, supponiamo di voler calcolare i coefficienti β_0 e β_1 del seguente modello, con x e y scalari:

$$y \sim \beta_0 + \beta_1 x,$$

dove abbiamo introdotto un termine costante, trasformando la dipendenza da lineare ad affine. Possiamo riscrivere il modello come

$$y \sim \beta_0\varphi_0(x) + \beta_1\varphi_1(x),$$

dove $\varphi_0(x) = x^0 = 1$ (funzione costante), mentre $\varphi_1(x) = x$ (identità). Possiamo dunque applicare l'equazione dei minimi quadrati alla matrice X' con due colonne, la prima delle quali contiene la costante 1, mentre la seconda contiene i valori originari di x:

$$X' = \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_m \end{pmatrix}.$$

Regressione polinomiale

Più in generale, consideriamo di voler determinare i coefficienti del seguente modello polinomiale (con x scalare) di grado d:

$$y \sim \beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_d x^d = \sum_{i=0}^{d} \beta_i x^i;$$

in questo caso, le funzioni di base saranno le potenze di x:

$$\varphi_i(x) = x^i, \qquad i = 0, \ldots, d.$$

La matrice X' è la seguente:

$$X' = \begin{pmatrix} 1 & x_1 & x_1^2 & \ldots & x_1^d \\ 1 & x_2 & x_2^2 & \ldots & x_2^d \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_m & x_m^2 & \ldots & x_m^d \end{pmatrix}. \tag{2.9}$$

2.2.4 Problemi di classificazione

È possibile utilizzare la regressione polinomiale per problemi di classificazione. L'adattamento è abbastanza semplice se la categoria in uscita ha due valori. In tal caso, si possono associare due valori reali (ad esempio, ± 1) alle due classi e trattare ciò che risulta come problema di regressione. Ad esempio, se la categoria in output può assumere i valori "verde" e "rosso":

1	rosso
2	rosso
3	verde
4	rosso
5	verde
6	rosso
7	verde

$$\Rightarrow \qquad \boldsymbol{y} = \begin{pmatrix} +1 \\ +1 \\ -1 \\ +1 \\ -1 \\ +1 \\ -1 \end{pmatrix}$$

Determinato il modello $y \sim f(\boldsymbol{x})$, il suo risultato su un input $\boldsymbol{x}$ non sarà necessariamente in ± 1, ma una semplice regola di decisione ci permetterà di stabilire il valore della classe:

$$\tilde{y} = \begin{cases} \text{rosso} & \text{se } f(\boldsymbol{x}) \geq 0 \\ \text{verde} & \text{altrimenti.} \end{cases}$$

2.3 Regressione logistica

Introduciamo la funzione *sigmoide*:

$$\sigma(t) = \frac{1}{1 + e^{-t}}. \tag{2.10}$$

Come si può vedere dal suo grafico in Fig. 2.1, la funzione assume valori in $(0, 1)$; alcune proprietà utili:

$$\lim_{t \to -\infty} \sigma(t) = 0; \qquad \lim_{t \to +\infty} \sigma(t) = 1; \qquad \sigma(t) = 1 - \sigma(-t); \qquad \sigma'(t) = \sigma(t)\big(1 - \sigma(t)\big).$$

Dato che la funzione restringe la variabilità del proprio argomento all'intervallo aperto $(0, 1)$, può essere utilizzata per modellare un problema di classificazione a due classi. In particolare, se assegniamo i valori 0 e 1 alle due classi, possiamo utilizzare un modello lineare, filtrandone il risultato attraverso una sigmoide:

$$y \sim \sigma\left(\boldsymbol{\beta} \cdot \boldsymbol{x}\right).$$

Il modello è detto *regressione logistica*, o *logit*[1]. Dato che il risultato può variare con continuità fra 0 e 1, il modello può esprimere una probabilità.

[1] `https://en.wikipedia.org/wiki/Logistic_regression`

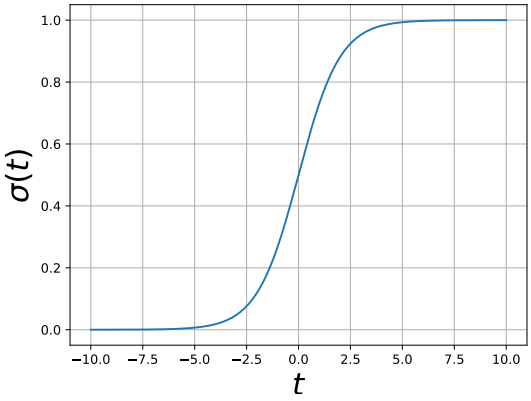


Figura 2.1: La funzione sigmoide

2.3.1 Stima dei coefficienti del modello

Proviamo a stimare il vettore dei coefficienti $\boldsymbol{\beta}$ che minimizza gli scarti al quadrato[2]. Sia

$$\tilde{y}_i = \sigma(\boldsymbol{\beta} \cdot \boldsymbol{x}_i),$$

dove $\boldsymbol{x}_i$ è il vettore di attributi dell'i-esimo campione. Allora vogliamo trovare il vettore $\boldsymbol{\beta}$ che minimizza le somme degli scarti:

$$f(\boldsymbol{\beta}) = \sum_{i=1}^{m} (\tilde{y}_i - y_i)^2.$$

Come nel caso della regressione lineare, possiamo trovare le derivate parziali rispetto ai coefficienti:

$$\frac{\partial}{\partial \beta_k}(\boldsymbol{\beta}) = 2 \sum_{i=1}^{m} (\tilde{y}_i - y_i)\tilde{y}_i(1 - \tilde{y}_i)x_{ik} \qquad \forall k = 1, \ldots, n.$$

L'azzeramento di tutte queste equazioni non lineari in $\boldsymbol{\beta}$ non porta a un sistema di equazioni risolvibili analiticamente. Per minimizzare la RMSE, dunque, utilizziamo il metodo della *discesa lungo il gradiente* (gradient descent)[3]. Ricordiamo che il gradiente di una funzione scalare in n variabili $\beta_1, \ldots, \beta_n$ è il vettore delle derivate parziali rispetto agli argomenti:

$$\boldsymbol{\nabla} f = \left(\frac{\partial f}{\partial \beta_1}, \frac{\partial f}{\partial \beta_2}, \ldots, \frac{\partial f}{\partial \beta_n} \right).$$

Il gradiente di f rappresenta la direzione di massima crescita della funzione. Per minimizzare la funzione, dunque, applichiamo il seguente metodo:

- Inizializza $\boldsymbol{\beta}$ con valori casuali.
- Ripeti:
  - Calcola $\boldsymbol{\nabla} f(\boldsymbol{\beta})$;
  - Sposta $\boldsymbol{\beta}$ di un piccolo passo in direzione opposta al gradiente:

$$\boldsymbol{\beta} \leftarrow \boldsymbol{\beta} + \eta \boldsymbol{\nabla} f(\boldsymbol{\beta}).$$

[2]Trattandosi di una stima di probabilità, il modo corretto sarebbe una stima di massima verosimiglianza (maximum likelihood), ma non introdurremo qui il concetto.

[3]`https://en.wikipedia.org/wiki/Gradient_descent`

Il parametro η, detto *learning rate*, serve a evitare di spostare troppo rapidamente $\boldsymbol{\beta}$: l'approssimazione che lega il gradiente alla direzione di discesa vale solo localmente, e salti troppo ampi possono causare peggioramenti. Versioni più sofisticate dell'algoritmo di base prevedono che η possa variare in base ai risultati ottenuti, oppure che diminuisca col tempo. Il numero di iterazioni del metodo può essere impostato a priori; in alternativa, una condizione di terminazione potrebbe essere il raggiungimento di un livello di errore prefissato, o la mancanza di miglioramenti significativi da un certo numero di iterazioni.

2.3.2 Utilizzo del modello

Previsioni di classi

Il modello logistico può essere usato per restituire una probabilità. Nel caso considerato, $\tilde{y}$ stima la probabilità $\Pr(y = 1)$. Se vogliamo invece che il modello restituisca un valore di classe (0 oppure 1), dobbiamo fissare un valore di soglia $\theta \in [0, 1]$ ed applicare una *funzione di decisione*. Dato il vettore di attributi $\boldsymbol{x}$ e il corrispondente valore ottenuto dal modello logit $\tilde{y} = \sigma(\boldsymbol{x} \cdot \boldsymbol{\beta})$, la previsione sarà data dalla funzione a soglia:

$$\text{Previsione}_\theta(\boldsymbol{x}) = \begin{cases} 0 & \text{se } \sigma(\boldsymbol{x} \cdot \boldsymbol{\beta}) < \theta \\ 1 & \text{altrimenti.} \end{cases} \tag{2.11}$$

Reti neurali

Un singolo regressore (lineare o logistico) può essere visto come un modello estremamente semplificato di *neurone*, con gli ingressi in luogo dei dendriti, l'uscita come assone e i coefficienti $\boldsymbol{\beta}$ in luogo della funzione di attivazione della sinapsi ($\beta_i < 0$ se la sinapsi inibisce, $\beta_i > 0$ se eccita). La combinazione di più regressori, che in questo contesto sono detti "percettroni"[4], costituisce una *rete neurale artificiale*[5].

[4] `https://en.wikipedia.org/wiki/Perceptron`
[5] `https://en.wikipedia.org/wiki/Artificial_neural_network`

Capitolo 3

Algoritmi di classificazione

3.1 K-Nearest-Neighbors (KNN)

Supponiamo di avere un problema di classificazione (y_i da un insieme finito). Se abbiamo un dataset $D = \{(\boldsymbol{x}_i, y_i)\}$ di esempi e ci viene fornito un nuovo elemento $\boldsymbol{x}$ per il quale prevedere il valore di y, possiamo decidere che la classe dell'elemento incognito sia uguale alla classe dell'elemento più simile che troviamo in D:

- cerchiamo l'elemento del dataset $(\boldsymbol{x}_i, y_i) \in D$ in cui $\boldsymbol{x}_i$ è più simile al nuovo $\boldsymbol{x}$;
- assumiamo che la classe dell'elemento incognito sia uguale alla classe di quest'elemento: $y = y_i$.

La motivazione di quest'algoritmo è che la classe non è distribuita *casualmente* fra gli elementi (il che renderebbe completamente inutile qualunque tentativo di prevederla); al contrario, è probabile che elementi simili in $\boldsymbol{x}$ siano simili anche in y. In altre parole, assumiamo una certa "continuità" nella mappa $\boldsymbol{x} \mapsto y$.

Per ovviare a possibili errori nel dataset e per "ammorbidire" i risultati nelle sone di confine fra una classe e l'altra, possiamo prendere più di un elemento "simile" a $\boldsymbol{x}$ e scegliere la classe più rappresentata al loro interno[1]:

- sia $N = \{i_1, \ldots, i_K\}$ l'insieme degli indici dei K elementi del dataset più simili a $\boldsymbol{x}$;
- scegliere per y la classe che appare più spesso nella sequenza $y_{i_1}, \ldots, y_{i_K}$.

La "somiglianza" di un elemento del dataset $\boldsymbol{x}_i$ con l'elemento incognito $\boldsymbol{x}$ può essere determinata in base alla loro distanza euclidea:

$$\mathrm{d}(\boldsymbol{x}_i, \boldsymbol{x}) = \|\boldsymbol{x}_i - \boldsymbol{x}\|_2.$$

Ovviamente, due elementi sono tanto più simili quanto la loro distanza è piccola. Si veda un esempio di implementazione in Python nella seconda esercitazione di laboratorio (capitolo 2).

3.1.1 Applicazione di KNN a problemi di regressione

L'algoritmo di classificazione KNN decide il valore dell'output in base alla classe più rappresentata fra i primi K vicini. Se l'output è continuo, la decisione a maggioranza non ha più senso (i valori possono essere tutti diversi); in tal caso si può assumere come valore dell'output la media delle y, modificando in tal senso l'algoritmo:

- sia $N = \{i_1, \ldots, i_K\}$ l'insieme degli indici dei K elementi del dataset più simili a $\boldsymbol{x}$;
- scegliere per y la media dei valori di output $y_{i_1}, \ldots, y_{i_K}$.

[1] `https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm`

In altri termini, se $N = \{i_1, \ldots, i_K\}$ sono gli indici dei K primi vicini del campione $\boldsymbol{x}$, la classe del campione può essere prevista come

$$\tilde{y} = \frac{1}{K} \sum_{i \in N} y_i,$$

oppure pesando la media con pesi che decrescono al crescere della distanza fra gli elementi del dataset e il campione incognito:

$$\tilde{y} = \frac{\displaystyle\sum_{i \in N} \frac{y_i}{\|\boldsymbol{x} - \boldsymbol{x}_i\| + \epsilon}}{\displaystyle\sum_{i \in N} \frac{1}{\|\boldsymbol{x} - \boldsymbol{x}_i\| + \epsilon}}$$

dove un piccolo valore ϵ è sommato ai denominatori per evitare divisioni per zero.

Capitolo 4

Valutare la bontà di un modello

In questo capitolo raccoglieremo varie osservazioni su come valutare se un modello, determinato sulla base di un insieme di esempi, è utile allo scopo di effettuare previsioni su dati nuovi oppure no.

Assumiamo, per ora, che il dataset complessivo D sia stato ripartito in due sottoinsiemi:

- un sottoinsieme T, detto di *addestramento* (training), utilizzato per costruire il modello;
- un sottoinsieme V, detto di *validazione*, usato per valutare se il modello è adeguato.

I due sottoinsiemi costituiscono una *partizione* di D (la loro unione è D e la loro intersezione è nulla).

Per valutare la bontà di un algoritmo, utilizzeremo i dati contenuti in T per costruire il modello f. Valuteremo poi il modello f sui dati di V. Si osservi che i dati di V non sono utilizzati nella definizione del modello, ma sono "tenuti da parte" per valutarlo.

4.1 Criteri di valutazione

4.1.1 Valutare un modello di regressione

Supponiamo che la variabile dipendente y sia un valore reale che varia con continuità in un intervallo. Dato il modello f, costruito in base agli esempi contenuti in T, e un esempio di validazione $(\boldsymbol{x}, y) \in V$, l'errore che il modello commette su questo campione può essere definito come la differenza fra il valore previsto $f(\boldsymbol{x})$ e il valore effettivo y:

$$\epsilon = f(\boldsymbol{x}) - y.$$

L'errore quadratico medio (root mean square error, RMSE) è calcolato su tutti i campioni dell'insieme di validazione:

$$\text{RMSE} = \sqrt{\frac{1}{|V|} \sum_{(\boldsymbol{x},y)\in V} \big(f(\boldsymbol{x}) - y\big)^2}.$$

Trattandosi di un errore, il modello è tanto migliore quanto questo valore è piccolo. Si veda ad esempio la terzaesercitazione di laboratorio (capitolo 3).

4.1.2 Valutare un modello di classificazione

Nel caso della classificazione, l'errore commesso su un singolo esempio di validazione non varia con continuità, ma è binario (o c'è, o non c'è).

Accuratezza

Una prima valutazione possibile è il conteggio del numero di errori commessi dal modello (rapportato alla sua numerosità). Si tratta del cosiddetto "tasso di errore" (error ratio):

$$\text{Tasso d'errore} = \frac{|\{(\boldsymbol{x}, y) \in V : f(\boldsymbol{x}) \neq y\}|}{|V|}.$$

La misura complementare al tasso di errore è detta *accuratezza* (accuracy) e conta il numero di previsioni corrette, sempre rapportato al numero totale di esempi in V:

$$\text{Accuratezza} = \frac{|\{(\boldsymbol{x}, y) \in V : f(\boldsymbol{x}) = y\}|}{|V|} = 1 - \text{Tasso d'errore}.$$

Matrice di confusione

Non sempre gli errori hanno la stessa importanza. Si pensi al problema di classificare i funghi tra "velenosi" e "mangerecci". Un errore in un senso comporta lo spreco di un fungo; un errore nell'altro senso comporta una possibile intossicazione. Un modello con un'accuratezza del 99% può essere ottimo, ma quell'1% di errore può avere conseguenze molto diverse.

Immaginiamo che il modello debba prevedere la velenosità di un fungo. Alla domanda "questo fungo è velenoso?", il sistema può dare una risposta *positiva* o *negativa*. La risposta fornita può essere corretta oppure no. Possiamo dunque distribuire le previsioni del nostro modello nella seguente "matrice (o tabella) di confusione"[1]:

		Previsione	
		Sì	No
Risposta corretta	Sì	TP	FN
	No	FP	TN

Le quattro caselle della tabella contano il numero di previsioni positive corrette (veri positivi, *true positives*, TP), risposte positive errate (falsi positivi, *false positives*, FP), risposte negative errate (falsi negativi, *false negatives*, FN) e risposte negative corrette (veri negativi, *true negatives*, TN).

Si noti che, nella nomenclatura, i termini "positivo" e "negativo" si riferiscono alla previsione fornita dal modello, mentre i termini "vero" e "falso" fanno riferimento alla correttezza di queste risposte rispetto alle risposte esatte contenute in V, note al sistema di valutazione, ma non al modello. Si tratta di contatori, e

$$\text{TP} + \text{FP} + \text{TN} + \text{FN} = |V|.$$

La matrice di confusione contiene tutte le informazioni necessarie a ricavare numerose stime di bontà di un classificatore. Ad esempio, l'accuratezza sopra definita può essere definita come

$$\text{Accuratezza} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}.$$

Sensibilità

Possiamo ottenere anche misure più "specializzate"; ad esempio, nel caso del modello per la classificazione dei funghi vogliamo essere sicuri che il sistema non fornisca falsi negativi (che non risponda di "no" in presenza di un fungo velenoso), cioè che colga tutti gli esempi positivi. In questo caso ci interessa un modello con la *sensibilità* (sensitivity) il più elevata possibile:

$$\text{Sensibilità} = \frac{\text{TP}}{\text{TP} + \text{FN}}.$$

Questa misura trascura completamente i funghi "negativi" (mangerecci) e si limita a contare quante volte, fra i funghi "positivi" (velenosi) il modello ha fornito la risposta corretta. Altrimenti detto, questa misura considera solo la prima riga della tabella di confusione.

Si osservi che è facile realizzare un modello con sensibilità massima: è sufficiente che risponda "Sì" a ogni istanza che gli viene presentata. È facile, insomma, realizzare un sistema che elimina tutti i funghi velenosi: basta buttarli via tutti. Le istanze positive saranno tutte individuate, e la sensibilità sarà del 100%. La sensibilità, quindi, è una misura poco utile in sé, e va sempre associata a qualche altra misura nella valutazione della bontà del modello.

In alcuni contesti, la sensibilità è detta "Recall", in quanto misura della capacità del modello di "richiamare" tutti i casi positivi.

[1] `https://en.wikipedia.org/wiki/Confusion_matrix`

Precisione

Un'altra misura, utile in altre situazioni (ad esempio nella valutazione dei risultati forniti da un motore di ricerca), è la *precisione* (precision):

$$\text{Precisione} = \frac{\text{TP}}{\text{TP} + \text{FP}}.$$

Questa misura conta quante volte le risposte positive del modello sono corrette.

Anche in questo caso è possibile realizzare un sistema con precisione massima: basta che il modello risponda "Sì" soltanto nei casi in cui è assolutamente certo della risposta (ad esempio, quando vede un'*Amanita Muscaria* rossa a pallini bianchi), rispondendo sempre "No" in caso di incertezza. Tutte le sue risposte positive saranno corrette (con alta probabilità) e la precisione sarà pari al 100%.

Misure combinate

Per poter tenere conto sia della precisione che della sensibilità di un modello in un'unica misura, si utilizza la F_1-*score*, calcolata come media armonica delle due:

$$F_1\text{-score} = \frac{1}{\frac{\frac{1}{\text{Sensibilità}} + \frac{1}{\text{Precisione}}}{2}} = \frac{2 \cdot \text{Sensibilità} \cdot \text{Precisione}}{\text{Sensibilità} + \text{Precisione}}.$$

La F_1-score è un caso particolare (con pesi unitari) della F_β-score, in cui β^2 è il peso della sensibilità.

Un'altra misura frequentemente utilizzata è il coefficiente di correlazione di Matthews (Matthews correlation coefficient, MCC):

$$\text{MCC} = \frac{\text{TP} \cdot \text{TN} - \text{FP} \cdot \text{FN}}{\sqrt{(\text{TP} + \text{FP})(\text{TP} + \text{FN})(\text{TN} + \text{FP})(\text{TN} + \text{FN})}}$$

A differenza delle altre misure, il MCC varia in $[-1, 1]$.

Classificazione a più di due classi

La definizione di matrice di confusione può facilmente essere generalizzata a più di due classi, con una riga e una colonna per ogni valore di classe; la tabella ha una casella per ogni combinazione di valore predetto e valore reale; la diagonale contiene le previsioni corrette.

Solitamente, si considera a turno ogni classe come "positiva" e tutte le altre negative, ottenendo quindi una diversa matrice di confusione binaria, con relative misure di precisione e sensibilità per ogni classe.

4.2 Addestramento e validazione

La definizione degli insiemi di addestramento (training) e di validazione non è semplice. Si vorrebbe che l'insieme di training fosse il più vasto possibile, in modo da ottenere un modello attendibile, ma è necessario che l'insieme di validazione non sia troppo piccolo, altrimenti le misure risentono di una varianza eccessiva. Esistono varie tecniche di suddivisione del dataset, dette di *cross-validation*[2].

4.2.1 K-fold cross-validation

Si ripartisce il dataset in K insiemi di uguale numerosità (± 1), detti *fold*, $D = F_1 \cup \cdots \cup F_K$.

- Per $i = 1, \ldots, K$
  - Sia $T_i \leftarrow D \setminus F_i$ (tutti i fold tranne l'i-esimo sono usati come apprendimento);

[2] `https://en.wikipedia.org/wiki/Cross-validation_(statistics)`

  - Addestrare il modello f_i usando T_i;
  - Utilizzare il modello f_i per prevedere i valori "tenuti da parte": $\tilde{y} = f_i(\boldsymbol{x})$, $\boldsymbol{x} \in F_i$ (usando dunque l'i-esimo fold F_i, tenuto fuori dalla fase di apprendimento, come insieme di validazione).

- Calcolare l'errore di predizione.

Si tratta di un metodo statisticamente robusto e molto utilizzato. Valori comuni per il parametro K sono $K = 5$ e $K = 10$.

Stratificazione

Se il problema è di classificazione e una delle classi è molto meno frequente delle altre, può accadere che la "concentrazione" della classe meno numerosa vari molto da un fold all'altro, comportando stime dell'errore molto rumorose. Lo stesso effetto si può avere anche con classi bilanciate, ma dataset piccoli.

In questi casi è opportuno che la procedura di suddivisione del dataset nei diversi fold mantenga costanti i rapporti di numerosità delle varie classi, ad esempio operando la suddivisione separatamente per ciascuna classe. Si parla in questo caso di suddivisione *stratificata*[3].

4.2.2 Una soluzione estrema: l'approccio *leave-one-out*

Invece di utilizzare una singola partizione come insieme di addestramento, ripetiamo l'addestramento m volte, ogni volta tenendo da parte un diverso campione da usare per la validazione. Stimiamo l'errore su tutte le previsioni effettuate dai diversi modelli.

- Per $i = 1, \ldots, m$
  - Sia $T_i \leftarrow D \setminus \{(\boldsymbol{x}_i, y_i)\}$ (tutto il dataset tranne l'i-esima riga è usato come addestramento);
  - Addestrare il modello f_i usando T_i;
  - Usare il modello f_i per prevedere il valore "tenuto da parte": $\tilde{y}_i = f_i(\boldsymbol{x}_i)$.
- Calcolare l'errore di predizione.

La metodologia leave-one-out è un caso estremo di K-fold cross-validation con $K = m$.

Vantaggi: ogni addestramento utilizza quasi l'intero dataset, quindi è rappresentativo di tutti i dati raccolti; la validazione copre l'intero dataset, quindi è una stima molto robusta dell'errore che ci si può aspettare su nuovi dati. Svantaggi: richiede l'addestramento di m modelli diversi, con m potenzialmente molto elevato.

[3] `https://en.wikipedia.org/wiki/Stratified_sampling`, parte introduttiva

Capitolo 5

Pretrattare i dati

5.1 K-nearest-neighbors

Abbiamo già visto questo algoritmo alla Sezione 3.1. In questa sezione descriveremo alcuni "trucchi" per applicare l'algoritmo a problemi con caratterisctiche diverse.

5.1.1 Input categorici

Finora abbiamo sempre assunto che il vettore degli attributi (ingresso) sia sempre composto di quantità reali: $\boldsymbol{x} \in \mathbb{R}^n$. Può accadere però che alcune osservazioni non riguardino grandezze continue, ma qualità discrete e non ordinate (colore, sesso, specie).

L'algoritmo KNN si basa però sul calcolo di una distanza in uno spazio $\mathbb{R}^n$, quindi è necessario convertire le grandezze categoriche in una o più grandezze reali. Il metodo più intuitivo consiste nel sostituire a ogni colonna categorica del dataset tante colonne reali quanti sono i valori che la categoria può assumere, e utilizzare una rappresentazione unaria per decidere i valori delle colonne sostitutive.

Ad esempio, supponiamo che il seguente dataset descriva dei fiori, con le seguenti osservazioni: lunghezza del petalo (continua), colore (classi "rosso", "giallo" o "azzurro"), altezza (continua), presenza di spine ("sì", "no"). Ecco come un dataset può essere trasformato in questo caso:

1	2.1	rosso	5.0	no
2	3.2	azzurro	5.4	no
3	2.6	rosso	7.2	sì
4	1.9	giallo	4.4	no
5	3.0	azzurro	5.6	sì
6	2.5	giallo	6.0	no

$$\Rightarrow \qquad X = \begin{pmatrix} 2.1 & 1 & 0 & 0 & 5.0 & 0 \\ 3.2 & 0 & 0 & 1 & 5.4 & 0 \\ 2.6 & 1 & 0 & 0 & 7.2 & 1 \\ 1.9 & 0 & 1 & 0 & 4.4 & 0 \\ 3.0 & 0 & 0 & 1 & 5.6 & 1 \\ 2.5 & 0 & 1 & 0 & 6.0 & 0 \end{pmatrix}$$

La colonna del colore è stata spezzata in tre colonne numeriche; la prima codifica il valore "rosso" (vale 1 se l'attributo categorico è "rosso", 0 altrimenti), e così via. Come eccezione, se l'attributo è binario può essere trasformato in una sola colonna numerica, nella quale 0 e 1 codificano i due possibili valori categorici.

5.2 Normalizzazione e standardizzazione dei dati

Osserviamo che spesso, in un dataset, le diverse colonne di ingresso hanno diversi ordini di grandezza, con valori che pososno spaziare nel campo delle unità o nelle migliaia a seconda del loro significato e delle unità di misura in cui sono espresse. Per evitare problemi di instabilità numerica, è spesso importante "normalizzare" le colonne del dataset in modo che i valori abbiamo lo stesso intervallo di variabilità.

5.2.1 Normalizzazione

Il caso più semplice è quello in cui si porta una colonna di dati a variare nell'intervallo $[0, 1]$. Data la colonna $j = 1, \ldots, n$ del nostro dataset, siano

$$m_j = \min_{i=1,\ldots,m} x_{ij}; \qquad M_j = \max_{i=1,\ldots,m} x_{ij}$$

il minimo e il massimo della colonna j. La normalizzazione della matrice avviene dunque con la seguente mappa affine:

$$x'_{ij} = \frac{x_{ij} - m_j}{M_j - m_j}.$$

In alcuni casi, il valore 0, anche se non compare mai nella matrice, ha un significato particolare e si desidera mantenerlo. Allora, è sufficiente porre $m_j = 0$.

5.2.2 Standardizzazione

Un difetto della normalizzazione è la sua sensibilità agli "outlier": se nella colonna è presente un valore molto più grande degli altri, questo viene mappato sul valore normalizzato 1, mentre tutti gli altri valori sono mappati vicino allo zero. Se si desidera una tecnica più "morbida", è possibile ricorrere alla "standardizzazione", cioè si riporta la colonna a una distribuzione di media nulla e varianza unitaria. Siano, per ogni colonna $j = 1, \ldots, n$

$$\mu_j = \frac{1}{m} \sum_{i=1}^{m} x_{ij}; \qquad \sigma_j = \sqrt{\frac{1}{m} \sum_{i=1}^{m} (x_{ij} - \mu_j)^2}$$

la media e la deviazione standard dei dati della colonna j. La standardizzazione avviene con la seguente mappa affine:

$$x'_{ij} = \frac{x_{ij} - \mu_j}{\sigma_j}.$$

5.2.3 Insiemi di validazione

Un'avvertenza: se il dataset è suddiviso fra insiemi di addestramento e validazione, bisogna assicurarsi che i minimi m_j e i massimi M_j siano calcolati soltanto sulle righe di addestramento. Infatti, i valori dell'insieme di validazione non devono influire in alcun modo nel training. Gli stessi minimi e massimi andranno poi utilizzati anche nella normalizzazione dei valori di validazione, che potrebbero quindi uscire dall'intervallo $[0, 1]$.

In generale, una volta ottenuti i parametri per la normalizzazione (m_j e M_j) o per la standardizzazione (μ_j e σ_j), è fondamentale ricordarli in modo da applicare le stesse trasformazioni anche a nuovi insiemi di dati.

Capitolo 6

Alberi di decisione

Un *albero di decisione* è un albero, tipicamente binario, nel quale ogni nodo interno rappresenta una possibile domanda a risposta (binaria o multipla), mentre le foglie rappresentano delle *decisioni* (ad esempio, una classe da attribuire a un elemento incognito). Nel contesto del machine learning, un albero rappresenta una mappa $y \sim f(\boldsymbol{x})$: a partire dalla radice, ciascun nodo interno è una domanda relativa al vettore di attributi $\boldsymbol{x}$ (tipicamente nella forma "$x_j \leq \theta$?") e ogni foglia rappresenta una stima $\tilde{y}$ del valore di y (o una distribuzione di probabilità sui suoi possibili valori).

6.1 Addestramento di un albero di ricerca

Nel seguito, consideriamo la variabile casuale Y da cui sono estratti i valori di uscita del dataset D. Alcune osservazioni di base sulla "prevedibilità" di Y sono le seguenti:

- se Y ha un solo valore, ovvero se ha più valori, ma uno solo di essi ha probabilità pari a 1, allora la variabile è prevedibile senza bisogno di ulteriori informazioni;
- più in generale, se un valore è molto più probabile degli altri (sole, pioggia o neve nel mezzo del Sahara oggi?), allora è possibile "azzardare" una previsione anche in assenza di informazioni aggiuntive (gli attributi $\boldsymbol{x}$);
- più Y è vicina all'uniformità (tutti i suoi valori sono equiprobabili), più difficile è azzardare una previsione.

6.1.1 Prima misura di imprevedibilità: l'entropia di Shannon

Supponiamo che Y abbia ℓ valori diversi, di probabilità $p_1, \ldots, p_\ell$; Alice e Bob conoscono la distribuzione di probabilità; Alice osserva una sequenza di eventi estratti da Y e vuole comunicare questa sequenza a Bob. Quanti bit deve usare Alice, come minimo? Consideriamo alcuni casi semplici.

- Se $p_1 = 1$ e $p_2 = \cdots = p_\ell = 0$, allora non c'è bisogno di inviare informazioni: Bob sa già che ogni evento risulterà nell'unico valore certo.
- Se ℓ è una potenza di 2 ($\ell = 2^r$) e $p_1 = \cdots = p_\ell = \frac{1}{\ell}$, allora il meglio che Alice possa fare è codificare ogni valore di Y con una diversa combinazione di r bit.
- Supponiamo che $\ell = 4$ e che Y abbia la seguente disribuzione:

  $$p_1 = \frac{1}{2}, \qquad p_2 = \frac{1}{4}, \qquad p_3 = p_4 = \frac{1}{8}.$$

  Allora è possibile adottare una codifica, detta di Huffman[1], in cui la lunghezza del codice dipende dalla probabilità del valore:

  $$1 \mapsto 0, \qquad 2 \mapsto 10, \qquad 3 \mapsto 110, \qquad 4 \mapsto 111.$$

[1] `https://en.wikipedia.org/wiki/Huffman_coding`

Questa codifica permette di ridurre il numero atteso di bit da spedire al seguente valore:

$$\overbrace{\frac{1}{2}\cdot 1}^{\text{1 bit se } Y=1} \quad + \quad \overbrace{\frac{1}{4}\cdot 2}^{\text{2 bit se } Y=2} \quad + \quad \overbrace{\frac{1}{8}\cdot 3}^{\text{3 bit se } Y=3} \quad + \quad \overbrace{\frac{1}{8}\cdot 3}^{\text{3 bit se } Y=4} \quad = \frac{7}{4} = 1.75,$$

il che costituisce un miglioramento rispetto all'uso della codifica uniforme a 2 bit.

Il numero atteso di bit da trasmettere rappresenta una misura dell'uniformità della variabile casuale. Si noti che, in tutti i casi elencati sopra, il numero di bit da trasmettere per comunicare l'esito i della variabile casuale è pari a

$$b(i) = \log_2 \frac{1}{p_i} = -\log_2 p_i. \tag{6.1}$$

Un risultato fondamentale della Teoria dell'Informazione di Shannon è precisamente che l'equazione (6.1) è vera in generale. Quindi, il numero atteso di bit necessari a trasmettere un evento estratto dalla variabile casuale Y è

$$\mathrm{H}(Y) = -\sum_{i=1}^{\ell} p_i \log_2 p_i. \tag{6.2}$$

La grandezza $\mathrm{H}(Y)$ è detta *entropia di Shannon*[2] della variabile casuale Y. Se la distribuzione di probabilità di Y è concentrata in un singolo valore di probabilità 1, allora $\mathrm{H}(Y) = 0$. Invece, l'entropia è massima quando Y è uniforme, e vale $\mathrm{H}(Y) = \log_2 \ell$.

6.1.2 Seconda misura di imprevedibilità: l'impurità di Gini

Supponiamo che Alice osservi un evento i estratto da Y. Bob cerca di indovinare i generando un valore casuale $\tilde{\imath}$ con la stessa distribuzione di probabilità, a lui nota. Qual è la probabilità che Bob sbagli?

- Se Y ha un solo valore di probabilità 1, allora Bob indovina di certo.
- Più in generale, se Y ha un valore molto più probabile degli altri, è poco probabile che Bob sbagli.
- Intuitivamente, la probabilità di errore è massima quando la distribuzione di Y è uniforme.

Se Alice osserva il valore i Bob genererà il valore corretto con probabilità p_i, quindi sbaglierà con probabilità $1 - p_i$. La probabilità di errore complessiva, mediata su tutti i possibili esiti, è

$$\mathrm{GI}(Y) = \sum_{i=1}^{\ell} p_i(1 - p_i) = 1 - \sum_{i=1}^{\ell} p_i^2. \tag{6.3}$$

La grandezza $\mathrm{GI}(Y)$ è detta *impurità di Gini*[3] della variabile casuale Y. Se la distribuzione di probabilità di Y è concentrata in un singolo valore di probabilità 1, allora $\mathrm{GI}(Y) = 0$. Invece, l'impurità di Gini è massima quando Y è uniforme, e vale $\mathrm{GI}(Y) = 1 - \frac{1}{\ell}$, valore che tende asintoticamente a 1 al crescere del dominio di Y.

6.2 Passo induttivo: riduzione dell'imprevedibilità

A partire da un dataset D, la cui variabile di output è modellata dalla variabile casuale Y, vogliamo spezzarlo in due parti, sulla base di un criterio della forma $x_j < \theta$. Chiamiamo la partizione risultante $(D_{x_j<\theta}, D_{x_j\geq\theta})$. Ciascuno dei due sottoinsiemi del dataset induce una diversa variabile casuale di output, $Y|x_j < \theta$ e $Y|x_j \geq \theta$. La partizione dipende dunque da due parametri, j e θ. Vogliamo che

[2] `https://en.wikipedia.org/wiki/Entropy_(information_theory)` — si noti che la lettera H è in realtà una Eta maiuscola.

[3] `https://en.wikipedia.org/wiki/Decision_tree_learning#Gini_impurity`, da non confondere con il *coefficiente di Gini* usato in econometria.

l'impurità delle due variabili casuali indotte sia minima, in particolare che lo sia l'impurità "attesa", basata sulla numerosità dei due sottoinsiemi. Se utilizziamo l'entropia come misura di impurità, allora vogliamo trovare j e θ che minimizzano il seguente valore atteso:

$$E_{j,\theta}(\mathrm{H}) = \frac{|D_{x_j<\theta}|}{|D|}\mathrm{H}(Y|x_j < \theta) + \frac{|D_{x_j\geq\theta}|}{|D|}\mathrm{H}(Y|x_j \geq \theta). \tag{6.4}$$

Il criterio di minimizzazione dell'entropia attesa è detto "Information Gain Criterion". Tecnicamente, l'information gain è la diminuzione dell'entropia passando dal dataset intero ai due sotto-dataset:

$$\mathrm{IG}_{j,\theta}(Y) = \mathrm{H}(Y) - E_{j,\theta}(\mathrm{H}). \tag{6.5}$$

È chiaro che minimizzare l'entropia attesa (6.4) equivale a massimizzare il guadagno informativo (6.5).

Allo stesso modo, possiamo utilizzare l'impurità di Gini GI come criterio da minimizzare.

In questo modo, l'informazione "$x_j < \theta$" permette di fare riferimento a uno di due sotto-dataset nei quali l'incertezza sul valore di Y è ridotta.

Come abbiamo visto dalle formule (6.2) e (6.3), la stima dell'entropia e dell'impurità di Gini richiederebbero la conoscenza delle distribuzioni di probabilità delle variabili casuali, che noi possiamo stimare sulla base delle frequenze dei valori assunti.

6.3 Partizione ricorsiva del dataset e condizione di terminazione

Per ottenere un albero, detto "albero di decisione" (*decision tree*)[4], possiamo applicare ripetutamente il passo induttivo a ciascun sotto-dataset, riducendo di volta in volta l'impurità. La procedura termina "naturalmente" quando il sotto-dataset associato a un nodo contiene un solo elemento: a questo punto la variabile casuale associata all'output ha un solo valore quindi è necessariamente pura.

Per limitare la profondità dell'albero risultante, è possibile introdurre ulteriori criteri di terminazione per il passo induttivo:

- quando la profondità dell'albero raggiunge un limite massimo;
- quando l'impurità di un nodo è inferiore a una soglia predefinita;
- quando il numero di elementi nel sotto-dataset associato a un nodo è inferiore a un minimo predefinito.

Riassumendo, l'algoritmo per costruire un albero di decisione è il seguente:

1. Se il dataset D soddisfa un criterio di terminazione, non fare nulla. Altrimenti:
   (a) per ogni combinazione j, θ:
      i. Calcola i due sotto-dataset $D_{x_j<\theta}$ e $D_{x_j\geq\theta}$;
      ii. Calcola l'impurità attesa usando (6.4) o una sua variante;
      iii. Se l'impurità è la minore trovata finora, ricorda i valori ottimali $j^* \leftarrow j, \theta^* \leftarrow \theta$.
   (b) Associa i parametri migliori (j^*, θ^*) alla radice e genera due figli, sinistro e destro, associati rispettivamente a $D_{x_{j^*}<\theta^*}$ e $D_{x_{j^*}\geq\theta^*}$.
   (c) Applica ricorsivamente la procedura ai figli sinistro e destro.

In presenza di un vettore di attributi $\boldsymbol{x} = (x_1, \ldots, x_n)$, la procedura per "decidere" il corrispondente valore di y è la seguente:

1. Se il nodo corrente è una foglia, allora:
   - y è la classe prevalente nel sotto-dataset associato a quel nodo.

[4] `https://en.wikipedia.org/wiki/Decision_tree_learning`

2. Altrimenti:

   - Se $x_j < \theta$, ripeti la procedura sul sottoalbero di sinistra;
   - Altrimenti scendi a destra.

Si rinvia all'esercitazione 5 per l'implementazione "ingenua" di un algoritmo di creazione ed utilizzo di un albero di decisione, e per l'uso di una libreria e la rappresentazione grafica dell'albero.

6.3.1 Variabili in input categoriche: l'algoritmo ID3

Se la variabile x_j in ingresso è categorica a due valori $\{v_1, v_2\}$, l'algoritmo già descritto richiede la sola sostituzione della domanda "$x_j < \theta$" con la domanda "$x_j = v_1$". Se la variabile ha più di due valori, sono possibili due varianti:

1. la domanda assume la forma "$x_j = v_k$", quindi pone un valore specifico della classe contro tutti gli altri, oppure
2. l'albero non è più binario, ma il nodo ha tanti sottoalberi quanti sono i valori che x_j può assumere.

Quest'ultima variante, quand'è applicata a variabili di ingresso puramente categoriche, è solitamente nota come "algoritmo ID3". Si osservi come, discendendo l'albero risultante, ogni colonna di input x_j venga usata una sola volta, perchè a monte di una domanda il valode di x_j è determinato, quindi l'albero non può avere profondità maggiore del numero di colonne n del dataset.

Capitolo 7

Selezione degli attributi

Spesso, l'insieme degli attributi presentati da un dataset è sovrabbondante: molti attributi possono essere irrilevanti per la predizione dell'output, altri possono essere inutilmente ripetitivi. La necessità di trattare un numero eccessivo di attributi comporta un certo numero di problemi:

- i tempi di apprendimento (ovvero di costruzione del modello) si allungano;
- il rischio di overfit è maggiore, in quanto il modello ha più informazioni sulle quali "specializzarsi";
- più dati possono comportare più rumore, quindi predizioni meno precise.

È dunque importante saper riconoscere a priori quali attributi sono più importanti per saper prevedere l'output.

Dal punto di vista dell'utilizzo in un contesto più ampio, come componenti di un sistema di apprendimento automatico, gli algoritmi di selezione degli attributi possono distinguersi in:

- Metodi "wrapper": dato un modello, lo utilizzano per capire in che modo ogni attributo o combinazione di attributi influiscono sul risultato; sono computazionalmente costosi, ma visto che possono prendere decisioni basate sulle prestazioni del modello sono molto efficaci. Ad esempio, un sistema che riaddestra una rete neurale con sottoinsiemi di attributi diversi cercando di minimizzare l'errore.
- Metodi "filter": applicati prima di addestrare un modello, utiizzano misure statistiche (correlazione, informazione mutua — vedi sotto) per cercare di prevedere l'influenza dei vari attributi sulla variabile di uscita.
- Metodi "embedded": sono parte integrante dell'algoritmo di addestramento del modello; ad esempio, la selezione dell'attributo da utilizzare a un certo nodo di un albero di decisione può rientrare in questa categoria, e una volta che l'albero è stato addestrato gli attributi "selezionati" sono quelli che vengono effettivamente utilizzati in qualche nodo.

Gli strumenti matematici per quantificare l'importanza di un attributo nella determinazione dell'output (nei metodi "filter") sono molti. Vediamo due esempi, a seconda del tipo di attributi e di output che stiamo considerando. Nel seguito considereremo due colonne del dataset, di cui una tipicamente sarà l'output, e le chiameremo X e Y. Con lo stesso nome ci riferiremo sia alle colonne della tabella (quindi alle osservazioni), sia alle variabili casuali sottostanti, una relazione tra le quali intendiamo stimare.

7.1 Dati continui: il coefficiente di correlazione

Se X e Y sono continue, la loro *covarianza* può essere stimata come

$$\sigma_{X,Y} \sim \frac{1}{m} \sum_{i=1}^{m} (x_i - \bar{x})(y_i - \bar{y}), \tag{7.1}$$

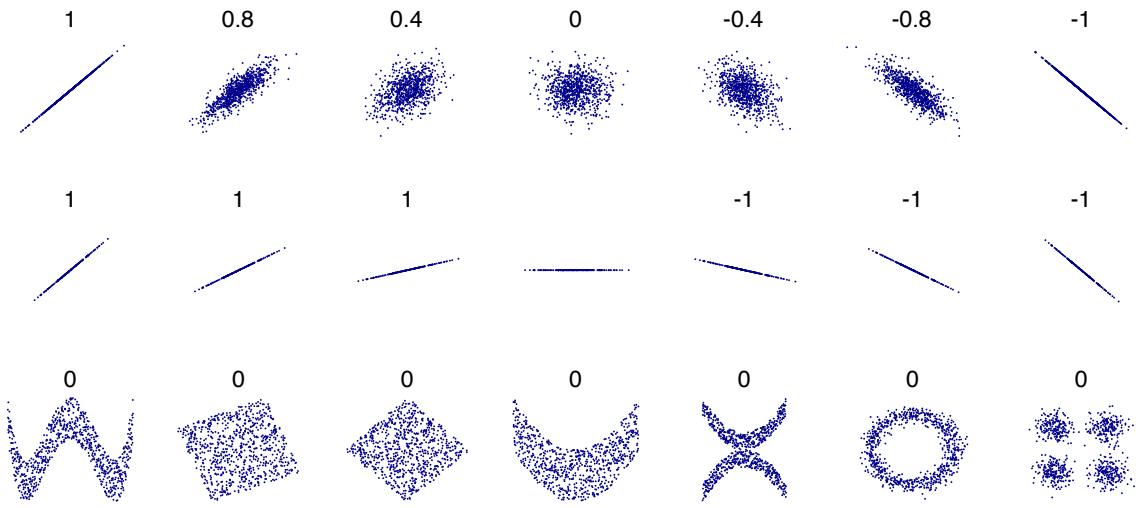


Figura 7.1: Esempi di indici di correlazione calcolati per insiemi di punti con diverse dispersioni, pendenze, relazioni. Si osservi come le relazioni non lineari nella terza riga non siano assolutamente rilevate (immagine presa da Wikipedia, stesso articolo).

dove $\bar{x}$ e $\bar{y}$ sono i valori medi delle osservazioni di X e Y rispettivamente. La covarianza tende ad essere positiva quando gli scarti delle x_i e delle y_i rispetto alla media tendono ad avere segno concorde, negativa se tendono ad essere discordi: cattura quindi possibili dipendenze lineari (affini) fra X e Y. La covarianza però dipende molto dalla magnitudine di X e Y, e quindi le covarianze fra variabili diverse non sono sempre confrontabili.

Per migliorare la confrontabilità, possiamo normalizzare la covarianza (7.1) rispetto alla variabilità delle due variabili casuali, misurata in base allo scarto quadratico medio, ottenendo il *coefficiente di correlazione* di Pearson[1]:

$$\rho_{X,Y} = \frac{\sigma_{X,Y}}{\sigma_X \sigma_Y}. \tag{7.2}$$

Il coefficiente varia tra -1 e 1. I valori estremi significano che i punti (x_i, y_i) sono perfettamente allineati. Un valore pari a 0 non significa che non vi siano dipendenze, ma solo che queste non sono lineari. La Figura 7.1 mostra alcuni esempi.

7.2 Variabili categoriche: l'informazione mutua

Abbiamo già visto il concetto di entropia $H(Y)$ di una variabile casuale Y. Data una seconda variabile casuale X, l'*entropia condizionata* $H(Y|X)$ risponde alla domanda "Quanti bit servono mediamente per comunicare l'esito di Y se si è già a conoscenza dell'esito di X?" Ovviamente, se X e Y non sono del tutto indipendenti, allora l'osservazione di X conterrà qualche informazione sull'esito di Y, quindi in generale

$$H(Y|X) \leq H(Y),$$

con l'uguaglianza verificata quando X e Y sono indipendenti.

Il calcolo dell'entropia condizionata è una generalizzazione di (6.4) al caso in cui X può avere più di due casi distinti. L'entropia condizionata è il valore atteso dell'entropia di Y al variare di X fra i suoi possibili valori:

$$H(Y|X) = \sum_{x \in X} \Pr(X = x) H(Y|X = x). \tag{7.3}$$

[1] `https://en.wikipedia.org/wiki/Pearson_product-moment_correlation_coefficient`

Infine, l'*informazione mutua* $I(X;Y)$ di X e Y è la diminuzione dell'entropia di Y data dalla conoscenza di X:

$$I(X;Y) = H(Y) - H(Y|X). \tag{7.4}$$

È possibile verificare che si tratta di una grandezza simmetrica in X e Y, ed è possibile calcolarla con la seguente formula:

$$I(X;Y) = \sum_{x \in X} \sum_{y \in Y} p(x,y) \log_2 \frac{p(x,y)}{p(x)p(y)}, \tag{7.5}$$

dove x e y variano rispettivamente nel dominio delle variabili casuali X e Y, $p(x,y) = \Pr(X = x \wedge Y = y)$ è la probabilità congiunta, mentre $p(x)$ e $p(y)$ sono le probabilità marginali.

Dalla definizione stessa, è chiaro che l'informazione mutua varia fra 0 e l'entropia di Y[2]. Alcuni casi particolari:

- $I(X;Y) = 0$ significa che $H(Y) = H(Y|X)$, ossia che la conoscenza della sola X è ininfluente sulla determinazione di Y;
- $I(X;Y) = H(Y)$ significa che $H(Y|X) = 0$, quindi l'osservazione di X determina esattamente il valore di Y, e non c'è bisogno di informazioni aggiuntive per comunicarne il valore.

In generale, più alto è il valore di $I(X;Y)$ più possiamo concludere che la conoscenza di X è importante per determinare il valore di Y.

7.2.1 Stima dell'informazione mutua in presenza di variabili continue

Possiamo trasformare una variabile X continua in un'equivalente variabile categorica X' a ℓ classi impostando un numero finito $\ell - 1$ di *soglie* $\theta_1 < \theta_2 < \cdots < \theta_{\ell-1}$, e trasformando un valore continuo $x \in X$ in un valore discreto $x' \in \{1, \ldots, \ell\}$ nel seguente modo:

$$x' = \begin{cases} 1 & \text{se } x < \theta_1 \\ 2 & \text{se } \theta_1 \leq x < \theta_2 \\ \vdots & \\ \ell - 1 & \text{se } \theta_{\ell-2} \leq x < \theta_{\ell-1} \\ \ell & \text{se } x \geq \theta_{\ell-1}. \end{cases} \tag{7.6}$$

Questo procedimento si chiama *discretizzazione* o *quantizzazione*. Nel caso in cui $\ell = 2$ (quindi con una soglia) si parla anche di *binarizzazione*. La determinazione del numero di soglie e dei loro valori può avvenire in diversi modi:

- impostazione a priori: ad esempio, spesso nei questionari la trasformazione dell'età avviene per fasce predeterminate (minore di 18, da 19 a 25, ecc.);
- suddivisione dell'intervallo di variabilità in parti uguali: si prende l'intervallo $[\min X, \max X]$ e lo si suddivide in ℓ parti uguali:

$$\theta_i = \min X + i \cdot \frac{\max X - \min X}{\ell}. \tag{7.7}$$

- Suddivisione per quantili: si individuano le soglie in modo che le ℓ classi siano ugualmente rappresentate; ad esempio, se $\ell = 2$ si sceglierà la mediana di X; se $\ell = 4$ si utilizzeranno il 25°, il 50° e il 75° percentile.

La suddivisione in parti uguali dell'intervallo di variabilità, per quanto semplice da realizzare, può essere pericolosa nel caso di distribuzioni molto asimmetriche o concentrate: si pensi al caso in cui una delle x sia molto più grande delle altre: quasi tutta la distribuzione potrebbe cadere nel primo intervallo. La determinazione dei quantili è invece più onerosa (richiede un ordinamento), ma estremamente robusta rispetto allo stesso problema.

[2] In realtà, dato che $I(X;Y) = I(Y;X)$, il massimo valore che l'informazione mutua può assumere è il minimo fra $H(Y)$ e $H(X)$.

7.2.2 Qualche tranello

Non sempre l'uso dell'informazione mutua tra singoli attributi e l'output è sufficiente a identificare gli attributi più importanti. Consideriamo il seguente esempio (tavola di verità dell'OR esclusivo):

X_1	X_2	Y
0	0	0
0	1	1
1	0	1
1	1	0

Osserviamo che Y è uniformemente distribuita fra i due valori, quindi la sua entropia, in bit, è $H(Y) = 1$. Anche le distribuzioni parziali di Y per $X_1 = 0$ e per $X_1 = 1$ sono uniformi. Segue che $H(Y|X_1) = 1$, quindi l'informazione mutua fra Y e X_1 è nulla: $I(X_1; Y) = 0$. Per simmetria, lo stesso risultato vale per X_2. Il risultato ci dice che la conoscenza della sola X_1, o della sola X_2, non fornisce alcuna informazione sul valore di Y. Nonostante ciò, l'informazione *combinata* di X_1 e X_2 determina perfettamente Y; quindi, non sempre ha senso escludere gli attributi che, in isolamento, sembrano poco correlate con l'output.

Per contro, supponiamo che la seguente tabella serva a determinare se un esemplare di una data specie ittica possa essere venduto o no, in base alla lunghezza:

Lunghezza (centimetri)	Lunghezza (pollici)	Vendibile (sì/no)
6,42	2,53	sì
3,21	1,26	no
4,94	1,94	sì
⋮	⋮	⋮
2,23	0,88	no

Se un'analisi dell'informazione mutua identifica una forte dipendenza dell'output "vendibile" dalle due colonne "lunghezza", ha senso utilizzarle entrambe? Molto probabilmente no: una volta nota una delle due colonne "lunghezza", l'altra non apporta nessuna informazione aggiuntiva, e probabilmente andrà scartata.

7.3 Un algoritmo greedy per la selezione degli attributi

Il seguente algoritmo mantiene un insieme di attributi selezionati, inizialmente vuoto. Ad ogni iterazione, valuta l'informazione mutua fra la tupla composta da tutti gli attributi selezionati fino a quel momento più un attributo candidato X_j e l'uscita Y.

1. `selected` $\leftarrow \{\}$;
2. `available` $\leftarrow$ {attributi del dataset};
3. Ripetere finché $|$`selected`$|$ < numero desiderato di attributi:
   (a) $j^* \leftarrow \underset{j \in \texttt{available}}{\operatorname{argmax}} I(X_{\texttt{selected}}, X_j; Y)$;
   (b) Aggiungere j^* a `selected`;
   (c) Levare j^* da `available`.

Capitolo 8

Cenni di apprendimento non supervisionato

Il contesto analizzato finora prevede la descrizione delle osservazioni tramite un vettore di attributi, la definizione di un "output" (numerico o categorico) e si richiede al sistema di generare un "modello" (funzione) che mappi ogni vettore di attributi su un valore di output, minimizzando l'errore atteso su esempi nuovi.

In alcuni casi l'output non è disponibile; si pensi al dataset `iris` senza la colonna che determina la varietà del fiore. Un problema alternativo a quello considerato finora potrebbe essere proprio quello di identificare, pur non conoscendo a priori la classificazione, gruppi di fiori simili, stabilendo quindi dei raggruppamenti senza conoscerli a priori. Questo è l'ambito del *clustering*[1], o *apprendimento non supervisionato* (unsupervised learning).

In questo genere di problemi, abbiamo a disposizione:

- una lista di oggetti
- un vettore di attributi per ciascun oggetto, *oppure* una matrice che assegna a ciascuna coppia di oggetti una distanza o una somiglianza.

Sulla base di queste sole informazioni, si richiede di formulare una partizione degli oggetti in gruppi onogenei, detti *cluster*.

8.1 Distanze e somiglianze

Se gli attributi sono numerici, $\boldsymbol{x}_i \in \mathbb{R}^n$, allora possiamo calcolare la *distanza euclidea* fra due oggetti $\boldsymbol{x}_i$ e $\boldsymbol{x}_j$:

$$\text{dist}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2 = \sqrt{\sum_{k=1}^{n} (x_{ik} - x_{jk})^2}; \tag{8.1}$$

allo stesso modo, possiamo definire distanze basate su altre norme (in particolare la norma 1 e la norma infinito).

In altri contesti è vantaggioso definire una misura di *somiglianza* o *similarità* fra oggetti. In tal caso, è possibile utilizzare la cosiddetta *cosine similarity*[2] come il prodotto scalare dei due vettori normalizzato rispetto alle loro lunghezze:

$$\text{sim}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \frac{\boldsymbol{x}_i \cdot \boldsymbol{x}_j}{\|\boldsymbol{x}_i\| \|\boldsymbol{x}_j\|}; \tag{8.2}$$

la cosine similarity è pari al coseno dell'angolo compreso fra i due vettori.

Alcune proprietà:

[1] Vedere `https://en.wikipedia.org/wiki/Cluster_analysis`, parte introduttiva

[2] `https://en.wikipedia.org/wiki/Cosine_similarity`

- la distanza, per definizione, è piccola per oggetti vicini (quindi simili), grande per oggetti lontani fra loro (quindi dissimili);
- al contrario, una misura di somiglianza è grande per oggetti simili (vicini) e piccola per oggetti diversi (quindi lontani);
- la cosine similarity varia da -1 (per oggetti rappresentati da vettori con verso opposto) a 1 (per vettori allineati e concordi).

8.1.1 Dati non numerici

Se vogliamo calcolare distanze o similarità fra oggetti non rappresentati da vettori numerici, abbiamo varie scelte, ad esempio convertendo ciascun attributo non numerico in un corrispondente insieme di attributi numerici utilizzando la rappresentazione unaria già vista in 5.1.1.

Documenti testuali

Un documento di testo può essere rappresentato da un lungo vettore di attributi in cui ogni attributo indica il numero di apparizioni un dato termine all'interno del testo.

In alternativa, un documento di testo può essere visto come un *insieme* di parole. In tal caso, per misurare la somiglianza di due documenti si può utilizzare il cosiddetto *coefficiente di Jaccard*[3]. Dati due insiemi A e B, il loro coefficiente di Jaccard è dato dal numero di elementi in comune, normalizzato rispetto al numero di elementi dei due insiemi:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}. \tag{8.3}$$

Il coefficiente di Jaccard varia tra 0 e 1; è nullo soltanto se $A \cap B = \emptyset$, vale 1 soltanto quando $A = B$.

8.2 Classificazione degli algoritmi di clustering

Esistono molte tecniche per il clustering, e si dividono in due tipologie:

- tecniche agglomerative, o bottom-up: si parte dagli individui che vengono a poco a poco raggruppati in cluster sempre più grossi;
- tecniche divisive, o top-down: si parte da un unico cluster contenente tutti gli individui e lo si divide ripetutamente in modo appropriato agli obiettivi.

8.3 Una tecnica bottom-up: clustering agglomerativo gerarchico

Supponiamo di avere un insieme di m oggetti, numerati da 1 a m, e di conoscere una matrice di distanze mutue $D = (d_{ij})$, con $i, j = 1, \ldots, m$. La matrice D è ovviamente simmetrica e contiene zeri sulla diagonale (ogni oggetto ha distanza 0 da sé stesso).

L'idea dell'algoritmo di clustering agglomerativo gerarchico[4] è quella di partire da tutti gli elementi separati, considerando ciascun elemento come un cluster a sé stante, e iterativamente unire insieme i due cluster più vicini, fino a ridurre l'insieme a un unico cluster.

[3] `https://en.wikipedia.org/wiki/Jaccard_index`
[4] `https://en.wikipedia.org/wiki/Hierarchical_clustering`

8.3.1 Distanze fra insiemi

Possiamo estendere la nozione di distanza fra elementi a distanza fra insiemi in vari modi. Tale estensione è detta *linkage*[5], e le varianti più comuni sono:

- *Single linkage* — la distanza fra due insiemi A e B è la minima distanza fra elementi nei due insiemi:
$$\text{dist}(A, B) = \min_{\substack{i \in A \\ j \in B}} d_{ij}. \tag{8.4}$$
Questa definizione corrisponde all'idea intuitiva di distanza come la "minima strada" necessaria per spostarsi fra due insiemi.

- *Complete linkage* — la distanza fra due insiemi A e B è la massima distanza fra elementi nei due insiemi:
$$\text{dist}(A, B) = \max_{\substack{i \in A \\ j \in B}} d_{ij}. \tag{8.5}$$
Questa definizione corrisponde all'idea di distanza come "caso peggiore" per spostarsi fra due insiemi.

- *Average linkage* — la distanza fra due insiemi A e B è la distanza media fra gli elementi nei due insiemi:
$$\text{dist}(A, B) = \frac{1}{|A||B|} \sum_{\substack{i \in A \\ j \in B}} d_{ij}. \tag{8.6}$$
Questa definizione corrisponde al "caso medio".

Si osservi che tutte queste definizioni si riducono alla distanza già definita nella matrice nel caso in cui A e B siano composti da un solo elemento.

8.3.2 L'algoritmo

1. $C \leftarrow \{1, 2, \ldots, m\}$
2. Finché $|C| > 1$:
   (a) Trova i due elementi $c_1, c_2 \in C$ per i quali $\text{dist}(c_1, c_2)$ è minima;
   (b) Registra i due elementi c_1, c_2 e la loro distanza $\text{dist}(c_1, c_2)$;
   (c) Leva i due elementi c_1 e c_2 da C, sostituendoli con la loro unione $c_1 \cup c_2$.

La registrazione delle operazioni eseguite permette di ricostruire una struttura gerarchica detta *dendrogramma*. Vediamo un semplice esempio di clustering agglomerativo in Fig. 8.1, con a fianco il dendrogramma risultante.

8.3.3 Similitudini e distanze

Così come presentato, l'algoritmo si basa sulla ricerca sistematica dei cluster di distanza minima. Nel caso in cui abbiamo a disposizione una matrice di *similitudini*, invece delle distanze, possiamo operare in due modi:

- Trasformare le similitudini in distanze utilizzando una funzione decrescente. Ad esempio, la cosine similarity e il coefficiente di Jaccard possono essere entrambi trasformati in distanze sottraendoli da 1.
- Oppure, possiamo riformulare l'algoritmo e le funzioni di linkage invertendo sistematicamente tutti i minimi e i massimi. Negli esercizi seguiamo questo secondo approccio.

[5] `https://en.wikipedia.org/wiki/Hierarchical_clustering#Linkage_criteria`

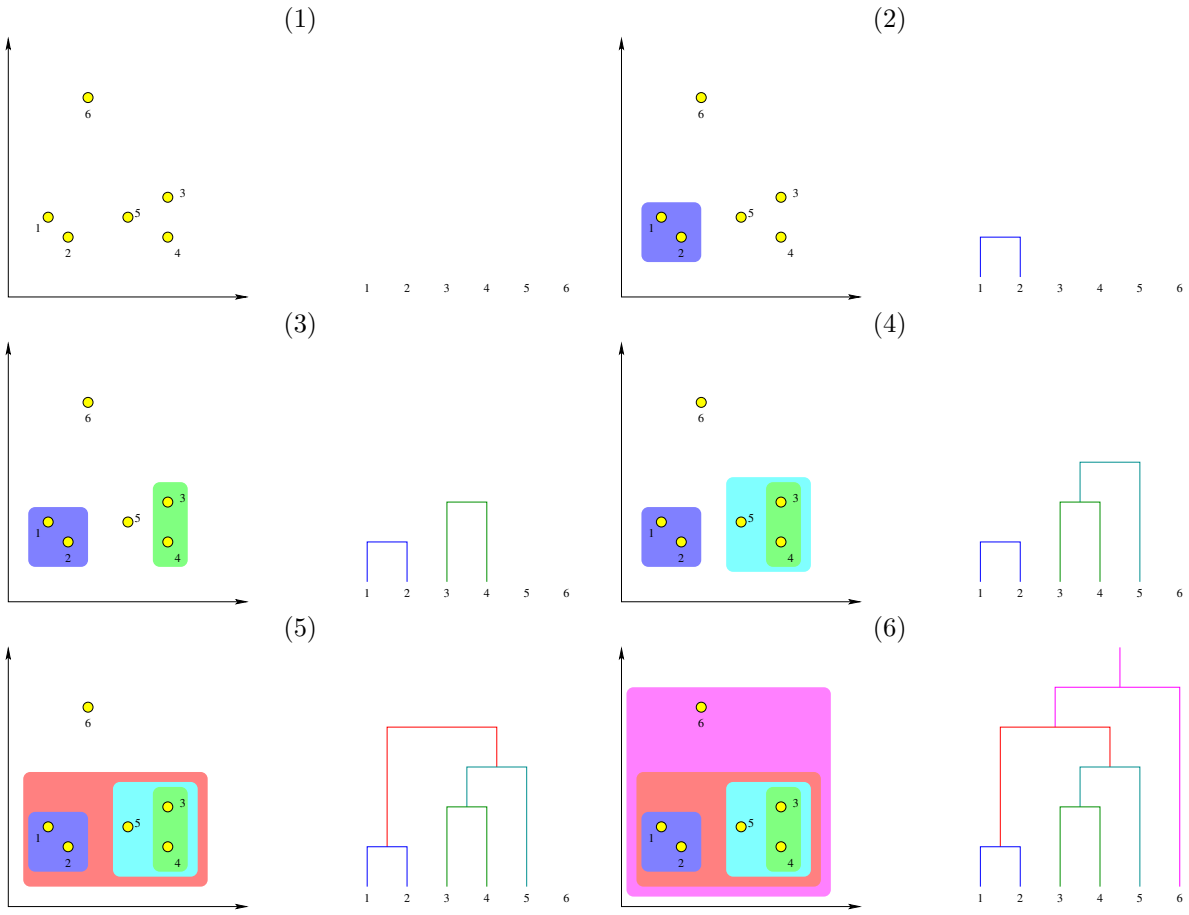


Figura 8.1: I passi del clustering agglomerativo gerarchico per un insieme di cinque punti nel piano.

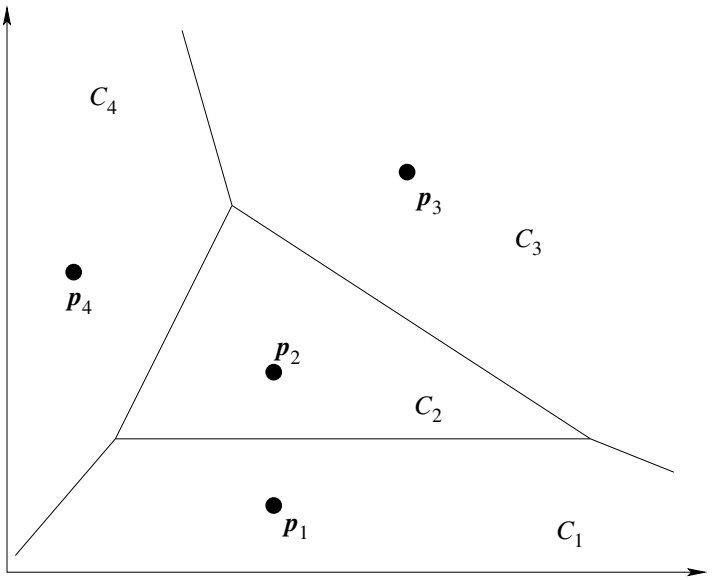


Figura 8.2: Suddivisione di Voronoi del piano $\mathbb{R}^2$ da parte di quattro vettori $\boldsymbol{x}1, \ldots, \boldsymbol{x}_4$.

8.3.4 Rappresentazione interna ed esterna

Si osservi che nell'apprendimento supervisionato di cui abbiamo parlato per buona parte del corso ogni "individuo" del nostro dataset è rappresentato da una collezione (solitamente un vettore) di attributi. Si parla di "rappresentazione interna" dei dati: la rappresentazione di ogni individuo è soggettiva e indipendente dagli altri individui (una persone è alta 171cm indipendentemente dalle altre persone).

L'algoritmo di clustering appena visto, però, prescinde completamente dagli attributi, e utilizza una tabella di relazioni (similarità, distanze) tra individui. Si parla in questo caso di "rappresentazione esterna" dei dati: non è nemmeno necessario conoscere glli attributi misurati.

È evidente che il passaggio da una rappresentazione interna a una esterna è sempre possibile: dato un vettore di attributi, abbbiamo visto molte tecniche per passare a una tabella di distanze o di similarità. Il viceversa è più complesso e, sebbene fattibile, non sarà oggetto di questo corso.

La prossima sezione si occupa di un altro algoritmo di clustering che, a differenza del precedente, richiede una rappresentazione interna dei dati.

8.4 Una tecnica top-down: il clustering K-means

Supponiamo di avere K vettori a n componenti reali: $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_K \in \mathbb{R}^n$. Questi K vettori, interpretati come coordinate di punti in uno spazio n-dimensionale, lo partizionano in K sottoinsiemi $C_1, \ldots, C_K$, dove C_i è il luogo dei punti di R^n che sono più vicini ad $\boldsymbol{p}_i$ che ad ogni altro punto $\boldsymbol{p}_j$[6]:

$$\begin{aligned} C_i &= \left\{ \boldsymbol{x} \in \mathbb{R}^n : i = \arg\min_{j=1,\ldots,K} \|\boldsymbol{x} - \boldsymbol{p}_j\| \right\} \\ &= \left\{ \boldsymbol{x} \in \mathbb{R}^n : \|\boldsymbol{x} - \boldsymbol{p}_i\| < \|\boldsymbol{x} - \boldsymbol{p}_j\| \quad \forall j = 1, \ldots, K, \quad j \neq i \right\} \end{aligned}$$

In Figura 8.2 si può vedere un esempio con $K = 4$ punti nel piano a $n = 2$ dimensioni. I domini $C_1, \ldots, C_4$ sono poligonali, e separati da segmenti o semirette posti sugli assi tra le varie coppie di punti.

Nel caso del clustering top-down, assumiamo che i nostri m campioni siano rappresentati dai consueti vettori di n attributi $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m \in \mathbb{R}^n$. il principio è quello di cercare (una volta fissato K) K

[6]Si noti come la ripartizione trascuri i punti del piano equidistanti da due o più elementi $\boldsymbol{p}_i$. Trattandosi di un insieme di misura nulla, l'attribuzione di quei punti a uno o all'altro dominio non ha rilevanza.

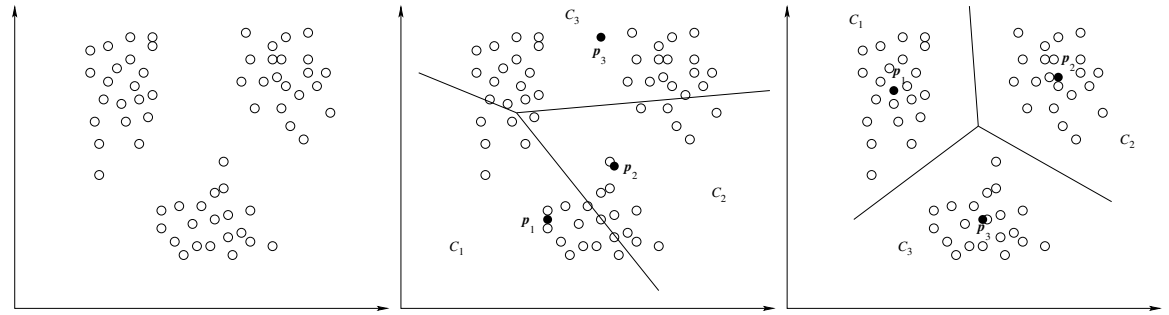


Figura 8.3: Considerato il dataset in $\mathbb{R}^2$ rappresentato a sinistra (punti vuoti), è migliore l'insieme di $K = 3$ prototipi (punti pieni) nella figura di centro, o in quella a destra?

punti $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_K$ che inducono una suddivisione dello spazio degli attributi $\mathbb{R}^n$ che sia "compatibile" con la struttura a cluster del nostro dataset. I K punti che andremo a collocare saranno detti, per ragioni che dovrebbero chiarirsi di qui a poco, "prototipi" del nostro dataset.

Osserviamo il dataset $D \subset \mathbb{R}^2$ rappresentato nella Figura **??** a sinistra. È evidentemente costituito da 3 gruppi separati di punti. Se piazziamo $K = 3$ prototipi a caso, è improbabile che la suddivisione ricalchi la struttura del nostro dataset. Se invece disponiamo i punti vicino ai "centri" dei nostri cluster, la suddivisione sarà molto più descrittiva del dataset.

Per valutare la rappresentatività del nostro insieme di prototipi, consideriamo la distanza (quadratica) media di ciascun punto del nostro dataset D dal prototipo ad esso più vicino:

$$E(\boldsymbol{p}_1, \ldots, \boldsymbol{p}_K) = \sqrt{\frac{1}{m} \sum_{i}^{K} \sum_{\boldsymbol{x} \in D \cap C_i} \|\boldsymbol{x} - \boldsymbol{p}_i\|^2}. \tag{8.7}$$

Questa grandezza può essere interpretata come l'errore che si commetterebbe se si sostituisse ogni individuo del dataset con una copia del prototipo ad esso più vicino. È evidente che l'errore (8.7) può essere ridotto fissando il corretto numero K di proptotipi e piazzandone ciascuno nel "mezzo" di una diversa nuvola di punti.

8.4.1 L'algoritmo "hard" K-means

Dato il dataset $D \subset \mathbb{R}^n$, $|D| = m$, fissato $K > 0$:

1. Scegliere "casualmente" K prototipi $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_K \in \mathbb{R}^n$.
   Questi prototipi inducono una partizione di Voronoi $C_1, \ldots, C_K$ dello spazio degli attributi.

2. Finché non si è soddisfatti:

   (a) Ridefinire i prototipi come centroidi della partizione indotta sul dataset:

   $$\boldsymbol{p}_i = \frac{1}{|D \cap C_i|} \sum_{\boldsymbol{x} \in D \cap C_i} \boldsymbol{x};$$

   (b) Questi nuovi prototipi ridefiniscono, a loro volta, una nuova partizione di Voronoi.

Sotto opportune condizioni, questo ciclo "converge" nel senso che i prototipi si stabilizzano, quindi spesso la condizione di teerminazione del ciclo è la stabilità dei prototipi da un'iterazione alla successiva.

Sono possibili altre regole per aggiornare i prototipi da un'iterazione alla successiva, ad esempio imponendo variazioni incrementali o basate su suddivisioni più "sfumate" dello spazio rispetto alla partizione di Voronoi (algoritmi K-means "soft" o "fuzzy").

Parte II

Esercitazioni di laboratorio

Esercitazione 1

Lettura e utilizzo di un semplice dataset

1.1 Scopo dell'esercitazione

- Scaricare un dataset da un archivio internet e importarlo in uno script Python;
- creare un modello utilizzando funzioni di libreria;
- generare semplici rappresentazioni grafiche del dataset;
- scrivere un primo, semplice, algoritmi di machine learning.

La creazione del modello avviene semplicemente tramite la chiamata a opportune funzioni, utilizzate a "scatola chiusa", senza conoscerne il funzionamento interno.

1.2 Script

Il dataset utilizzato in quest'esercitazione è `iris.data`:

`https://archive.ics.uci.edu/ml/datasets/Iris`.

Lo script completo e commentato è accessibile dalla pagina web del corso:

`https://disi.unitn.it/~brunato/AA/iris.py`.

1.3 Osservazioni

Lo script utilizza la libreria `pandas` per leggere e trattare il dataset, la libreria `scikit-learn` per creare e utilizzare il modello, e la libreria `matplotlib` per visualizzare i grafici.

Il modello di previsione (una support vector machine) viene addestrato sull'intero dataset e valutato su alcuni vettori di esempio. Notare che l'addestramento e l'utilizzo del modello avvengono completamente alla cieca: nelle prossime lezioni impareremo alcune metodologie per valutare la bontà di un modello.

Dopo aver creato e provato il modello, lo script disegna un grafico che rappresenta la distribuzione di due coordinate del dataset in base alla varietà. Due possibili grafici sono raffigurati in Figura 1.1. Osservare come il problema di classificazione sia, tutto sommato, molto semplice: i dati sono ben distinti, in particolare la varietà *Iris setosa* è ben separata dalle altre due. L'uso simultaneo delle quattro coordinate dovrebbe essere sufficiente a distinguere anche le altre due varietà.

Infine, lo script contiene una funzione che, dato un dataset (X, y) e un vettore di misure incognite $\boldsymbol{x}$, cerca l'elemento di X più vicino a $\boldsymbol{x}$ e ne restituisce la classe. Questo tipo di classificatore, detto *nearest neighbor*, sarà ulteriormente esaminato nelle prossime lezioni.

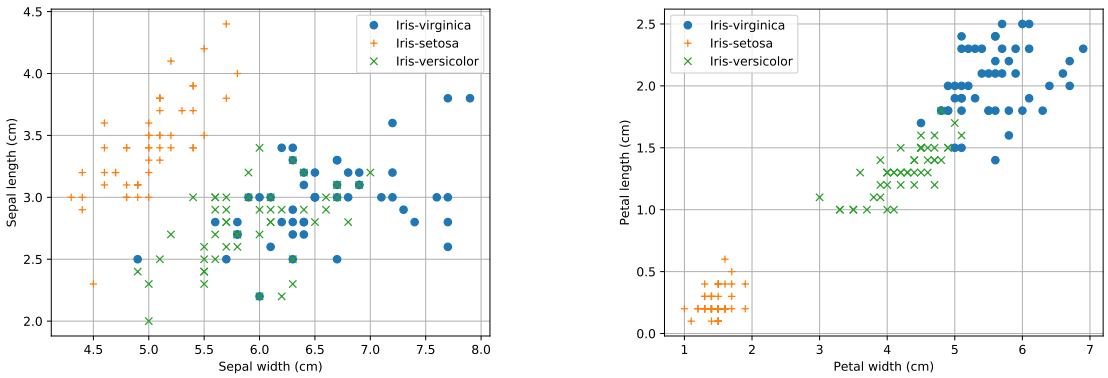


Figura 1.1: Diagramma a dispersione: distribuzione delle dimensioni dei sepali (sinistra) e dei petali (destra) in base alla varietà di iris.

Esercitazione 2

K-Nearest Neighbors e minimi quadrati a una dimensione

2.1 Scopo dell'esercitazione

- Realizzare due semplici algoritmi di machine learning, uno adatto a un problema di classificazione e uno per la regressione.

2.2 Script

Il dataset utilizzato in quest'esercitazione è `iris.data`:

`https://archive.ics.uci.edu/ml/datasets/Iris`.

Un pacchetto completo e commentato contenente l'implementazione dei due algoritmi è accessibile dalla pagina web del corso:

`https://disi.unitn.it/~brunato/AA/ml.py`.

Uno script che utilizzale funzioni definite nel pacchetto

`https://disi.unitn.it/~brunato/AA/iris2.py`.

2.3 Osservazioni

I diagrammi realizzati la scorsa volta (Fig. 1.1) mostrano come le dimensioni (larghezza e lunghezza) dei petali siano correlate. Per provare i minimi quadrati a una dimensione, dunque, abbiamo scelto quelle due colonne del dataset.

In Figura 2.1, vediamo il modello lineare sovrapposto ai dati. Osserviamo che il modello è penalizzato dall'assenza di un termine costante: la retta è obbligata a passare per l'origine, quindi non riesce a "passare in mezzo ai dati" nel modo migliore. Una generalizzazione del modello permetterà di ovviare al problema.

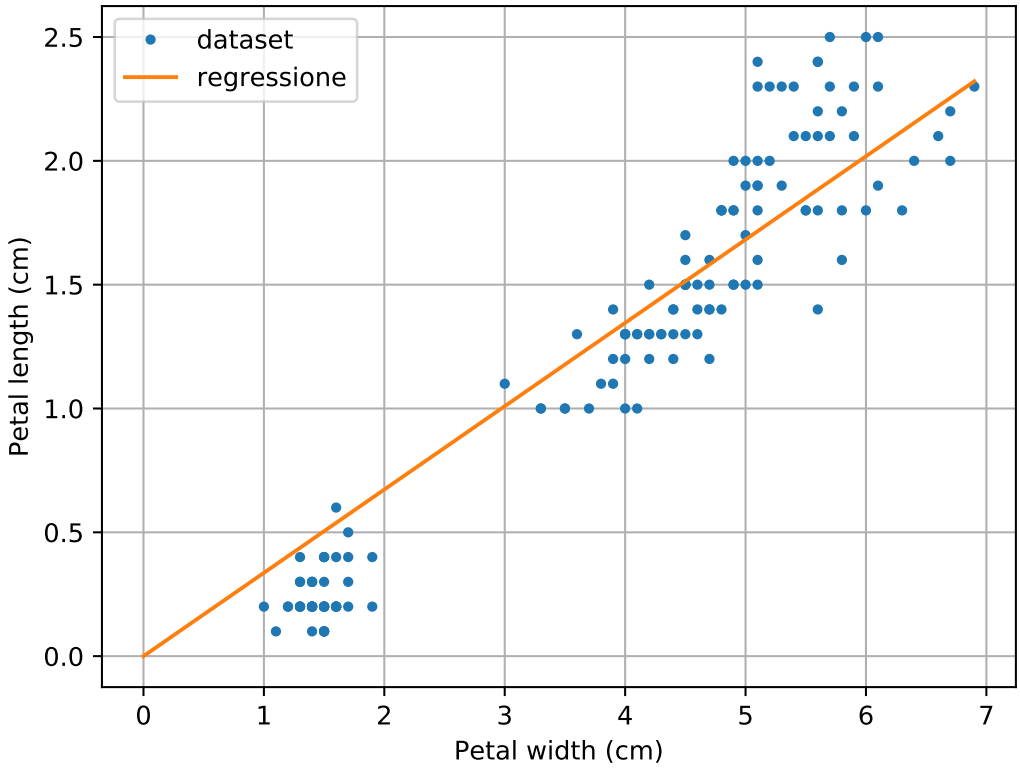


Figura 2.1: Diagramma a dispersione: distribuzione delle dimensioni dei petali (punti) e retta di regressione.

Esercitazione 3

Regressione polinomiale e suddivisione del dataset

3.1 Scopo dell'esercitazione

- Realizzare un modello lineare a più dimensioni in base al metodo dei minimi quadrati come descritto in 2.2.2.
- Utilizzare il modello per effettuare una regressione polinomiale come descritto in 2.2.3.
- Valutare la bontà del modello polinomiale su un dataset semplice.

3.2 Script

Il dataset utilizzato in quest'esercitazione è `iris.data`:

`https://archive.ics.uci.edu/ml/datasets/Iris`.

Un pacchetto completo e commentato contenente l'implementazione dei due algoritmi è accessibile dalla pagina web del corso:

`https://disi.unitn.it/~brunato/AA/ml.py`.

Uno script che utilizza le funzioni definite nel pacchetto

`https://disi.unitn.it/~brunato/AA/iris3.py`.

Lo script suddivide il dataset in due parti, una per l'addestramento del modello e una per la sua validazione.

3.3 Osservazioni

Si osservi che, anche se il modello polinomiale può avere un grado arbitrario al di sopra di un certo grado si verificano fenomeni di instabilità numerica, con l'RMSE di addestramento che cresce al crescere dell'esponente (il che non dovrebbe accadere se si lavorasse con precisione infinita). La matrice delle potenze, infatti, è molto mal condizionata (contiene valori che spaziano su molti ordini di grandezza).

In Figura 3.1 vediamo quattro casi. Il grado 0 dà luogo a una funzione costante che si attesta sulla media delle y di addestramento. Il grado 1, a differenza di quello visto nella precedente esercitazione, comprende anche un termine noto, quindi non è più vincolato a passare per l'origine. Il polinomio di grado 3 iniia già a manifestare alcune stranezze: la corrispondenza non è sempre crescente. Infine, il polinomio di grado 7 si trova evidentemente a inseguire alcune fluttuazioni statistiche del dataset che non hanno nulla a che vedere con la struttura della popolazione nel suo complesso.

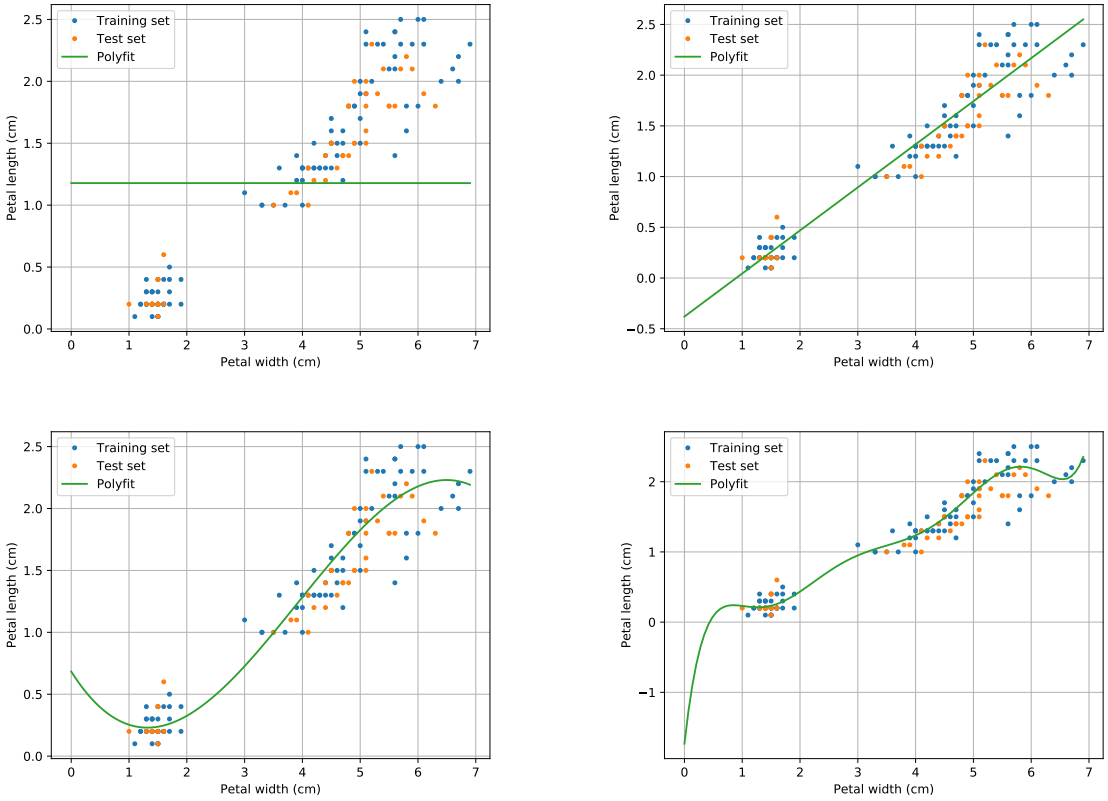


Figura 3.1: Diagramma a dispersione: distribuzione delle dimensioni dei petali (punti) e polinomi di regressione di grado 0, 1, 3 e 7. I polinomi sono determinati esclusivamente sul sottoinsieme di addestramento.

Eseguendo lo script per gradi crescenti, si può notare come, in generale, l'errore sull'insieme di addestramento tenda a scendere. L'errore sul sottoinsieme di validazione, invece, non solo smette presto di scendere, ma ha la tendenza a risalire al crescere del grado del polinomio. Questo perché il sistema, nel cercare di apprendere caratteristiche sempre più "minute" del dataset, perde la capacità di generalizzare.

Esercitazione 4

Regressione logistica

4.1 Scopo

Dato un dataset di classificazione, con due classi rappresentate dai valori 0 e 1, calcolare un modello logistico.

4.2 Procedimento

Utilizziamo il dataset *Blood Transfusion Service Center*[1], che riporta alcuni parametri relativi ai donatori di sangue (mesi dall'ultima donazione, mesi dalla prima donazione, numero di donazioni, quantità di sangue donato) e vi associa una variabile binaria che dice se il donatore si è presentato a donare nuovamente.

4.2.1 Suddivisione del dataset

Può essere interessante valutare le prestazioni del modello logit, man mano che questo procede, rispetto a un insieme di validazione. Per questo, separiamo il dataset complessivo in due parti, una per l'addestramento con circa il 75% dei dati e una per la validazione.

4.2.2 Normalizzazione

Osserviamo che le diverse colonne di ingresso hanno diversi ordini di grandezza: alcune hanno valori nel campo delle unità, altre nelle migliaia. Per evitare problemi di instabilità numerica, possiamo normalizzare le colonne del dataset in modo che i valori abbiano lo stesso intervallo di variabilità, $[0, 1]$. Ci limitiamo a riscalare i valori. Data la colonna $j = 1, \ldots, n$, sia

$$M_j = \max_{i=1,\ldots,m} x_{ij}.$$

La normalizzazione della matrice avviene con la seguente mappa lineare:

$$x_{ij} \mapsto \frac{x_{ij}}{M_j}.$$

4.2.3 Discesa lungo il gradiente

Come abbiamo visto, non è possibile trovare i valori ottimali dei coefficienti in forma chiusa. Utilizziamo dunque il metodo della discesa lungo il gradiente. Impostiamo un valore iniziale della learning rate η. Ad ogni passo, valutiamo sia l'RMSE sul training set, sia quello sul validation set.

[1] `https://archive.ics.uci.edu/ml/datasets/Blood+Transfusion+Service+Center`

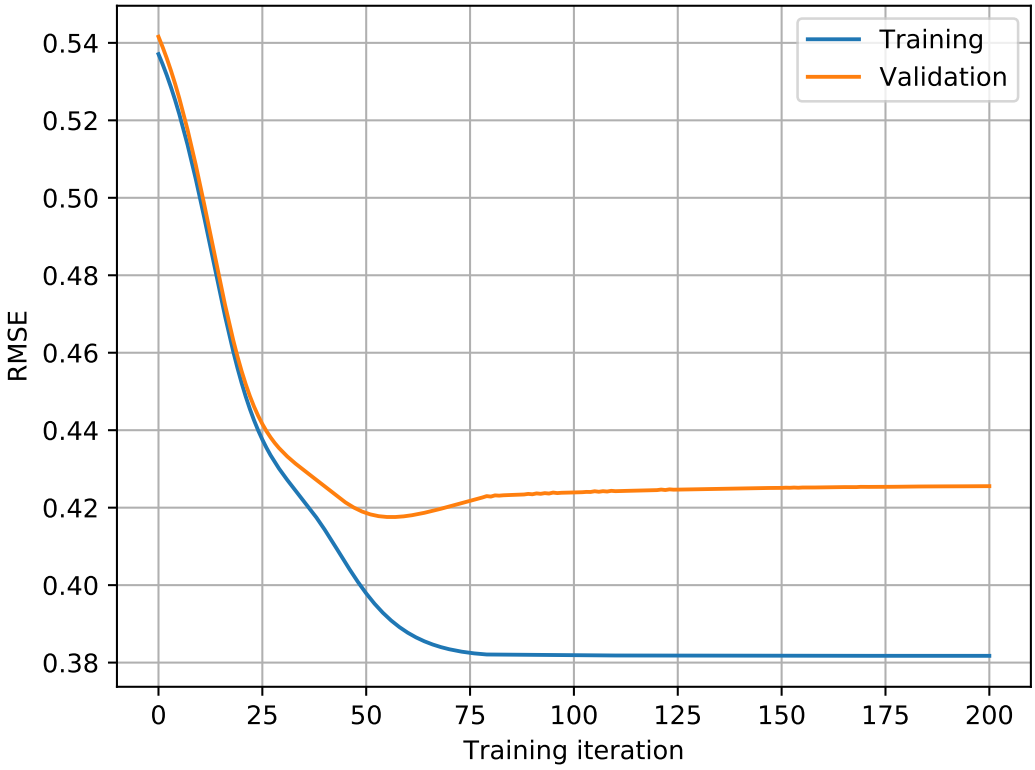


Figura 4.1: Andamento dell'RMSE sugli insiemi di addestramento e validazione.

Gradiente adattivo

Per migliorare le prestazioni, rendiamo la learning rate variabile a seconda dell'andamento della discesa. Ogni volta che uno spostamento con tasso η ha successo, ne aumentiamo il valore del 10%. Ogni volta che un passo troppo grande fa peggiorare la situazione, dimezziamo η.

4.2.4 Addestramento

La Figura 4.1 mostra un caso tipico di andamento dell'RMSE sull'insieme di addestramento e su quello di validazione. Si noti come l'andamento rispetto all'insieme di validazione non sia strettamente decrescente.

4.3 Script

Il codice dello script si trova alla pagina web del corso:

`https://disi.unitn.it/~brunato/AA/transfusion.py`.

Esercitazione 5

Alberi di decisione

5.1 Scopo

Dato un dataset di classificazione con feature continue, ci proponiamo di:

1. esplorare alcune funzionalità di una funzione di libreria per l'addestramento degli alberi di decisione;
2. implementare in Python un albero di decisione addestrato sulla base di un dataset;

5.2 Prima parte: utilizzo di una libreria esterna

In questo caso utilizziamo la libreria `scikit-learn` per addestrare un albero sul dataset `adult.data` di UCI. Si tratta di un dataset misto, con alcune colonne numeriche ed altre di tipo categorico.

5.2.1 Codice

Il codice consiste nella lettura del dataset e nella costruzione e addestramento dell'oggetto della libreria. È disponibile in `http://disi.unitn.it/~brunato/AA/adult.py`. Dopo la lettura dei dati, è necessario convertire ciascuna delle colonne categoriche in colonne numeriche utilizzando la consueta rappresentazione unaria (descritta in 5.1.1).

La rappresentazione dell'albero di profondità 3 viene salvata nel file `adult.dot`, dal quale si può generare un documento PDF tramite il comando

```
dot -Tpdf adult.dot -o adult.pdf
```

5.2.2 Risultati

Il primo test riguarda l'addestramento e la visualizzazione di un albero di decisione. Dato il grande numero di campioni e di colonne, l'albero di default, senza limiti in profondità o in purezza, risulta molto grande. Una volta limitato a tre livelli, otteniamo l'albero rappresentato in Fig. 5.1. Per chiarezza, abbiamo operato una sostituzione lessicale delle stringhe nella forma `X[`n`]` con i corrispondenti nomi di colonna.

Per verificare la bontà dell'albero addestrato, utilizziamo la tabella di validazione `adult.test`. Addestriamo l'albero sulla tabella di addestramento e con diversi valori del parametro di profondità massima; ogni volta ne verifichiamo l'accuratezza sulla tabella di validazione. Otteniamo il risultato mostrato in Figura 5.2.

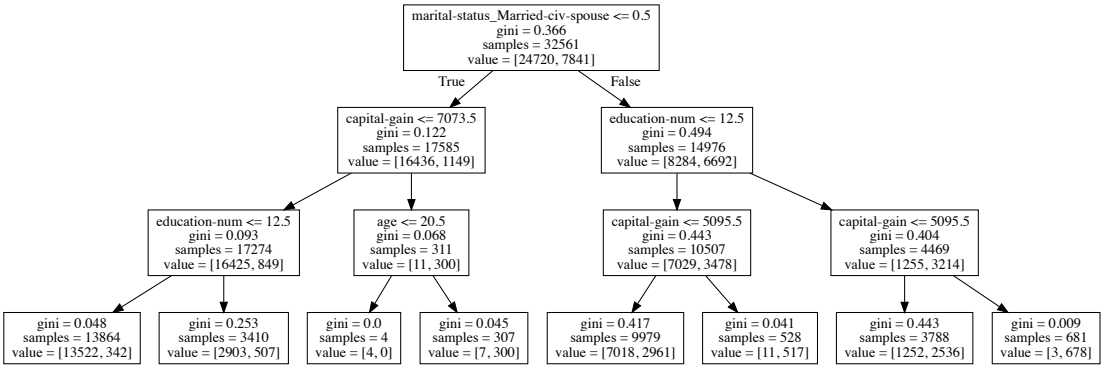


Figura 5.1: Albero di decisione generato da `scikit-learn` sul dataset `adult` con impurità di Gini e profondità massima pari a 3.

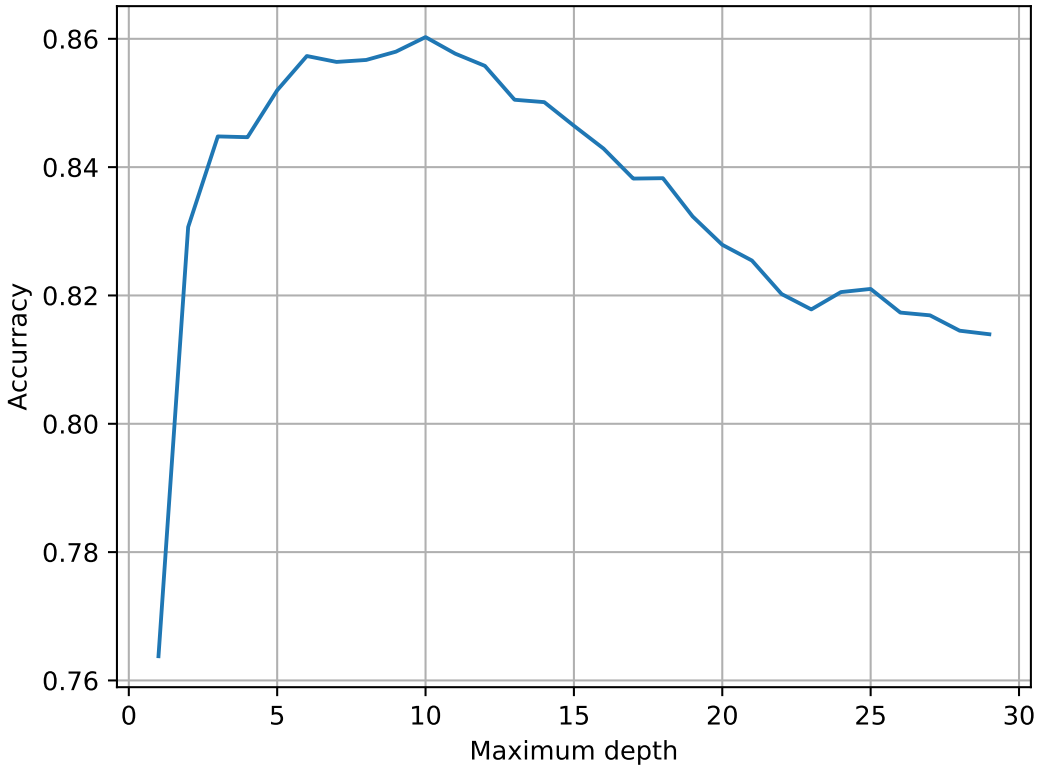


Figura 5.2: Accuratezza dell'albero di decisione generato da `scikit-learn` sul dataset `adult` con impurità di Gini e profondità variabile da 1 a 29.

```
{'column': 3,
'threshold': 1.3,
'yes': {'column': 2,
        'threshold': 1.6000000000000001,
        'yes': {'y': 'Iris-setosa'},
        'no': {...}}
'no': {'column': 2,
        'threshold': 5.0999999999999996,
        'yes': {...},
        'no': {'y': 'Iris-virginica'}}}
```

Figura 5.3: Rappresentazione dell'albero di decisione sul dataset `iris`, massimizzazione dell'information gain; sono rappresentati solo i primi due livelli.

```
Predizione di prova
X[3] <= 1.300000?
Yes
X[2] <= 1.600000?
Yes
Found: Iris-setosa
La predizione per il vettore di ingresso [5, 3.3, 1.4, 0.2] è Iris-setosa.
```

Figura 5.4: Discesa lungo l'albero di decisione per un vettore di esempio.

5.3 Seconda parte: implementazione Python

Utilizziamo Pandas per leggere il dataset `iris.data` di UCI; per prima cosa scriviamo una funzione per stimare l'entropia di una distribuzione sulla base di un vettore di sue manifestazioni, da applicare all'output del dataset.

Per i nostri scopi, un albero sarà rappresentato da una quadrupla:

$$\mathcal{A} = (|^*, \theta^*, \mathcal{A}_{\S|^* < \theta^*}, \mathcal{A}_{\S|^* \geq \theta}),$$

dove $\mathcal{A}_{\S|^* < \theta^*}$ e $\mathcal{A}_{\S|^* \geq \theta^*}$ sono rispettivamente i sottoalberi sinistro e destro; una foglia sarà invece rappresentata da un singolo valore di uscita, $\tilde{y}$, da utilizzare come predizione.

Entrambi i tipi di nodo sono realizzati in Python come dizionari.

5.3.1 Codice

Il codice è disponibile in `http://disi.unitn.it/~brunato/AA/tree.py`. La costruzione e la navigazione dell'albero sono ricorsive; la costruzione si basa su una funzione ausiliaria che, sulla base di un dataset, determina la suddivisione migliore dal punto di vista del guadagno informativo.

5.3.2 Risultati

L'albero, addestrato sulla base dell'intero dataset, è riportato in figura 5.3. I nodi foglia hanno il solo campo `y`; ogni nodo interno contiene quattro campi: `column` (indice cella colonna su cui decidere), `threshold` (valore di soglia), `yes` (sottoalbero per la risposta positiva), `no` (sottoalbero per la risposta negativa).

Si noti come già al secondo livello, con due sole domande, sia già possibile trovare dei nodi puri. Ad esempio, con il vettore di ingresso $\boldsymbol{x} = (5, 3.3, 1.4, 0.2)$ la procedura di decisione è mostrata in Figura 5.4.

Esercitazione 6

Curve di prestazione

6.1 Scopo

Dato un problema di classificazione a due classi, il modello logit produce un output continuo in $(0, 1)$. Resta il problema di determinare un opportuno valore di soglia $\theta \in [0, 1]$ da applicare alla funzione di decisione (2.11).

6.2 Procedimento

Lo script creato per l'esercitazione 4 determina, alla fine del processo iterativo di discesa lungo il gradiente, i valori ottimali dei coefficienti (pesi) $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_n)$.

Sulla base di questi, e degli esempi di validazione $\boldsymbol{x}_i^{\text{val}}$, possiamo determinare i corrispondenti valori del modello $\tilde{y}_i^{\text{val}} = \sigma(\boldsymbol{\beta} \cdot \boldsymbol{x}_i^{\text{val}})$. A seconda della soglia $\theta \in [0, 1]$, ciascun valore sarà poi interpretato come appartenente alla classe positiva (1) o negativa (0) sulla base della funzione (2.11).

In particolare, al variare di θ varierà la matrice di confusione, e quindi le misure di prestazione ad essa collegate: precisione, sensibilità, score $F_1 \ldots$

6.3 Script

Il codice dello script si trova alla pagina web del corso[1]. Una volta determinati i pesi $\boldsymbol{\beta}$ e i valori del modello logit per gli esempi di validazione, ripetiamo un ciclo al variare di $\theta \in [0, 1]$ a intervalli di 10^{-2}. A ciascuna iterazione, calcoliamo la matrice di confusione e le misure ad essa collegate, raccogliendole in alcune liste per poi rappresentarle in alcuni grafici.

In particolare, siamo interessati ai seguenti grafici:

- **Sensibilità, precisione, accuratezza contro** θ — sappiamo che la sensibilità e la precisione sono obiettivi spesso contrastanti, quindi una curva che riporta la relazione fra le due ci permette di individuare il valore di θ corrispondente a un compromesso fra le due.

- **Curva ROC** — consideriamo i casi estremi: per $\theta = 0$, classifichiamo tutti gli esempi come positivi. Questo significa che la nostra sensibilità vale 1, però vale 1 anche la "False Positive Rate", definita come
$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}}.$$
Quando invece $\theta = 1$, noi rigettiamo come negativi tutti gli esempi. Quindi la sensibilità crolla a zero, come pure la FPR appena definita.
Il grafico che rappresenta la sensibilità (che in questo contesto è detta "True Positive Rate") contro la FPR si chiama *curva ROC* ("Receiver Operating Characteristic", perché era utilizzata

[1] `http://disi.unitn.it/~brunato/AA/logit-theta.py`

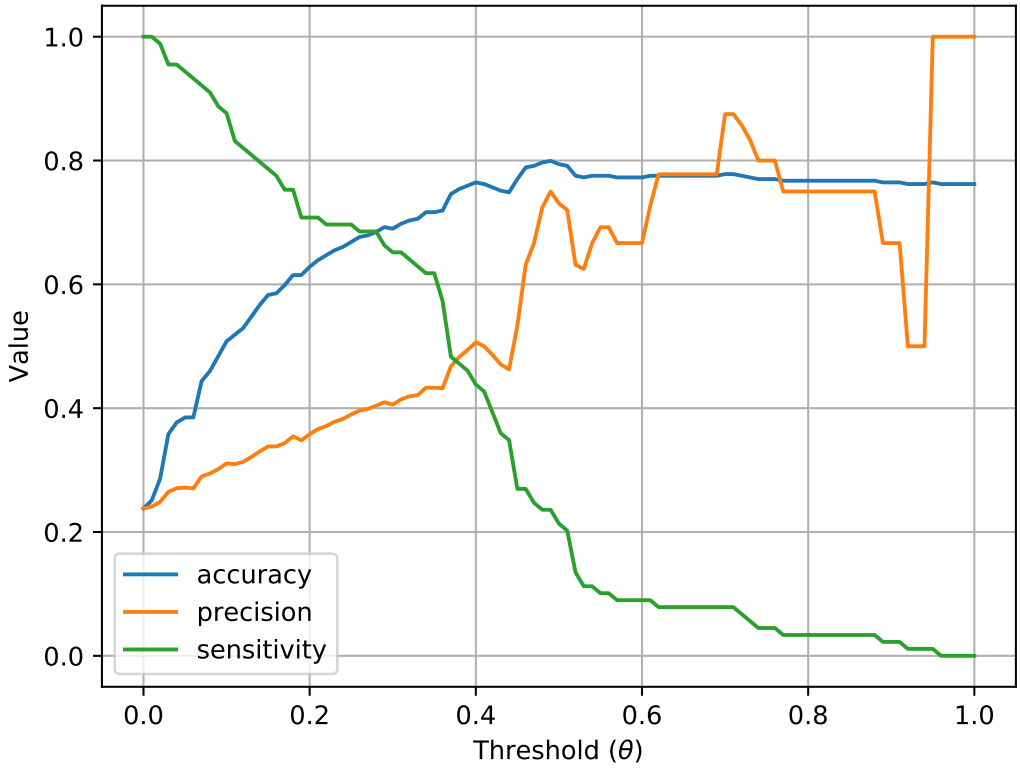


Figura 6.1: Precisione, sensibilità e accuratezza contro θ.

per analizzare segnali radar)[2] ed è molto utile per valutare la prestazione di un classificatore binario. In generale, più l'area sottesa dalla curva è alta, più il classificatore funziona bene per molti valori di θ.

6.4 Discussione

Lo script fornisce risultati simili a quelli rappresentati nelle Figure 6.1–6.2. Si osservi come, al di là del rumore statistico causato dalla bassa numerosità del dataset, la precisione e la sensibilità siamo effettivamente obiettivi in contrasto (anche se localmente il grafico di Fig. 6.1 può essere crescente).

Infine, la curva ROC di Fig. 6.2, la cui area sottesa è decisamente minore del massimo ottenibile (l'intero rettangolo 1×1), dimostra che il classificatore è intrinsecamente poco affidabile, indipendentemente dal valore di soglia scelto.

[2] `https://en.wikipedia.org/wiki/Receiver_operating_characteristic`

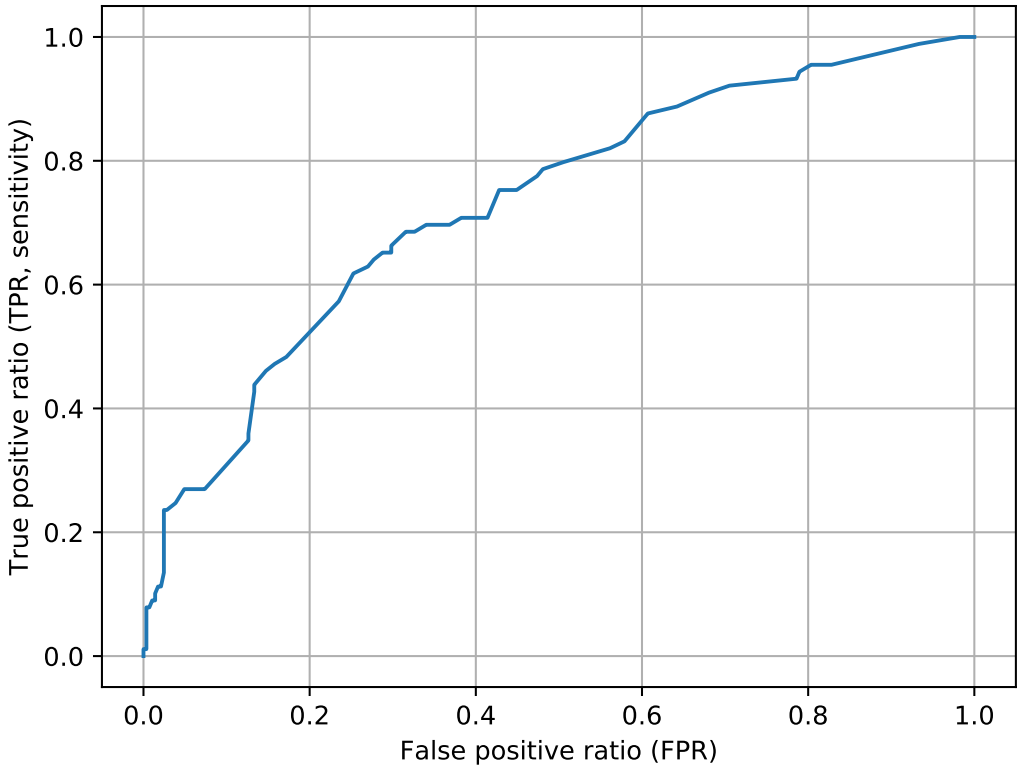


Figura 6.2: Curva ROC.

Esercitazione 7

Clustering

7.1 Scopo

In quest'esercitazione lavoreremo sul dataset `iris`, ma trascureremo l'informazione di categoria (l'ultima colonna, contenente la varietà del fiore) e rimescoleremo le righe per eliminare qualunque informazione esplicita sulla classificazione dei fiori, mantenendo solamente le misure dei petali e dei sepali.

Utilizzeremo una procedura di clustering gerarchico per valutare se le sole informazioni di misura permettono di ricostruire le tre classi in cui il dataset era originariamente organizzato.

7.2 Il codice

Il codice è disponibile in `http://disi.unitn.it/~brunato/AA/clustering.py`.

Leggiamo il file `iris.data` e rimescoliamone le righe. Normalizziamo le colonne dividendo ciascuna di esse per il suo massimo (in questo caso il valore 0 è significativo, quindi lo teniamo invariato).

In seguito, calcoliamo la matrice delle distanze mutue. Passiamo tale matrice a una procedura di clustering gerarchico. Il pacchetto che useremo si chiama `scipy.cluster.hierarchy`, e permette di effettuare il clustering gerarchico sulla base di vari criteri di collegamento. L'operazione di clustering restituisce una lista delle operazioni di aggregazione operate in successione, e il pacchetto offre una funzione che permette di tracciare un dendrogramma.

Infine, utilizziamo l'ordine di rappresentazione delle foglie nel dendrogramma per riordinare i dati (che avevamo inizialmente permutato) in modo da visualizzare un eventuale ordine "scoperto" dalla procedura di clustering.

7.3 Risultati

Una volta calcolata la matrice delle distanze, possiamo rappresentarla graficamente tramite una "mappa di calore" (heatmap), ovvero una matrice di pixel il cui colore codifica (in una scala che va tipicamente dal blu al rosso) i valori contenuti in una matrice numerica. Nel nostro caso, la matrice delle distanze è rappresentata in Fig. 7.1. Si noti come, al di là dell'ovvia simmetria, le distanze risultino casualmente distribuite.

Nonostante la casualità dell'ordine e il fatto di aver trascurato le informazioni di classe, la procedura di clustering produce un dendrogramma che evidenzia chiaramente la presenza di tre cluster distinti, di cui uno particolarmente diverso dagli altri due.

La rappresentazione del dendrogramma, oltre ad essere utile a verificare la struttura dei cluster a colpo d'occhio, permette anche di riordinare gli elementi in modo da tenere vicino elementi simili. Infatti, il dendrogramma è tracciato in modo che i suoi rami non si accavallino, e l'ordine delle foglie tiene vicini elementi che si trovano in cluster più compatti. Riordiniamo dunque la matrice delle distanze di Fig. 7.1 utilizzando l'ordine delle foglie del dendrogramma con complete linkage di Fig. 7.2;

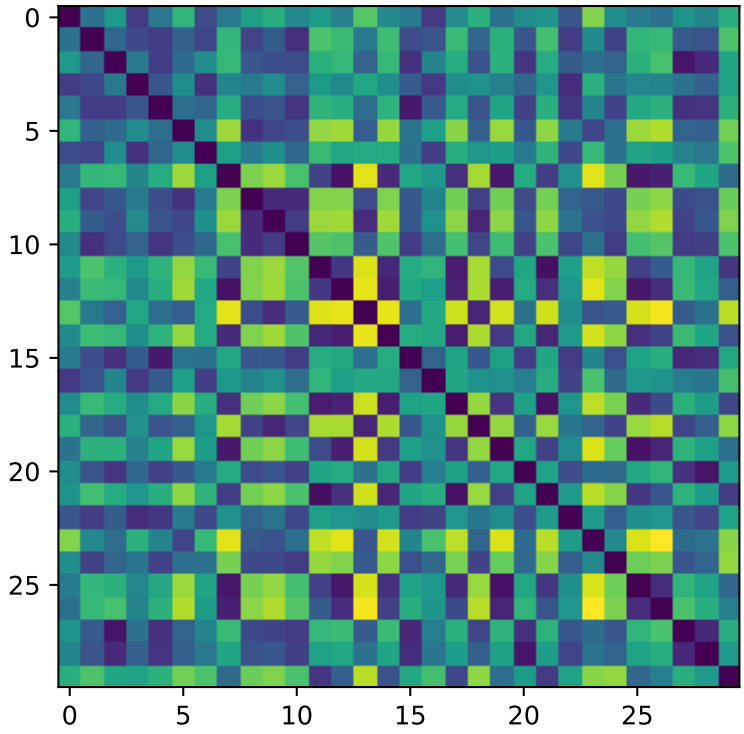


Figura 7.1: Matrice delle distanze fra gli elementi di `iris.data`, righe rimescolate casualmente, colonne normalizzate con massimo a 1.

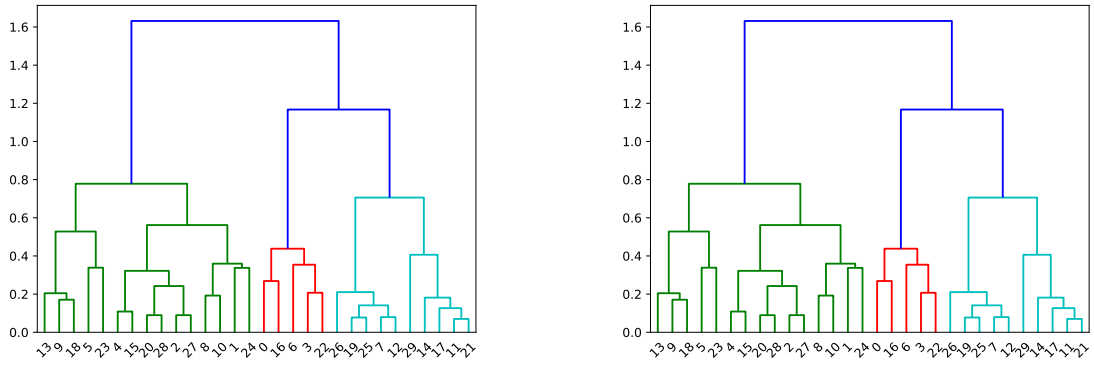


Figura 7.2: Dendrogrammi generati dalla procedura di clustering gerarchico scritta in Python (sinistra) e dalla libreria di Scikit-Learn (destra).

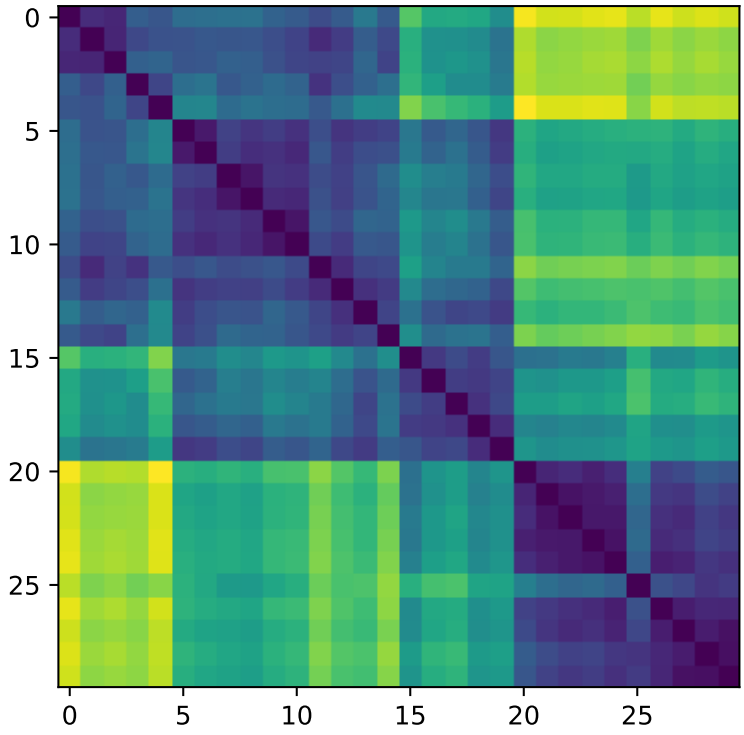


Figura 7.3: La matrice delle distanze di Fig. 7.1, con le righe e le colonne permutate nell'ordine delle foglie del dendrogramma di Fig. 7.2.

la matrice permutata (sia nell'ordine delle righe che delle colonne, ovviamente) mostra chiaramente una struttura a blocchi con tre gruppi di ugual dimensione sulla diagonale (Fig. 7.3).

Possiamo concludere che, nonostante tutte le informazioni di classificazione presenti nel file siano state eliminate (rimescolamento) o trascurate (ultima colonna), la procedura di clustering ha chiaramente "riscoperto" l'esistenza di tre gruppi distinti ed equinumerosi all'interno del dataset.

Parte III

Domande ed esercizi

Appendice A

Domande teoriche e spunti di riflessione

Questo capitolo raccoglie alcune domande di autovalutazione per verificare autonomamente la comprensione degli argomenti del corso.

A.1 Domande a risposta aperta

In questa sezione sono raccolte domande a risposta aperta, da utilizzare come stimolo alla riflessione su alcuni argomenti del corso

A.1.1 Introduzione alla Data Science

Prerequisiti e modello di dati

- So descrivere il formato CSV? E XML?
- So interrogare un database relazionale?
- So calcolare la media e la varianza di una serie di dati numerici?

Formalizzazione

- Qual è la distinzione fra un problema di classificazione e uno di predizione?
- Perché in generale dobbiamo assumere che le $\boldsymbol{x}$ siano un vettore, mentre non si perde in generalità se si considerano solo y scalari?
- Perché non si punta sempre a un modello in grado di riprodurre perfettamente gli output dell'insieme con cui è addestrato?
- Perché il valor medio di ε in (1.1) è nullo?
- Perché abbiamo usato KNN come esempio per problemi di classificazione e non di regressione? Perché, viceversa, non abbiamo usato la regressione polinomiale per un problema di classificazione?

A.1.2 Algoritmi di machine learning

Modelli lineari, minimi quadrati

- Ho dimestichezza con la notazione vettoriale e matriciale?

- Riesco a riprodurre i passaggi matematici della soluzione del problema ai minimi quadrati nel caso scalare?
- E nel caso multidimensionale?
- Perché le $\phi_i(\cdot)$ si chiamano "funzioni di base"?

A.1.3 Metodologia

- Perché è importante che i sottoinsiemi di addestramento T e di validazione V costituiscano una partizione del dataset D?
- So motivare i termini "Sensibilità" e "Precisione"?
- Perché la K-fold cross validation è così utilizzata?
- Quali accorgimenti sono importanti per una buona cross-validation, soprattutto in condizioni estreme?

Modello logit

- So dimostrare le proprietà della funzione sigmoide?
- Come mai non si può ricavare il valore ottimale dei coefficienti lineari in forma algebrica, ma bisogna usare un metodo iterativo?
- Qual è il ruolo del coefficiente η nella discesa lungo il gradiente?
- Che cosa ottengo nei casi estremi della funzione di decisione (2.11), quando cioè imposto $\theta = 0$ oppure $\theta = 1$?

Alberi di decisione

- Perché l'entropia e l'impurità di Gini sono buoni indicatori della "purezza" dell'output di un dataset?
- Perché un albero rappresenta una struttura comoda per prendere delle decisioni?

A.1.4 Selezione degli attributi

- Perché il riquadro centrale in figura 7.1 a pag. 25 non riporta nessun valore? Non dovrebbe essere 0?
- Il coefficiente di correlazione è sempre nullo in caso di dipendenza non lineare?
- La "mutual information" definita in (7.4) sulla base dell'"entropia condizionale" (7.3) è sospettosamente simile all'"information gain" (6.5) definito sulla base dell'"entropia attesa" (6.4). Sono la stessa cosa?

Clustering

- In che caso (similitudine o distanza) devo minimizzare, e in che caso devo massimizzare?
- Che cos'è il *linkage criterion*?
- So leggere un dendrogramma?

A.2 Domande a risposta multipla

In questa sezione raccogliamo alcune domande a risposta multipla. Ogni domanda ha una sola risposta "corretta", anche se in alcuni casi è possibile che qualche ambiguità causi incertezze.

1. Quando si dice che un sistema di machine learning è "overfitting"?
   (a) Quando genera previsioni sistematicamente troppo alte.
   (b) Quando è troppo specializzato sui dati di addestramento e non è in grado di generare previsioni adeguate su dati nuovi.
   (c) Quando è stato addestrato su un insieme di addestramento troppo ampio e restano pochi dati per validarlo.

2. Quanti coefficienti vanno determinati un una regressione polinomiale di secondo grado in una variabile?
   (a) Quattro.
   (b) Tre.
   (c) Cinque.

3. Quale misura indica il numero di previsioni corrette per un modello di classificazione?
   (a) Accuratezza.
   (b) Precisione.
   (c) Sensibilità.

4. Perché è preferibile che gli insiemi di addestramento e di validazione siano disgiunti?
   (a) Perché siamo interessati a valutare le prestazioni del sistema su esempi non visti durante l'addestramento.
   (b) Si tratta di una precauzione per evitare che la presenza di due elementi identici nei dataset causi divisioni per zero.
   (c) Perché dobbiamo minimizzare le dimensioni dei due insiemi.

5. Che effetto ha la funzione sigmoide sull'uscita di un regressore?
   (a) Mappa il valore reale in uscita sull'intervallo $[0, 1]$.
   (b) Determina un valore di soglia per la decisione della classe di uscita.
   (c) Decide la classe di uscita.

6. Come può essere definita una funzione sigmoide?
   (a) $\frac{1}{1+e^{-t}}$.
   (b) $\frac{e^t}{e^t+1}$.
   (c) In entrambi i modi.

7. Quale misura occorre massimizzare per ridurre il numero di falsi negativi per un modello di classificazione?
   (a) Precisione.
   (b) Sensibilità.
   (c) Accuratezza.

8. Quando una partizione di un dataset si dice *stratificata*?

(a) Quando i campioni di ciascun sottoinsieme della partizione si trovano nello stesso ordine in cui compaiono nel dataset originale.

(b) Quando le numerosità dei diversi valori della classe di uscita nei sottoinsiemi della partizione sono approssimativamente uguali a quelle dell'intero dataset.

(c) Quando campioni simili vengono messi di preferenza nella stessa partizione.

9. È possibile utilizzare KNN per la classificazione se la classe in uscita ha più di due valori?

(a) Sì, ma è necessario addestrare un classificatore KNN per ciascun valore della classe, e poi confrontarne l'output.

(b) Sì, in quanto la funzione di decisione si basa sul valore di maggioranza, indipendentemente dal loro numero.

(c) No, è adatto solo a problemi di regressione.

10. In cosa consiste la K-fold cross-validation?

(a) Si separano i campioni in K gruppi distinti che si usano a rotazione per la validazione.

(b) Si usano K campioni alla volta per l'addestramento e i restanti per la validazione.

(c) Si separano gli attributi di ingresso in K gruppi distinti che si usano a rotazione per la validazione.

11. Che cos'è il metodo della discesa lungo il gradiente?

(a) Un metodo per trovare il valore ottimale di un classificatore KNN.

(b) Un metodo per trovare un minimo locale di una funzione differenziabile in più variabili reali.

(c) Un metodo per trovare le derivate parziali dell'output di un regressore logistico.

12. A cosa serve normalizzare le colonne di un dataset?

(a) A eguagliare gli intervalli di variabilità delle colonne.

(b) A eliminare eventuali elementi uguali a zero.

(c) A rimuovere eventuali valori negativi.

13. Qual è la definizione dell'Impurità di Gini di una variabile casuale Y?

(a) La probabilità di errore nel prevedere un esito $y \in Y$ se si sceglie un valore casuale $\tilde{y}$ con distribuzione di probabilità uniforme.

(b) La probabilità di errore nel prevedere un esito $y \in Y$ se si sceglie un valore casuale $\tilde{y}$ con la stessa distribuzione di probabilità.

(c) La probabilità di prevedere correttamente un esito $y \in Y$ se si sceglie un valore casuale $\tilde{y}$ con distribuzione di probabilità uniforme.

14. L'entropia di una variabile casuale Y...

(a) ... dipende soltanto dal dominio di Y.

(b) ... dipende sia dai valori di probabilità, sia dal dominio di Y.

(c) ... dipende soltanto dai valori di probabilità di Y.

15. L'impurità di Gini di una variabile casuale Y...

(a) ... dipende soltanto dal dominio di Y.

(b) ... dipende soltanto dai valori di probabilità di Y.

(c) ... dipende sia dai valori di probabilità, sia dal dominio di Y.

16. Per quale dei seguenti motivi è spesso opportuno usare la mediana della distribuzione come soglia per binarizzare una variabile continua, invece della media?

    (a) Perché il calcolo della mediana, non richiedendo somme e divisioni, è computazionalmente più efficiente del calcolo della media

    (b) Perché la mediana non risente molto della presenza di valori estremi (outliers).

    (c) Gli altri due motivi sono entrambi validi.

17. Se abbiamo una collezione di punti della forma (x, x^2), con $x \in [-1, 0]$ distribuito uniformemente, che valore ha il coefficiente di correlazione fra le due coordinate?

    (a) $\rho < 0$, perché la relazione fra la prima e la seconda coordinata è decrescente.

    (b) $\rho = 0$, perché la relazione non è lineare.

    (c) $\rho > 0$, perché x^2 è sempre positivo.

18. Quale tra i seguenti algoritmi visti a lezione **non** è greedy?

    (a) Il calcolo dell'entropia di una variabile casuale.

    (b) La costruzione di un albero di decisione.

    (c) La selezione iterativa degli attributi sulla base dell'informazione mutua.

19. Ad ogni iterazione dell'algoritmo di clustering agglomerativo gerarchico su una matrice di distanze, come si scelgono i due cluster da aggregare?

    (a) Si scelgono sempre i due cluster aventi distanza minima.

    (b) Si scelgono sempre i due cluster aventi distanza massima.

    (c) Si scelgono i due cluster aventi distanza minima o massima, a seconda del linkage criterion scelto.

20. Quale linkage criterion tende a generare dendrogrammi più bilanciati?

    (a) Il single linkage.

    (b) Il complete linkage.

    (c) Il linkage criterion non ha influenza sul bilanciamento.

Appendice B

Esercizi

Esercizio 1

Un modello di previsione è addestrato a valutare la presenza di una specifica patologia in un paziente sulla base di alcuni attributi.
Il modello viene valutato su una popolazione di pazienti per i quali l'effettiva presenza della patologia è stata controllata da un medico, ottenendo i seguenti responsi:

Paziente	1	2	3	4	5	6	7	8	9	10
Patologia	Sì	No	No	Sì	Sì	No	No	Sì	No	Sì
Previsione	Sì	No	Sì	Sì	Sì	No	No	No	No	Sì

Paziente	11	12	13	14	15	16	17	18	19	20
Patologia	No	No	No	Sì	Sì	No	Sì	No	No	Sì
Previsione	Sì	No	No	No	Sì	No	Sì	No	Sì	Sì

La riga "Patologia" riporta il responso del medico, da considerarsi sempre corretto; la riga "Previsione" riporta la valutazione del modello.
1.1) Calcolare la matrice di confusione dell'esperimento.
1.2) Calcolare le varie misure considerate: accuratezza, sensibilità, precisione, F_1-score del modello.

Soluzione 1

1.1) Contiamo quante volte compare ciascuna delle quattro combinazioni "Sì/Sì" (TP), "Sì/No" (FN), "No/Sì" (FP), "No/No" (TN):

		Previsione	
		Sì	No
Patologia	Sì	7	2
	No	3	8

1.2)

$$\text{Accuratezza} = \frac{7+8}{20} = \frac{3}{4} = 0,75; \qquad \text{Sensibilità} = \frac{7}{7+2} = \frac{7}{9} \approx 0,78;$$

$$\text{Precisione} = \frac{7}{7+3} = \frac{7}{10} = 0,7; \qquad F_1\text{-score} = \frac{2 \cdot \frac{7}{9} \cdot \frac{7}{10}}{\frac{7}{9} + \frac{7}{10}} = \frac{14}{19} \approx 0,74.$$

Possiamo osservare che in un sistema diagnostico medico i falsi negativi sono particolarmente gravi. Infatti, mentre una diagnosi falsamente positiva è solitamente corretta da esami ulteriori, una diagnosi negativa rischia di non avere seguito e di lasciar progredire la patologia. È dunque necessario prestare particolare attenzione alla casella FN = 2.

Esercizio 2

È dato il seguente dataset ($m = 4$ campioni, x e y scalari):

	x	y
1	6	2
2	9	5
3	5	3
4	12	6

Determinare il coefficiente β del modello lineare

$$y \sim \beta x$$

che minimizza l'errore quadratico medio rispetto ai dati forniti.

Soluzione 2

Vogliamo determinare il valore di β che minimizza la seguente somma di scarti al quadrato:

$$f(\beta) = (6\beta - 2)^2 + (9\beta - 5)^2 + (5\beta - 3)^2 + (12\beta - 6)^2.$$

Deriviamo f rispetto a β ed eguagliamo a zero:

$$\frac{\mathrm{d}}{\mathrm{d}\beta} f(\beta) = 2 \cdot 6 \cdot (6\beta - 2) + 2 \cdot 9 \cdot (9\beta - 5) + 2 \cdot 5 \cdot (5\beta - 3) + 2 \cdot 12 \cdot (12\beta - 6) = 0.$$

Semplificando i fattori pari a 2 in tutti i termini:

$$36\beta - 12 + 81\beta - 45 + 25\beta - 15 + 144\beta - 72 = 286\beta - 144 = 0,$$

infine

$$\beta = \frac{144}{286} = \frac{72}{143} \approx 0,503.$$

Osserviamo infatti che i valori di y sono sempre vicini alla metà di x, quindi è prevedibile un valore di β vicino a $1/2$.

Esercizio 3

Nello stesso contesto dell'esercizio 1, supponiamo che il modello di previsione sia sostituito da un modello logistico che restituisce la probabilità che la patologia sia positiva:

i	1	2	3	4	5	6	7	8	9	10
y_i	Sì	No	No	Sì	Sì	No	No	Sì	No	Sì
$\tilde{y}_i$	.7	.4	.6	.8	.7	.1	.4	.6	.5	.8

i	11	12	13	14	15	16	17	18	19	20
y_i	No	No	No	Sì	Sì	No	Sì	No	No	Sì
$\tilde{y}_i$	.7	.2	.4	.4	.6	.3	.5	.6	.6	.7

Per rispondere "Sì" oppure "No", il sistema si basa su un valore di soglia θ preimpostato. La previsione sarà:

$$\text{Previsione}_i = \begin{cases} \text{Sì} & \text{se } \tilde{y}_i \geq \theta \\ \text{No} & \text{altrimenti.} \end{cases}$$

3.1) Supponiamo che il sistema decida di effettuare una previsione positiva se il risultato del modello logistico è maggiore o uguale a $\theta = 0.5$. Ricavare la corrispondente matrice di confusione e i principali criteri di valutazione delle prestazioni: accuratezza, precisione, sensibilità, F_1-score.
3.2) Valutare gli stessi criteri per una probabilità di soglia pari a 0.7. Trattandosi di una decisione su una patologia, è meglio utilizzare un valore di θ grande o piccolo?

Soluzione 3

3.1) Con la soglia pari a $\theta = 0,5$, la tabella delle previsioni diventa:

i	1	2	3	4	5	6	7	8	9	10
y_i	Sì	No	No	Sì	Sì	No	No	Sì	No	Sì
$\tilde{y}_i$	Sì	No	Sì	Sì	Sì	No	No	Sì	Sì	Sì

i	11	12	13	14	15	16	17	18	19	20
y_i	No	No	No	Sì	Sì	No	Sì	No	No	Sì
$\tilde{y}_i$	Sì	No	No	No	Sì	No	Sì	Sì	Sì	Sì

Contiamo quante volte compare ciascuna delle quattro combinazioni "Sì/Sì" (TP), "Sì/No" (FN), "No/Sì" (FP), "No/No" (TN):

		Previsione	
		Sì	No
Patologia	Sì	8	1
	No	5	6

$$\text{Accuratezza} = \frac{8+6}{20} = \frac{7}{10} = 0,7; \qquad \text{Sensibilità} = \frac{8}{8+1} = \frac{8}{9} \approx 0,89;$$

$$\text{Precisione} = \frac{8}{8+5} = \frac{8}{13} \approx 0,62; \qquad F_1\text{-score} = \frac{2 \cdot \frac{8}{9} \cdot \frac{8}{13}}{\frac{8}{9} + \frac{8}{13}} = \frac{16}{22} \approx 0,73.$$

3.2) Con la soglia pari a $\theta = 0,7$, la tabella delle previsioni diventa:

i	1	2	3	4	5	6	7	8	9	10
y_i	Sì	No	No	Sì	Sì	No	No	Sì	No	Sì
$\tilde{y}_i$	Sì	No	No	Sì	Sì	No	No	No	No	Sì

i	11	12	13	14	15	16	17	18	19	20
y_i	No	No	No	Sì	Sì	No	Sì	No	No	Sì
$\tilde{y}_i$	Sì	No	No	No	No	No	No	No	No	Sì

La matrice di confuzione ora sarà

		Previsione	
		Sì	No
Patologia	Sì	5	4
	No	1	10

$$\text{Accuratezza} = \frac{5+10}{20} = \frac{3}{4} = 0,75; \qquad \text{Sensibilità} = \frac{5}{5+4} = \frac{5}{9} \approx 0,56;$$

$$\text{Precisione} = \frac{5}{5+1} = \frac{5}{6} \approx 0,83; \qquad F_1\text{-score} = \frac{2 \cdot \frac{5}{9} \cdot \frac{5}{6}}{\frac{5}{9} + \frac{5}{6}} = \frac{2}{3} \approx 0,67.$$

Come osservato in precedenza, in un sistema diagnostico medico è meglio privilegiare la sensibilità, quindi abbassare la soglia. In questo caso si può osservare che non tutti i criteri sono concordi.

Esercizio 4

È dato il seguente dataset, di 12 campioni $i = 1, \ldots, 12$, composti da un solo attributo scalare $x_i \in [0, 10]$ come variabile indipendente e una classe a due valori $y_i \in \{\text{Freddo}, \text{Caldo}\}$ come valore da prevedere:

i	x_i	y_i
1	0.7	Caldo
2	8.7	Freddo
3	3.5	Freddo
4	1.5	Caldo

i	x_i	y_i
5	4.4	Freddo
6	6.1	Caldo
7	6.8	Caldo
8	7.2	Caldo

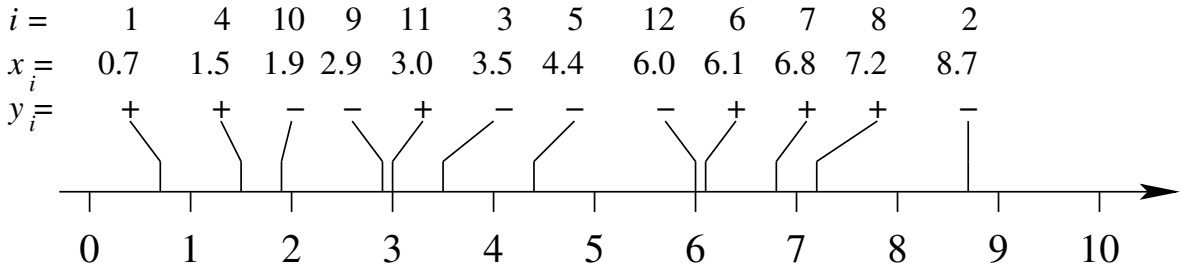
i	x_i	y_i
9	2.9	Freddo
10	1.9	Freddo
11	3.0	Caldo
12	6.0	Freddo

4.1) Tramite la metodologia leave-one-out, valutare la prestazione dell'algoritmo K-nearest-neighbors con $K = 1$ e $K = 3$ sulla base degli indici di accuratezza, precisione, sensibilità e dell'F_1-score.
4.2) Quale scelta si rivela essere la migliore per il parametro K? Lo è in maniera univoca, oppure dipende dal criterio considerato?

Suggerimento — *Riportare i valori x_i su un asse in modo da semplificare la ricerca dei vicini. In caso di parità di distanze, scegliere l'elemento di indice minore.*

Soluzione 4

Consideriamo "Caldo" come positivo, "Freddo" come negativo. In base al suggerimento, ecco i valori riportati sulle ascisse:



i	Vicini			y_i	$\tilde{y}_i$	
					$K = 1$	$K = 3$
1	+	−	−	+	+	−
2	+	+	+	−	+	+
3	+	−	−	−	+	−
4	−	+	−	+	−	−
5	−	+	−	−	−	−
6	−	+	+	+	−	+
7	+	+	−	+	+	+
8	+	+	−	+	+	+
9	+	−	−	−	+	−
10	+	−	+	−	+	+
11	−	−	−	+	−	−
12	+	+	+	−	+	+

4.1) La matrice di confusione per $K = 1$ è dunque

		Previsione	
		+	−
Risposta	+	3	3
corretta	−	5	1

da cui risulta:

$$\text{Accuratezza} = \frac{3+1}{12} = \frac{1}{3}, \quad \text{Precisione} = \frac{3}{3+5} = \frac{3}{8}, \quad \text{Sensibilità} = \frac{3}{3+3} = \frac{1}{2}, \quad F_1 = \frac{2\frac{3}{8}\frac{1}{2}}{\frac{3}{8}+\frac{1}{2}} = \frac{3}{7}.$$

Si osservi che, invertendo l'interpretazione della classe positiva e negativa, si ottiene invece la seguente matrice:

		Previsione	
		+	−
Risposta	+	1	5
corretta	−	3	3

da cui risulta:

$$\text{Accuratezza} = \frac{1+3}{12} = \frac{1}{3}, \quad \text{Precisione} = \frac{1}{1+3} = \frac{1}{4}, \quad \text{Sensibilità} = \frac{1}{1+5} = \frac{1}{6}, \quad F_1 = \frac{2\frac{1}{4}\frac{1}{6}}{\frac{1}{4}+\frac{1}{6}} = \frac{1}{5}.$$

Allo stesso modo, la matrice di confusione per $K = 3$ risulta

		Previsione	
		+	−
Risposta	+	3	3
corretta	−	3	3

da cui risulta:

$$\text{Accuratezza} = \frac{3+3}{12} = \frac{1}{2}, \quad \text{Precisione} = \frac{3}{3+3} = \frac{1}{2}, \quad \text{Sensibilità} = \frac{3}{3+3} = \frac{1}{2}, \quad F_1 = \frac{1}{2}.$$

Visto che la precisione e la sensibilità sono uguali, allora anche la loro media F_1 è lo stesso valore.
4.2) Rispetto a tutti gli indici considerati, $K = 3$ risulta preferibile rispetto a $K = 1$.

Esercizio 5

È dato il seguente dataset, con 15 campioni, un attributo in ingresso, e una classe a due valori in uscita.

i	x_i	y_i	i	x_i	y_i	i	x_i	y_i
1	4.0	Sì	6	7.3	Sì	11	3.2	No
2	2.8	No	7	1.2	No	12	8.4	No
3	6.1	Sì	8	0.7	No	13	0.2	No
4	9.2	No	9	5.5	Sì	14	6.8	Sì
5	4.5	Sì	10	6.0	No	15	4.3	Sì

5.1) Valutare la prestazione dell'algoritmo K-nearest-neighbors con $K = 1$ e $K = 3$, utilizzando una 3-fold cross validation in cui i tre fold sono gli elementi di indici 1–5, 6–10 e 11–15. Utilizzare gli indici di accuratezza, precisione, sensibilità e l'F_1-score.
5.2) Quale scelta si rivela essere la migliore per il parametro K? Lo è in maniera univoca, oppure dipende dal parametro considerato?

Soluzione 5

Vedere l'esercizio precedente; la differenza principale è nella richiesta di effettuare una 3-fold cross-validation invece del leave-one-out.

Esercizio 6

È dato il seguente dataset con variabili indipendenti scalari $x_i \in [0, 100]$ e variabile dipendente continua $y_i \in \mathbb{R}$:

i	1	2	3	4	5	6
x_i	9.2	4.8	5.5	8	2.7	6.9
y_i	11.4	6.8	7.7	10.8	5.7	519.5

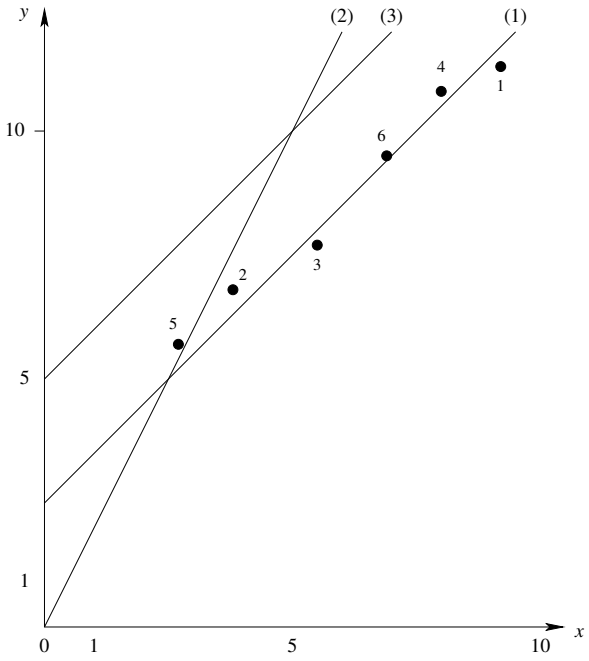
Si desidera modellare il dataset tramite una regressione affine nella forma

$$y \sim \beta_0 + \beta_1 x.$$

Determinare graficamente (tracciando i punti e le rette su un piano cartesiano) quale tra le seguenti coppie di coefficienti genera l'errore quadratico minore:
6.1) $\beta_0 = 2.5$, $\beta_1 = 1$
6.2) $\beta_0 = 0$, $\beta_1 = 2$
6.3) $\beta_0 = 5$, $\beta_1 = 1$

Soluzione 6

Ecco i punti e le rette richieste riportate nel piano cartesiano:



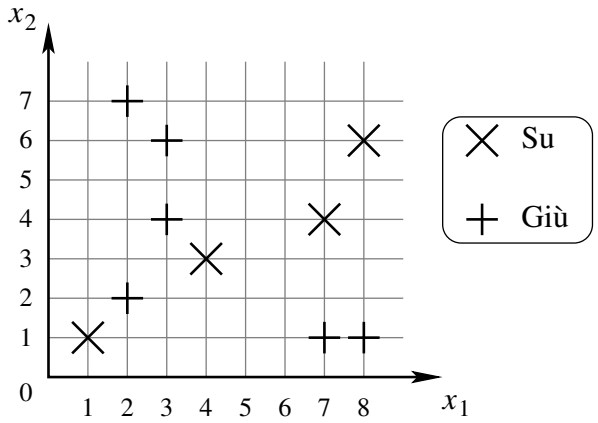
Risulta immediatamente chiaro che la retta che meglio approssima i punti fra le tre fornite è la **(1)**, corrispondente all'equazione affine $y \sim 2.5 + x$.

Esercizio 7

È dato il seguente dataset di $m = 10$ campioni, con $n = 2$ attributi numerici X_1, X_2 e un output Y categorico (binario):

i	1	2	3	4	5	6	7	8	9	10
x_{i1}	1	2	3	4	7	2	8	3	7	8
x_{i2}	1	2	4	3	1	7	1	6	4	6
y_i	Su	Giù	Giù	Su	Giù	Giù	Giù	Giù	Su	Su

In forma grafica:



Valutare sul dataset fornito il classificatore KNN con $K = 1$ e $K = 3$ rispetto ai principali indici di prestazione (accuratezza, precisione, sensibilità, F_1-score) utilizzando la metodologia leave-one-out.

Suggerimento — *La maggior parte delle distanze può essere valutata a occhio, ricorrere a calcoli solo per i pochi casi dubbi.*
Non è necessario calcolare le radici quadrate.

Soluzione 7

Riprendiamo la tabella del dataset e inseriamo per ogni elemento i primi tre vicini (metodologia leave-one-out: per l'addestramento si usa tutto il dataset tranne il campione su cui valutiamo, e si ripete per tutti i campioni):

i	1	2	3	4	5	6	7	8	9	10
x_{i1}	1	2	3	4	7	2	8	3	7	8
x_{i2}	1	2	4	3	1	7	1	6	4	6
y_i	Su	Giù	Giù	Su	Giù	Giù	Giù	Giù	Su	Su
1. vicino	Giù	Su	Su	Giù	Giù	Giù	Giù	Giù	Su	Su
2. vicino	Su	Giù	Giù	Giù	Su	Giù	Su	Giù	Giù	Giù
3. vicino	Giù	Su	Giù	Su*	Su	Su	Su	Su	Su*	Su

* — Alla pari con un "Giù", ho scelto arbitrariamente.

La matrice di confusione per $K = 1$, quindi utilizzando il primo vicino come predittore, risulta:

		Previsione	
		Su	Giù
Classe corretta	Su	2	2
	Giù	2	4

Un'altra scelta arbitraria: quale classe considerare "positiva". Nel mio caso scelgo "Su". Di conseguenza,

$$\text{accuratezza} = \frac{2+4}{10} = \frac{3}{5}; \quad \text{precisione} = \frac{2}{2+2} = \frac{1}{2}; \quad \text{sensibilità} = \frac{2}{2+2} = \frac{1}{2}.$$

La precisione e la sensibilità sono uguali, quindi lo è anche la loro media armonica:

$$F_1 = \frac{1}{2}.$$

La matrice di confusione per $K = 3$, quindi utilizzando i 3 primi vicini a maggioranza, risulta:

		Previsione	
		Su	Giù
Classe corretta	Su	2	2
	Giù	3	3

Di conseguenza la situazione peggiora:

$$\text{accuratezza} = \frac{2+3}{10} = \frac{1}{2}; \quad \text{precisione} = \frac{2}{2+3} = \frac{2}{5}; \quad \text{sensibilità} = \frac{2}{2+2} = \frac{1}{2}.$$

Infine,

$$F_1 = \frac{2 \cdot \frac{2}{5} \cdot \frac{1}{2}}{\frac{2}{5} + \frac{1}{2}} = \frac{4}{9}.$$

Esercizio 8

Sia dato il seguente dataset di $m = 6$ campioni, con 1 attributo numerico x e 1 output numerico y:

i	1	2	3	4	5	6
x	1	2	-2	0	-1	0
y	3	8	0	15	-1	0

Si richiede di apprendere il modello $y \sim \beta\phi(x)$ attraverso il metodo della regressione lineare ai minimi quadrati, applicato al dataset risultante dalla trasformazione non lineare $\phi(x) = x^2$.

Soluzione 8

Vedere la sezione 3.2.1 delle dispense.

8.1) Costruiamo il vettore z in modo che $z_i = x_i^2$:

$$z = \begin{pmatrix} 1 \\ 4 \\ 4 \\ 0 \\ 1 \\ 0 \end{pmatrix}.$$

Considerando z e y come vettori colonna, il parametro β si calcola come segue:

$$\beta = \frac{\sum_{i=1}^{6} y_i z_i}{\sum_{i=1}^{6} z_i^2} = \frac{34}{34} = 1.$$

Alternativa: applicazione diretta del metodo dei minimi quadrati — Consideriamo la somma dei quadrati degli scarti in funzione di β:

$$f(\beta) = (\beta - 3)^2 + (4\beta - 8)^2 + (4\beta)^2 + (-15)^2 + (\beta + 1)^2,$$

e cerchiamone il valore stazionario azzerandone la derivata prima rispetto a β:

$$0 = \frac{\mathrm{d}f(\beta)}{\mathrm{d}\beta} = 2(\beta - 3) + 2 \cdot 4(4\beta - 8) + 2 \cdot 4(4\beta) + 2(\beta + 1) = 68\beta - 68,$$

la cui soluzione è nuovamente $\beta = 1$.

Esercizio 9

Utilizzando lo stesso dataset dell'Esercizio 5, un albero di decisione deve discriminare in base alla colonna x utilizzando il criterio di impurità di Gini.
Dovendo scegliere fra le due soglie $\theta = 3.5$ e $\theta = 7$, quale risulterà migliore?

Soluzione 9

Iniziamo con $\theta = 3.5$. Stimando le probabilità con le frequenze otteniamo:

$$\begin{aligned} &\Pr(Y = \text{Sì}|X < 3.5) = 0; \qquad \Pr(Y = \text{No}|X \geq 3.5) = 1; \\ &\Pr(Y = \text{Sì}|X < 3.5) = \frac{7}{10}; \qquad \Pr(Y = \text{No}|X \geq 3.5) = \frac{3}{10}. \end{aligned} \tag{B.1}$$

Il caso $X < 3.5$ è puro (ci sono solo "No"), quindi

$$GI(Y|X < 3.5) = 0.$$

Calcoliamo l'impurità di Gini del caso $X \geq 3.5$:

$$\begin{aligned} GI(Y|X \geq 3.5) &= 1 - \left(\Pr(Y = \text{Sì}|X \geq 3.5)^2 + \Pr(Y = \text{No}|X \geq 3.5)^2\right) \\ &= 1 - \left(\frac{49}{100} + \frac{9}{100}\right) = \frac{21}{50}. \end{aligned}$$

La media pesata sulla base della probabilità dei due casi, che fornisce l'impurità attesa, è dunque:

$$\begin{aligned} GI(Y|X) &= GI(Y|X < 3.5)\Pr(X < 3.5) + GI(Y|X \geq 3.5)\Pr(x \geq 3.5) \\ &= 0\frac{5}{15} + \frac{21}{50}\frac{10}{15} = \frac{7}{25}. \end{aligned}$$

Per $\theta = 7$, un analogo conteggio delle righe e degli output fornisce:

$$\begin{aligned} &\Pr(Y = \text{Sì}|X < 7) = \Pr(Y = \text{No}|X \geq 7) = \frac{1}{2}; \\ &\Pr(Y = \text{Sì}|X < 7) = \frac{1}{3}; \qquad \Pr(Y = \text{No}|X \geq 7) = \frac{2}{3}. \end{aligned} \tag{B.2}$$

Osserviamo come entrambe le distribuzioni siano molto più uniformi di prima, il che ci fa pensare che $\theta = 7$ non sia una buona scelta. Infatti, l'impurità di Gini per il caso $X < 7$ è la più elevata possibile per una variabile a due valori:

$$\begin{aligned} GI(Y|X < 7) &= 1 - \left(\Pr(Y = \text{Sì}|X < 7)^2 + \Pr(Y = \text{No}|X < 7)^2\right) \\ &= 1 - \left(\frac{1}{4} + \frac{1}{4}\right) = \frac{1}{2}, \end{aligned}$$

L'impurità del caso $X \geq 7$ è invece:

$$\begin{aligned} GI(Y|X \geq 7) &= 1 - \left(\Pr(Y = \text{Sì}|X \geq 7)^2 + \Pr(Y = \text{No}|X \geq 7)^2\right) \\ &= 1 - \left(\frac{1}{9} + \frac{4}{9}\right) = \frac{4}{9}. \end{aligned}$$

L'impurità attesa, calcolata anche in questo caso come media pesata delle due impurità, risulta essere:

$$\begin{aligned} GI(Y|X) &= GI(Y|X < 7)\Pr(X < 7) + GI(Y|X \geq 7)\Pr(x \geq 7) \\ &= \frac{1}{2}\frac{12}{15} + \frac{4}{9}\frac{3}{15} = \frac{22}{45}, \end{aligned}$$

un valore decisamente più elevato rispetto alla prima soglia.
La soglia $\theta = 3.5$ è dunque migliore perché porta a un'impurità attesa più bassa.

Esercizio 10

È dato il seguente dataset di 8 elementi con due attributi categorici e output categorico:

x_1	Verde	Rosso	Rosso	Verde	Rosso	Verde	Rosso	Verde
x_2	Sì	No	Sì	Sì	No	No	Sì	No
y	+	+	+	-	+	-	-	-

10.1) Quale domanda apparirà al livello più alto (radice) di un albero di decisione, e perché?
10.2) Qual è l'impurità di Gini dell'output del dataset completo, e quale l'impurità media dei due nodi figli?

Soluzione 10

10.1) Osserviamo che nel dataset complessivo le due classi di output sono equiprobabili: ci troviamo dunque nel caso peggiore sia dal punto di vista dell'entropia, sia in base all'impurità di Gini:

$$\Pr(Y = +) = \Pr(Y = -) = \frac{1}{2}.$$

Se discriminiamo i dati sulla base della prima colonna, la distribuzione delle Y cambia:

$$\begin{aligned} \Pr(Y = +|X_1 = \text{Verde}) &= \frac{1}{4}; \qquad \Pr(Y = -|X_1 = \text{Verde}) = \frac{3}{4}; \\ \Pr(Y = +|X_1 = \text{Rosso}) &= \frac{3}{4}; \qquad \Pr(Y = -|X_1 = \text{Rosso}) = \frac{1}{4}. \end{aligned} \tag{B.3}$$

Ne segue che la prima colonna contiene informazione utile a determinare l'output. Discriminando in base alla seconda colonna, invece, le distribuzioni dell'output nei due sottocasi restano uniformi:

$$\Pr(Y = +|X_2 = \text{Sì}) = \Pr(Y = -|X_2 = \text{Sì}) = \Pr(Y = +|X_2 = \text{No}) = \Pr(Y = -|X_2 = \text{No}) = \frac{1}{2}.$$

Il contenuto informativo è nullo, quindi la radice conterrà la domanda "X_1 è Verde?" (o Rosso).
10.2) Date le distribuzioni trovate in (B.3), possiamo calcolare l'impurità di Gini dei due figli:

$$\begin{aligned} GI(Y|X_1 = \text{Verde}) &= 1 - \left(\Pr(Y = +|X_1 = \text{Verde})^2 + \Pr(Y = -|X_1 = \text{Verde})^2\right) \\ &= 1 - \left(\frac{1}{16} + \frac{9}{16}\right) = \frac{3}{8}; \end{aligned}$$

Esercizio 11

Si consideri il seguente dataset:

Genere	Auto possedute	Costo del viaggio	Reddito	Mezzo di trasporto
M	0	Basso	Basso	Autobus
M	1	Basso	Medio	Autobus
F	1	Basso	Medio	Treno
F	0	Basso	Basso	Autobus
M	1	Basso	Medio	Autobus
M	0	Normale	Medio	Treno
F	1	Normale	Medio	Treno
F	1	Alto	Alto	Auto
M	2	Alto	Medio	Auto
F	2	Alto	Alto	Auto

11.1) Considerando le prime quattro colonne come dati in ingresso, Costruire l'albero di decisione sul mezzo di trasporto basato sull'algoritmi ID3 in cui ogni nodo massimizza l'information gain.
11.2) Utilizzando l'albero appena costruito, prevedere il mezzo di trasporto per gli elementi seguenti:

Genere	Auto possedute	Costo del viaggio	Reddito
M	1	Normale	Alto
M	0	Basso	Medio
F	1	Basso	Alto

Esercizio 12

È dato un insieme di cinque elementi con le seguenti distanze mutue:

	B	C	D	E
A	6	1	5	3
B		7	2	4
C			6	3
D				5

12.1) Eseguire la procedura di clustering agglomerativo gerarchico con single-linkage criterion.
12.2) Ripetere la procedura con il complete linkage criterion.

Soluzione 12

12.1) I due elementi più vicini sono A e C, con distanza 1. Uniamoli in un singolo cluster e riscriviamo la tabella tenendo conto del single linkage (distanza minima):

	B	D	E
A,C	6	5	3
B		2	4
D			5

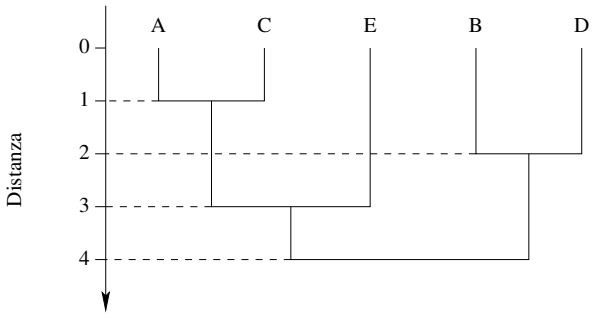
A questo punto selezioniamo B e D, ottenendo:

	B,D	E
A,C	5	3
B,D		4

Ora uniamo E ad A,C:

	B,D
A,C,E	4

Infine uniamo insieme i due cluster rimanenti. Il dendrogramma risultante è



12.2) La soluzione con il complete linkage è molto simile.

Esercizio 13

Considerare l'insieme $\mathcal{M}$ di cinque strumenti musicali:

$$\mathcal{M} = \{\text{Violino}, \text{Corno francese}, \text{Sassofono}, \text{Fagotto}, \text{Ottavino}\}.$$

La seguente tabella descrive l'intervallo di frequenze suonabili da ciascuno strumento:

Strumento	Tono più basso (Hz)	Tono più alto (Hz)
Violino	100	2600
Corno francese	80	1300
Sassofono	230	1400
Fagotto	60	660
Ottavino	600	4200

La *similarità* $\text{sim}(a, b)$ tra due strumenti è data dall'ampiezza dell'intervallo comune ai due strumenti. Ad esempio, il violino e il fagotto si sovrappongono fra 100Hz e 660Hz, quindi la loro similarità è

$$\text{sim}(\text{Violino}, \text{Fagotto}) = 660 - 100 = 560.$$

13.1) Applicare una procedura di clustering agglomerativo gerarchico all'insieme $\mathcal{M}$. Si utilizzi il *complete linkage criterion*, considerando che stiamo lavorando con similitudini e non distanze:

$$\text{sim}(A, B) = \min_{\substack{a \in A \\ b \in B}} \text{sim}(a, b).$$

Tracciare il corrispondente dendrogramma.
13.2) Ripetere l'operazione utilizzando il *single linkage criterion* e tracciare il corrispondente dendrogramma.

Soluzione 13

La tabella delle distanze, realizzata considerando per ogni coppia di strumenti l'ampiezza dell'intesezione fra gli intervalli delle frequenze, è la seguente:

	C	S	F	O
V	1200	1170	560	2000
C		1070	580	700
S			430	800
F				60

13.1) I due strumenti più simili sono il violino (V) e l'ottavino (O), dunque li raccogliamo in un cluster $\{\text{V}, \text{O}\}$ e calcoliamo le similarità di quest'ultimo con gli altri strumenti utilizzando il *complete linkage criterion* (minima similarità) richiesto in questo punto:

	C	S	F
$\{\text{V}, \text{O}\}$	700	800	60
C		1070	580
S			430

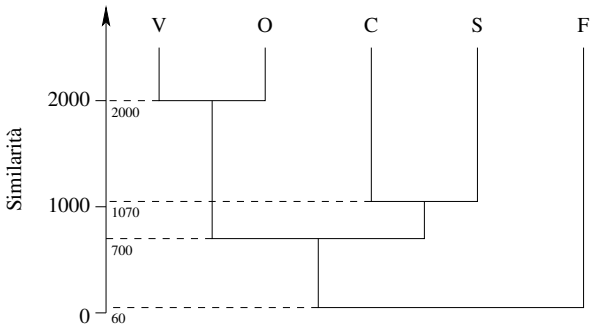
La massima similarità nella tabella è fra gli elementi C ed S, che raccogliamo nel cluster $\{\text{C}, \text{S}\}$. Calcoliamo le nuove distanze:

	$\{\text{C}, \text{S}\}$	F
$\{\text{V}, \text{O}\}$	700	60
$\{\text{C}, \text{S}\}$		430

A questo punto, uniremo i due cluster $\{\text{V}, \text{O}\}$ e $\{\text{C}, \text{S}\}$:

	F
$\{\text{V}, \text{O}, \text{C}, \text{S}\}$	60

Quindi uniamo gli ultimi due cluster. Risulta il seguente dendrogramma:



13.2) Come prima, alla prima iterazione i due strumenti più simili sono il violino (V) e l'ottavino (O), dunque li raccogliamo in un cluster $\{\mathrm{V}, \mathrm{O}\}$ e calcoliamo le similarità di quest'ultimo con gli altri strumenti, utilizzando però il *single linkage criterion* (massima similarità) richiesto in questo punto:

	C	S	F
$\{\mathrm{V}, \mathrm{O}\}$	1200	1170	660
C		1070	580
S			430

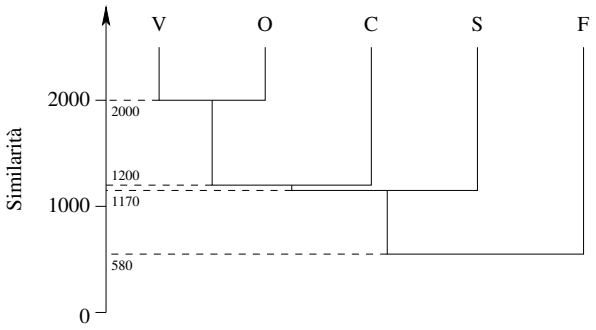
La massima similarità nella tabella è fra il cluster $\{\mathrm{V}, \mathrm{O}\}$ e l'elemento C; formiamo dunque il cluster $\{\mathrm{V}, \mathrm{O}, \mathrm{C}\}$. Calcoliamo le nuove distanze:

	S	F
$\{\mathrm{V}, \mathrm{O}, \mathrm{C}\}$	1170	580
S		430

A questo punto, uniremo il cluster $\{\mathrm{V}, \mathrm{O}, \mathrm{C}\}$ e l'elemento S:

	F
$\{\mathrm{V}, \mathrm{O}, \mathrm{C}, \mathrm{S}\}$	580

Quindi uniamo gli ultimi due cluster. Risulta il seguente dendrogramma:



Esercizio 14

La seguente tabella mostra due collezioni di papiri dell'antico Egitto (ogni riga è un diverso documento).

	La collezione di Alice		**La collezione di Bob**
Papiro 1.		Papiro 1.	
Papiro 2.		Papiro 2.	
Papiro 3.		Papiro 3.	
Papiro 4.		Papiro 4.	

14.1) Dato che ogni glifo rappresenta una parola, calcolare i coefficienti di Jaccard fra i quattro documenti della collezione di Alice. Ripetere il calcolo per i quattro documenti nella collezione di Bob.
14.2) Un esperto sostiene che i papiri della collezione di Bob riguardano due argomenti distinti, mentre la collezione di Alice è più uniforme.
Anche se probabilmente non conoscete l'antico egizio, applicate una procedura di clustering agglomerativo per sostenere questa tesi. Utilizzare le somiglianze fra documenti calcolate al punto 14.1 e il *single linkage criterion* (massima similarità).

Soluzione 14

È possibile verificare la soluzione con il codice disponibile al link

`http://disi.unitn.it/~brunato/AA/alicebob.py`

Per comodità, riscriviamo i documenti in possesso di Alice e Bob come insiemi, sostituendo ai geroglifici delle lettere latine:

$$a_1 = \{\mathrm{A}, \mathrm{B}, \mathrm{C}, \mathrm{D}\}; \qquad a_2 = \{\mathrm{E}, \mathrm{C}, \mathrm{B}, \mathrm{F}\}; \qquad a_3 = \{\mathrm{G}, \mathrm{F}, \mathrm{E}, \mathrm{B}\}; \qquad a_4 = \{\mathrm{B}, \mathrm{H}, \mathrm{F}, \mathrm{G}\};$$

$$b_1 = \{\mathrm{A}, \mathrm{B}, \mathrm{H}, \mathrm{D}\}; \qquad b_2 = \{\mathrm{A}, \mathrm{E}, \mathrm{F}, \mathrm{G}\}; \qquad b_3 = \{\mathrm{H}, \mathrm{B}, \mathrm{D}, \mathrm{F}\}; \qquad b_4 = \{\mathrm{E}, \mathrm{A}, \mathrm{G}, \mathrm{C}\}.$$

Ricordiamo che il coefficiente di Jaccard di due insiemi è dato dalla misura della loro intersezione divisa per la misura dell'unione. Ad esempio:

$$r'(a_1, a_2) = \frac{|a_1 \cap a_2|}{|a_1 \cup a_2|} = \frac{|\{\mathrm{B}, \mathrm{C}\}|}{|\{\mathrm{A}, \mathrm{B}, \mathrm{C}, \mathrm{D}, \mathrm{E}, \mathrm{F}\}|} = \frac{1}{3}.$$

La tabella completa della collezione di Alice risulta:

	a_2	a_3	a_4
a_1	1/3	1/7	1/7
a_2		3/5	1/3
a_3			3/5

Fra le due similitudini massime (3/5) scegliamo quella fra a_2 e a_3 e li uniamo in un cluster. La tabella delle distanze, tenuto conto del single linkage, diventa

	$\{a_2, a_3\}$	a_4
a_1	1/3	1/7
$\{a_2, a_3\}$		3/5

L'elemento a_4 viene dunque aggregato al cluster $\{a_2, a_3\}$ e le similarità diventano:

	a_1
$\{a_2, a_3, a_4\}$	1/7

Per la collezione di Bob, la tabella delle similitudini è la seguente:

	b_2	b_3	b_4
b_1	1/7	3/5	1/7
b_2		1/7	3/5
b_3			0

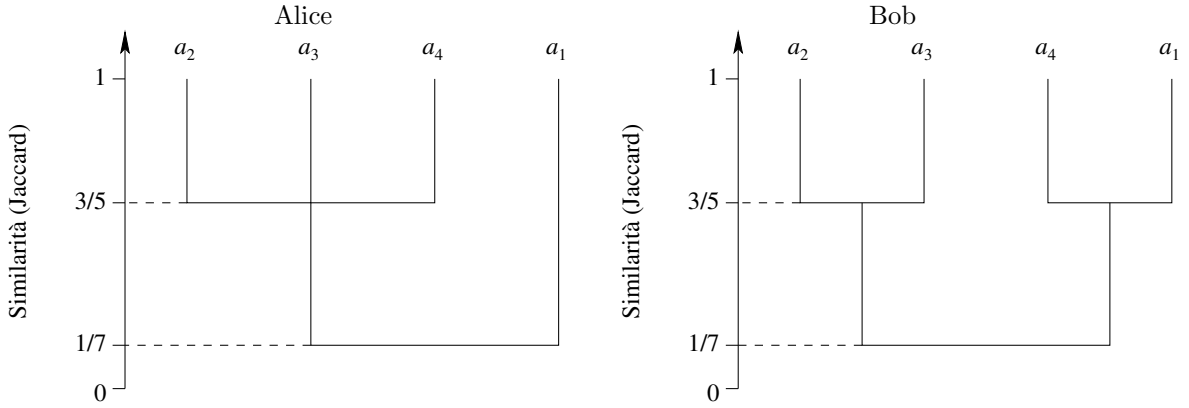
Al primo passo scegliamo di unire b_1 e b_3:

	b_2	b_4
$\{b_1, b_3\}$	1/7	1/7
b_2		3/5

Segue l'unione di b_2 e b_4:

	$\{b_2, b_4\}$
$\{b_1, b_3\}$	1/7

Ecco i due dendrogrammi a confronto:



Dalle figure si nota come la collezione di Bob sia maggiormente bilanciata su due argomenti distinti, a differenza di quella di Alice in cui tre papiri risultano simili.