

10. Apprendimento e Incertezza

Dispensa ICon

versione: 16/12/2024, 13:43

Apprendimento Probabilistico · Apprendimento Bayesiano · Apprendimento non Supervisionato · Apprendimento di Belief Network

1 Apprendimento Probabilistico

Le forme di apprendimento viste in precedenza possono essere estese con l'uso esplicito del ragionamento probabilistico: i dati forniscono l'evidenza fattuale sulla quale si può effettuare il condizionamento. Ciò è alla base dell'apprendimento automatico supervisionato e non.

Dato il *dataset* Es , per determinare un modello ottimale, m , per i dati a disposizione, Es , si può utilizzare il *teorema di Bayes*:

$$P(m | Es) = \frac{P(Es | m) \cdot P(m)}{P(Es)}$$

dove:

- $P(Es | m)$ rappresenta la **verosimiglianza** o *likelihood* del fatto che Es sia stato generato da m : è *alta* se m fornisce una buona approssimazione della distribuzione dei dati reale (sconosciuta);
- $P(m)$ è la **probabilità a priori** che codifica il *learning bias*, indicando i modelli più verosimili a priori e può essere usata per la *regolarizzazione* inducendo a scegliere modelli *più semplici* attribuendo loro una maggiore probabilità;
- $P(Es)$, detta **funzione di partizione**, viene usata ai fini della *normalizzazione*.

Nella selezione di un singolo modello, il modello **MAP** (*maximum a posteriori*) è quello che massimizza la probabilità a posteriori, determinabile in base al numeratore

$$P(Es | m) \cdot P(m)$$

essendo trascurabile il denominatore in quanto non dipende da m . Si basa, quindi, su *likelihood* (adattamento ai dati) e distribuzione a priori dei modelli, utile come *learning bias* per codificare una preferenza verso modelli più semplici (rasoio di Occam).

In alternativa, si può puntare a determinare il modello di **massima verosimiglianza** (ML) massimizzando solo

$$P(Es | m)$$

(come per diversi modelli di apprendimento visti in precedenza che ottimizzano la log-likelihood o la log-loss).

Quando lo *spazio dei modelli* è molto ricco, sorge il *problema* di avere spesso più di un modello m per cui $P(Es | m) = 1$; ad esempio tale caso è possibile quando si adottano modelli come gli alberi di decisione: essi possono rappresentare qualunque funzione discreta, quindi anche sovradattarsi ai dati; pertanto anche un modello molto poco probabile a priori non andrebbe escluso: potrebbe essere quello corretto e, se i dati sono sufficienti, risultare anche il migliore.

Si noti che, per la regola di Bayes, se $P(m)$ è *uniforme* (caso di massima incertezza sui modelli), il modello ML coincide con il modello MAP. Altrimenti conviene preferire m più semplici, adottando distribuzioni a priori che li rendano più probabili, in quanto meno soggetti a *sovradattamento*.

2 Apprendimento Bayesiano

L'**apprendimento Bayesiano** offre un'*alternativa* alla scelta di un *singolo* modello più probabile (MAP o ML): si sfrutta la distribuzione a posteriori di diversi modelli (le ipotesi).

Dati gli esempi di training Es , per predire Y per un nuovo esempio¹ \mathbf{x} , si deve determinare la probabilità a posteriori:

$$P(Y | \mathbf{x} \wedge Es)$$

Lo scopo di un modello è quello di *generare* esempi (compito più complesso di quello di *discriminare*). Dato un insieme esaustivo M di modelli disgiunti, si può scrivere (ragionando *per casi* e sfruttando la definizione della *probabilità condizionata*):

$$\begin{aligned} P(Y | \mathbf{x} \wedge Es) &\stackrel{\text{casi}}{=} \sum_{m \in M} P(Y \wedge m | \mathbf{x} \wedge Es) \\ &\stackrel{\text{p.cond.}}{=} \sum_{m \in M} P(Y | m \wedge \mathbf{x} \wedge Es) \cdot P(m | \mathbf{x} \wedge Es) \\ &= \sum_{m \in M} P(Y | m \wedge \mathbf{x}) \cdot P(m | Es) \end{aligned}$$

Qui l'ultimo passaggio dipende da due assunzioni sul modello m :

1. m racchiude tutte le informazioni su Es necessarie a una predizione, i.e. $P(Y | m \wedge \mathbf{x} \wedge Es) = P(Y | m \wedge \mathbf{x})$;
2. m non cambia per un solo nuovo esempio, ossia $P(m | \mathbf{x} \wedge Es) = P(m | Es)$.

Invece di scegliere il modello migliore, nel **model averaging** [HTF09] si mediano le predizioni di tutti i modelli, ponderando con $P(m | Es)$:

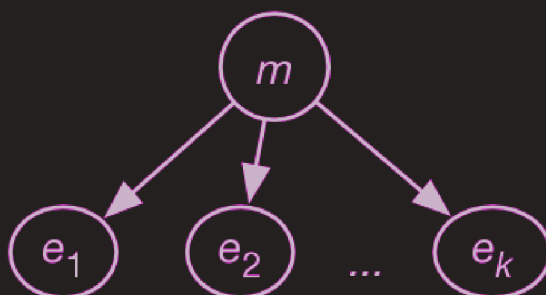
$$P(Y | \mathbf{x} \wedge Es) = \sum_{m \in M} P(Y | m \wedge \mathbf{x}) \cdot \underline{P(m | Es)}$$

dove $P(m | Es)$, calcolabile con la regola di Bayes, dipende dalla capacità di generare i dati osservati e dalla probabilità a priori di m , mentre $P(Es)$ serve solo alla normalizzazione e risulta difficile da calcolare se M contiene molti modelli, inutile se si deve solo massimizzare il numeratore.

Un'assunzione comune è che gli esempi $Es = \{e_1, \dots, e_k\}$ siano **indipendenti e identicamente distribuiti** (i.i.d.) dato m ossia che, per ogni coppia di esempi $\langle e_i, e_j \rangle$, $i \neq j$, essi siano indipendenti dato m . In tal caso si può scrivere:

$$P(Es | m) = \prod_{i=1}^k P(e_i | m)$$

il che può essere rappresentato da una semplice rete bayesiana:

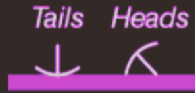


Nel ragionamento con questa rete si possono applicare metodi di inferenza esatta o approssimata, condizionando su ogni e_i osservato. Tipicamente si fanno interrogazioni su un e_i non osservato, ossia predizioni su esempi non visti oppure anche su m (distribuzione sui modelli).

2.1 Apprendere Probabilità

Un possibile *obiettivo* può essere quello di imparare le probabilità, ossia i parametri di un modello probabilistico in una delle diverse formulazioni possibili.

Esempio — Lanci di una puntina da disegno con esito: **Head** | **Tail**



Dato un certo numero di lanci *indipendenti* si avranno le osservazioni Es , nelle quali n_0 saranno **Tail** e n_1 **Head**.

Detta ϕ la probabilità di **Head**, la likelihood sarà $P(Es | \phi) = \phi^{n_1} \cdot (1 - \phi)^{n_0}$ da cui si può ottenere la log-likelihood:

$$\log P(Es | \phi) = n_1 \cdot \log \phi + n_0 \cdot \log(1 - \phi)$$

che risulta massima per $\phi = \frac{n_1}{n_0 + n_1}$, valore per cui si ha una minima log-loss.



se $n_0 = 0$ o $n_1 = 0$, una probabilità nulla per un evento resta comunque possibile: per una calcolare stima MAP serve anche una distribuzione a priori, al fine di evitare il sovradattamento.

Nell'approccio bayesiano si trattano i parametri (ϕ) non come costanti ma come variabili aleatorie a valori reali.

Si consideri il caso più semplice: variabile Y booleana e nessun input a disposizione. Per tale Y , si può considerare ϕ come variabile su $[0, 1]$ tale che $P(Y = \text{true} | \phi) = \phi$ e $P(Y = \text{false} | \phi) = 1 - \phi$. Quindi ϕ determina l'insieme M dei diversi modelli. Gli esempi in Es saranno del tipo $Y = \text{true}$ o $Y = \text{false}$.

Per $P(\phi)$ non c'è alcuna informazione pregressa (BK) su Y , oltre agli esempi in Es . In tal caso i valori sono equiprobabili ossia ϕ ha una *distribuzione uniforme* su $[0, 1]$ (nella figura che segue corrisponde alla densità per $n_0 = 0, n_1 = 0$). $P(\phi | Es)$ può essere considerata come l'aggiornamento di $P(\phi)$ dopo aver osservato gli esempi in Es :

$$P(\phi | Es) \stackrel{\text{Bayes}}{=} \frac{P(Es | \phi) \cdot P(\phi)}{P(Es)} \stackrel{i.i.d.}{\propto} \phi^{n_1} \cdot (1 - \phi)^{n_0} \cdot P(\phi)$$

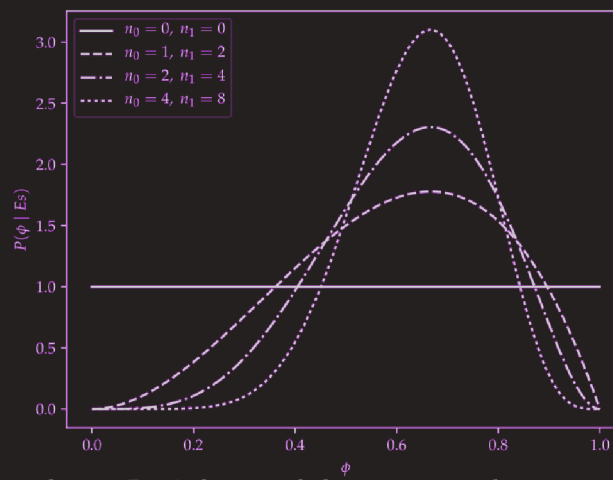
dove n_0 è il numero di esempi con $Y = \text{false}$ e n_1 quello di esempi con $Y = \text{true}$.

Sia ϕ il parametro per la distribuzione di una variabile Y binaria (ϕ probabilità di $Y = \text{true}$). Si definisce la *distribuzione Beta* nel modo seguente

$$\text{Beta}_{\alpha_0, \alpha_1}(\phi) = \frac{1}{Z} \phi^{\alpha_1 - 1} \cdot (1 - \phi)^{\alpha_0 - 1}$$

dove

- i parametri α_0 e α_1 sono conteggi iniziali, convenzionalmente si considerano $\alpha_i = n_i + 1$ $i \in \{0, 1\}$ e Z è una costante di normalizzazione: si noti che la distribuzione uniforme corrisponde a una $\text{Beta}_{1,1}$;
- la sua *moda* (valore più probabile) risulta $\phi = \frac{\alpha_1 - 1}{\alpha_0 + \alpha_1 - 2}$;
- il suo *valore atteso* è $\phi = \frac{\alpha_1}{\alpha_0 + \alpha_1}$, il che giustifica lo *smoothing di Laplace*.



Diverse densità (Beta) al variare di diversi campioni di esempi osservati

La distribuzione di $P(\phi)$ non è necessariamente uniforme. Tipicamente si ha che: $P(\phi) = \phi^{c_1-1} \cdot (1-\phi)^{c_0-1}$, come nel caso in cui si considerino c_1 *pseudo-esempi* con $Y = \text{true}$, e c_0 con $Y = \text{false}$.

Quindi, la distribuzione a posteriori è $P(\phi | Es) \propto \phi^{c_1+n_1-1} \cdot (1-\phi)^{c_0+n_0-1}$ ossia mantiene la stessa forma di quella a priori (si dice che le distribuzioni sono *conjugate*). La stima MAP per ϕ è $\hat{\phi}_{MAP} = \frac{c_1 + n_1 - 1}{c_0 + n_0 + c_1 + n_1 - 2}$ e il valore atteso è $E[\phi] = \frac{c_1 + n_1}{c_0 + n_0 + c_1 + n_1}$.

In caso si disponesse solo delle statistiche n_0 e n_1 , che possono corrispondere a sequenze diverse di esempi, si ricorre alla binomiale.

2.1.1 Variabili Categoricalhe

Nel caso di Y categorica, parametro con k valori, si considera la *distribuzione Dirichlet*, un'estensione della Beta a più dimensioni:

$$Dirichlet_{\alpha_1, \dots, \alpha_k}(p_1, \dots, p_k) = \frac{1}{Z} \prod_{j=1}^k p_j^{\alpha_j-1}$$

dove gli $\alpha_j \geq 0$ sono i conteggi iniziali dei diversi valori $j = 1, \dots, k$, e si considera in genere $\alpha_j = n_i + 1$ come un conteggio, p_j è la probabilità del j -esimo valore ($Y = y_j$) e Z è la costante di normalizzazione. Il *valore atteso* per il risultato i -esimo (mediando su tutti i p_j) sarà $\alpha_i / \sum_j \alpha_j$, con gli α_j tutti non-negativi e non tutti nulli.

Per predire un valore per Y , partendo da *pseudo-conteggi* $c_j > 0$ per ogni y_j (invece che da una distribuzione a priori uniforme) ottenuti prima di osservare dati (ossia $\forall j: n_j = 0$) considerata una Dirichlet anche quella a posteriori sarà Dirichlet e avendo osservato poi n_j esempi con $Y = y_j$, la probabilità di Y può essere stimata usando il *valore atteso*:

$$P(Y = y_j) = \frac{c_j + n_j}{\sum_h c_h + n_h}$$

Notare che con pseudo-conteggi tutti unitari si ottiene il cosiddetto *smoothing di Laplace*.

2.1.2 Opinione degli Esperti

Gli pseudo-conteggi si possono ottenere *integrando* l'opinione di esperti con i dati a disposizione. Questa integrazione comporta le seguenti problematiche:

- la difficoltà nel dare *valori esatti* di probabilità non modificabili nel seguito;
- la rappresentazione dell'incertezza d'una *stima* della probabilità;

- la *combinazione* di stime fatte da più esperti;
- l'*integrazione* tra *opinione* degli esperti e *dati* effettivi.

Una possibile *soluzione* è quella in cui le probabilità (numeri reali) provengano da conteggi (interi): invece di $P(A)$, l'esperto si limiterà a fornire $\langle n, m \rangle$, ossia il numero n occorrenze di A sulle m prove effettuate, una stima (non sempre credibile) delle *dimensioni* dei dati su cui fonda il parere.

Conteggi da diversi esperti *combinati* opportunamente² per ricavare gli pseudo-conteggi del sistema: mentre la proporzione riflette la *probabilità*, i diversi gradi di *confidenza* si riflettono nei valori assoluti.

Ad esempio ci sono diversi modi per rappresentare la probabilità di $3/7$ che corrispondono a diversi livelli di confidenza:

- con $\langle 3, 7 \rangle$ si ha una bassa confidenza: la stima sarà facilmente soppiantata da dati/stime di altri esperti;
- con $\langle 30, 70 \rangle$ si ha maggiore confidenza: pochi esempi non la modificherebbero di molto, non così se ce ne fossero decine;
- con $\langle 3000, 7000 \rangle$ si ha un'alta confidenza: anche centinaia di esempi non modificherebbero molto i conteggi a priori ricevuti dagli esperti, solo molte migliaia di dati ne ridurrebbero l'influenza.

2.2 Classificatori Probabilistici

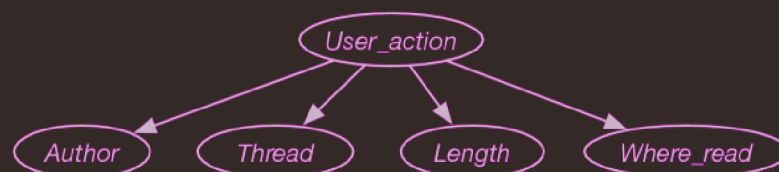
Un **classificatore Bayesiano** è un modello probabilistico di classificazione per il quale la distribuzione *a posteriori* viene calcolata mediante la regola di Bayes.

L'idea di fondo è che la *classe* influenzi / predica i valori delle feature di input per le istanze che vi appartengono: gli esempi sono raggruppati in classi perché hanno valori comuni delle feature quindi imparando le *dipendenze* delle feature dalla classe, si userà il modello risultante per predire la classificazione di nuove istanze.

Nel classificatore **Naive Bayes (NB)** l'assunzione si fa l'*indipendenza* condizionata reciproca tra le feature X_i nota la classe Y . Questo caso può essere descritto da una semplice rete bayesiana con un nodo genitore relativo alla classe e nodi figli per ciascuna delle feature di input X_i (si veda la figura nel seguente esempio). Tale rete richiede la specifica delle distribuzioni $P(Y)$ per Y e $P(X_i | Y)$ per ogni X_i . Per predire la classe per un nuovo esempio si dovrà calcolare $P(Y|X_s)$, condizionando sui suoi valori osservati delle diverse X_i .

Con più variabili-obiettivo Y_j si possono apprendere di modelli separati.

Esempio — Modello per predire l'azione-utente da un dataset precedente:



Dato un esempio con input $X_1 = v_1, \dots, X_k = v_k$, per il calcolo distribuzione a posteriori:

$$\begin{aligned}
 P(Y | X_1 = v_1, \dots, X_k = v_k) &\stackrel{\text{Bayes}}{=} \frac{P(X_1 = v_1, \dots, X_k = v_k | Y) \cdot P(Y)}{P(X_1 = v_1, \dots, X_k = v_k)} \\
 &\stackrel{\text{indip.}}{=} \frac{P(X_1 = v_1 | Y) \cdot \dots \cdot P(X_k = v_k | Y) \cdot P(Y)}{P(X_1 = v_1, \dots, X_k = v_k)} \\
 &= \frac{P(Y) \cdot \prod_{i=1}^k P(X_i = v_i | Y)}{\sum_Y P(Y \wedge (X_1 = v_1, \dots, X_k = v_k))} \\
 &\stackrel{\text{p. cong. \& indep.}}{=} \frac{P(Y) \cdot \prod_{i=1}^k P(X_i = v_i | Y)}{\sum_Y P(Y) \cdot \prod_{i=1}^k P(X_i = v_i | Y)}
 \end{aligned}$$

Si noti come il denominatore funga da costante di normalizzazione.

Il modello **NB** è capace di trattare anche esempi con **dati mancanti** ossia con valori non osservati per alcune X_i : si condiziona sulle feature osservate, facendo un'assunzione detta MAR, *Missing at Random*. Il **NB** risulta *ottimale* con X_i unica feature osservata (senza assunzioni di indipendenza): al crescere del numero di X_i osservate, l'accuratezza delle predizioni dipende dalla reciproca indipendenza delle X_i data Y . Nel caso in cui Y sia booleana e siano X_i tutte osservate si ha un modello simile a quello di *regressione logistica* (**REGLOG**). Essi fanno le stesse predizioni quando per X_i il peso è $w_i = \log [P(X_i | h)/P(X_i | \neg h)]$. La differenza sta nel fatto che **REGLOG** usa la **DG** (minimizza una log-loss) quindi ha il *punto debole* di non essere applicabile in caso di valori mancanti, mentre **NB** apprende i parametri dalla stima MAP, valida anche in caso di *valori mancanti*, ma con il *punto debole* dell'assunzione di indipendenza condizionata tra X_i data Y , spesso non giustificata.

Per apprendere un classificatore, ossia la distribuzione a posteriori $P(Y | \mathbf{x})$, serve determinare dai dati le distribuzioni $P(Y)$ e $P(X_i | Y)$ per ogni i . La stima di $P(X_i | Y)$ costituisce un *sotto-problema* di apprendimento, uno per ogni valore di Y .

Una soluzione semplice è quello data dalla *stima ML* in cui si calcola $P(X_i = x_i | Y = y)$ come *proporzione empirica* osservata nei dati di training, il rapporto tra numero dei casi in cui $X_i = x_i \wedge Y = y$ e quello dei casi in cui $Y = y$.

Esempio — Dal dataset di un esempio precedente, dato il modello **NB** in figura, si devono determinare le distribuzioni richieste come frequenze empiriche:

- $P(\text{User_action} = \text{reads}) = 9/18 = 0.5$
- $P(\text{Author} = \text{known} | \text{User_action} = \text{reads}) = 2/3$
- $P(\text{Author} = \text{known} | \text{User_action} = \text{skips}) = 2/3$
- $P(\text{Thread} = \text{new} | \text{User_action} = \text{reads}) = 7/9$
- $P(\text{Thread} = \text{new} | \text{User_action} = \text{skips}) = 1/3$
- $P(\text{Length} = \text{long} | \text{User_action} = \text{reads}) = 0$
- $P(\text{Length} = \text{long} | \text{User_action} = \text{skips}) = 7/9$
- $P(\text{Where_read} = \text{home} | \text{User_action} = \text{reads}) = 4/9$
- $P(\text{Where_read} = \text{home} | \text{User_action} = \text{skips}) = 4/9$
 - *Author* e *Where_read* non hanno potere predittivo: conoscere i loro valori non cambia la probabilità di *reads*. Quindi saranno ignorate nel seguito.

Casi di *probabilità nulle* possono avere effetti indesiderati. Alcune feature diventano *molto predittive*: con un solo valore nullo si potrebbe escludere un'intera categoria, ma un insieme finito di dati potrebbe essere insufficiente a fornire l'evidenza necessaria per supportare una tale conclusione. Alcune *combinazioni* di osservazioni possono risultare *impossibili*: nel classificatore si avrebbe una divisione per zero. Il *problema* nasce dall'uso delle frequenze empiriche come stime di probabilità. Vi si può ovviare incorporando pseudo-conteggi da scegliere accuratamente (ad esempio sfruttando il parere di esperti).

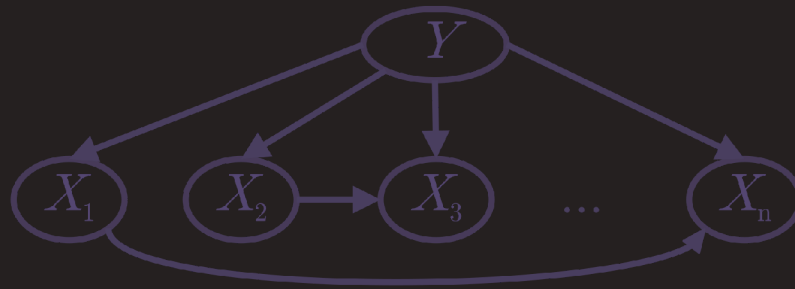
Esempio — Sistema di *help*: cerca la pagina di aiuto di interesse in base alle parole nella query Cfr. [PM23] Es. 10.5 ■

In conclusione un classificatore **NB** risulta spesso efficace, pur essendo semplice e facile da implementare quindi è un buon metodo per prototipizzare rapidamente. Funziona bene quando l'assunzione di indipendenza è appropriata nel caso in esame. Ad esempio nel caso di **tipi naturali** (spesso associati a *sostantivi*), classi che si sono evolute perché utili a distinguere oggetti.

Il modello **NB** può essere ampliato ammettendo estensioni in cui alcune X_i possano essere anche in nodi-genitori, mentre le altre rimangono figlie di Y : la probabilità della classificazione dati i genitori può essere rappresentata attraverso alberi di decisione, funzioni lineari compresse oppure reti neurali.

In altre estensioni sono ammesse figlie di Y non necessariamente indipendenti:

- **Tree Augmented Naive-Bayes (TAN) network**: oltre alla classe possono avere un solo altro genitore (purché il grafo resti aciclico), un modello semplice ed efficiente delle dipendenze tra X_i ;



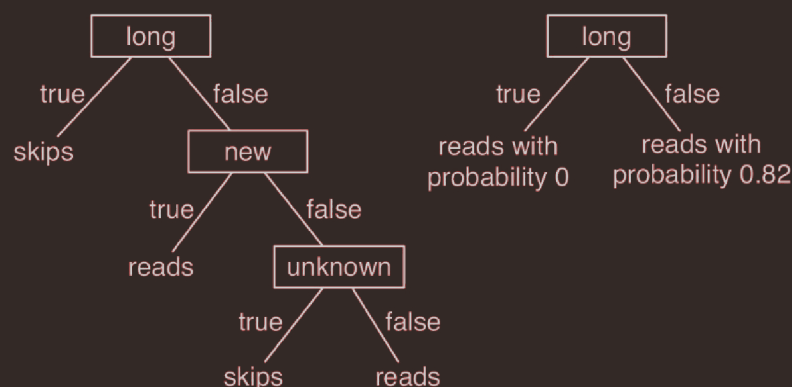
- **Latent Tree**: prevede una struttura più complessa nella Y , decomponibile in un numero di *variabili latenti* connesse ad albero, tale che ogni variabile osservata figlia di una variabile latente; si avrà così un modello della dipendenza tra variabili osservate.

2.3 Apprendimento MAP di Alberi di Decisione

Nell'apprendimento degli alberi di decisione serve un bias che tipicamente andrà a favorire alberi più piccoli e può essere imposto attraverso la scelta della distribuzione a priori (sui diversi alberi).

Più alberi potranno adattarsi perfettamente ai dati, a meno di *rumore*. Ad esempio considerando esempi di training con gli *stessi* valori delle X_i ma valori *diversi* per la Y oppure esempi con *valori mancanti* per qualche X_i , per ogni *assegnazione non osservata*, si possono adattare perfettamente al training set alberi che facciano predizioni *discordanti* su esempi non visti. In tali casi, nessuno degli alberi che si adattano perfettamente al training set potrà risultare ottimale. Questa può essere vista come conseguenza del *No Free Lunch Theorem* [Wol96] per il quale modelli apparentemente efficaci possono sempre rivelarsi peggiori su problemi/dataset diversi. Convienne quindi tenere in considerazione anche alberi che si adattino meno perfettamente ai dati, sfruttando anche la *distribuzione a priori* dei modelli (stima MAP).

Esempio — Si consideri il dataset precedente sulla lettura (training set con 9 *reads* e 9 *skips*)



- albero d_2 (a sinistra): likelihood dei dati $P(Es | d_2) = 1$, adattamento perfetto;
- albero d_0 : senza nodi interni con una sola foglia che predice sempre *reads* con probabilità $1/2$ (il più probabile fra quelli con 1 foglia, in base ai dati). Likelihood $P(Es | d_0) = (1/2)^9 \cdot (1/2)^9 \approx 0.00000149$;
- albero d_{1a} (a destra): likelihood $P(Es | d_{1a}) = 1^7 \cdot (9/11)^9 \cdot (2/11)^2 \approx 0.0543$.

Per il migliore si deve tener conto della *distribuzione a priori*.

2.4 Lunghezza delle Descrizioni

In base al criterio MAP, si deve massimizzare $P(Es | m) \cdot P(m)$. Passando ai logaritmi e negando, la funzione da ottimizzare (minimizzare) risulta:

$$(-\log_2 P(Es | m)) + (-\log_2 P(m))$$

Questo obiettivo trova corrispondenza con la *Teoria dell'informazione*, in particolare con il criterio della **minima lunghezza di descrizione**³ (*minimum description length*, MDL): vanno preferiti i modelli che minimizzano la somma precedente dove, $-\log_2 P(Es | m)$ rappresenta il numero di bit per descrivere i dati Es , in base a m , e $-\log_2 P(m)$ è il numero di bit per descrivere m . Come si vede, il *bit* è un'unità di misura che accomuna probabilità e complessità del modello.

Si può interpretare il modello ottimale per il *criterio MDL* come il più efficiente per la codifica e trasmissione dei dati: prima va comunicato il modello m e poi anche i dati, in termini di m . Essendo la funzione *log* monotona, il modello MAP *coincide* con quello MDL.

Esempio — (precedente): la probabilità a priori degli alberi non è specificata.

Per il principio MDL basato sul numero di bit che servono a descrivere un albero conviene siano più probabili gli alberi con descrizione *più corta*. Esiste una relazione *biunivoca* fra alberi e codifica usata per descriverli.

La definizione esatta di una codifica ottimale risulta in genere *difficile*: spesso si ricorre a una *misura approssimata* della complessità del modello.

Si possono considerare:

- $|t|$, il numero di *parametri probabilistici*:
 - per un albero (con probabilità alle foglie): $|t|$ = numero delle foglie;
 - per una funzione lineare o una rete neurale: $|t|$ = numero di parametri numerici;
- $|Es|$, numero di esempi (a partire da 0):
 - ci sono al più $|Es| + 1$ diverse probabilità da distinguere nel modello quindi servono $\log_2(|Es| + 1) \approx \log_2(|Es|)$ bit.

Come approssimazione del MDL da minimizzare si può usare:

$$-\log_2 P(Es | m) + |m| \cdot \log_2(|Es|)$$

detto indice **BIC** (*Bayesian information criterion*⁴).

3 Apprendimento non Supervisionato

Nell'**apprendimento non supervisionato** il training set non contiene la feature-target. Si cercano o costruiscono pattern nei dati.

Ci sono diversi compiti di tipo non supervisionato. Un *obiettivo* può essere quello di (ri)costruire una classificazione naturale dei dati. Ciò significa che la feature-obiettivo può essere considerata come variabile *latente* (i suoi valori non sono osservati).

Il metodo generale per ricostruire i valori della classificazione naturale è il **clustering**. Si partizionano gli esempi in **cluster**, le *classi* naturali, in modo tale che ogni cluster predica i valori delle feature per gli esempi in esso contenuti. Per ogni partizionamento dei dati (detto anche *clustering* o sistema di cluster) si può calcolare un *errore di predizione* associato, quindi si deve trovare il clustering migliore minimizzando tale errore.

Esempi

- Nella *diagnosi medica* un raggruppamento dei trattamenti dovrebbe consentire di predire gli effetti desiderabili e indesiderabili di ciascun trattamento; si potrebbe poi arrivare a non dare un farmaco a un paziente poiché farmaci simili potrebbero avere effetti disastrosi su pazienti simili.
- In un sistema di *tutoring intelligente* si raggruppano dati sui comportamenti degli studenti in fase di apprendimento. Strategie d'insegnamento che funzionano per uno dei membri di un cluster potrebbero essere indicate per gli altri membri.

Si distinguono l'**hard clustering** in cui ogni esempio viene assegnato a un cluster preciso (per cui il cluster può essere usato per *predire* i valori delle feature dell'esempio) e il **soft clustering** che calcola una distribuzione di appartenenza degli esempi ai cluster: la predizione dei valori delle feature di un esempio si può ottenere come *media ponderata* delle predizioni per cluster, pesata attraverso la distribuzione trovata.

3.1 K-MEANS

K-MEANS è un algoritmo per l'hard clustering. Prende in *input* gli esempi di training e il numero di cluster k da cercare. Si assumerà che il dominio di ogni feature sia *cardinale/numerico*, ossia con differenze tra valori significative. Esso produrrà come *output* i k cluster, ovvero l'assegnazione degli esempi ai cluster e, per ogni cluster, si avrà la predizione del valore (medio) per ogni feature (vettore medio, prototipo del cluster).

Dato il training set Es con X_1, \dots, X_n feature di input, si indicherà con $X_j(e)$ valore osservato della feature X_j per l'esempio $e \in Es$ (si assume che tali valori siano *osservati* direttamente). A ogni cluster è associato un indice intero $c \in \{1, \dots, k\}$.

Essenzialmente **K-MEANS** costruisce una funzione $cluster: Es \rightarrow \{1, \dots, k\}$ tale che $cluster(e) = c$ indica che e appartiene al cluster c . Per fare questo si basa su una funzione⁵ $pred(j, c)$ che restituisce il valore di X_j per il cluster c : $pred(j, c)$ sarà il valore di X_j predetto per ogni elemento del cluster c . Si noti che uno stesso valore viene predetto per ogni e del cluster: $pred(j, cluster(e))$ è il valore di X_j .

L'*obiettivo* dell'algoritmo è quindi quello di trovare $cluster$ e $pred$ ottimali, ossia che minimizzino la somma di **loss quadratiche**:

$$\sum_{e \in Es} \sum_{j=1}^n [pred(j, cluster(e)) - X_j(e)]^2$$

La *predizione* per un cluster verrà calcolata come *media* delle predizioni dei suoi esempi.

Una soluzione ottimale ottenuta da una ricerca tra le possibili assegnazioni degli esempi ai cluster che minimizzino l'errore è NP-hard, proponibile solo in caso di pochi esempi. Con molti esempi, invece, ci saranno moltissime k -partizioni possibili e una ricerca esaustiva non risulta praticabile.

L'*algoritmo* **K-MEANS** abbassa *iterativamente* la loss quadratica e può essere schematizzato come segue:

- Inizializzare (casualmente) le assegnazioni degli esempi ai cluster;

- Ripetere fino a un'assegnazione stabile:
 1. Per ogni feature X_j e ogni cluster c , definire $pred(j, c)$ con il valore medio di $X_j(e)$ per ogni e in c :

$$pred(j, c) \leftarrow \frac{\sum_{e: cluster(e)=c} X_j(e)}{|\{e : cluster(e) = c\}|}$$

- dove il denominatore conta il numero di esempi nel cluster c ;
2. Riassegnare ogni esempio e a un cluster c : scegliere c che minimizzi

$$\sum_{j=1}^n [pred(j, c) - X_j(e)]^2$$

(quadrato della distanza euclidea)

Esso *converge* quando non si hanno (sensibili) cambiamenti nelle assegnazioni.

```
procedure k-means( $Xs, Es, k$ )
```

```
  Input
```

```
     $Xs$  insieme di feature,  $Xs = \{X_1, \dots, X_n\}$ 
```

```
     $Es$  insieme di esempi di training
```

```
     $k$  numero di cluster
```

```
  Output
```

```
     $cluster$ : funzione dagli esempi ai cluster
```

```
     $pred$ : funzione dalle coppie feature-cluster ai valori delle feature per i cluster
```

```
  Local
```

```
    integer  $cc[c]$ ,  $cc\_new[c]$  // contatori per il cluster  $c$ 
```

```
    real  $fs[j, c]$ ,  $fs\_new[j, c]$  // somme valori della feature  $X_j$  per il cluster  $c$ 
```

```
    Boolean  $stable$ 
```

```
  Inizializzare  $fs$  e  $cc$  in base ai dati o casualmente
```

```
  define  $pred(j, c) = fs[j, c] / cc[c]$  // stima di  $pred(j, c)$ 
```

```
  define  $cluster(e) = \operatorname{argmin}_{c \in \{1, \dots, k\}} \sum_{j=1}^n (pred(j, c) - X_j(e))^2$ 
```

```
  repeat
```

```
    azzerare  $fs\_new$  e  $cc\_new$ 
```

```
    for each  $e \in Es$  do
```

```
       $c \leftarrow cluster(e)$ 
```

```
       $cc\_new[c] \leftarrow cc\_new[c] + 1$ 
```

```
      for each  $X_j \in Xs$  do
```

```
         $fs\_new[j, c] \leftarrow fs\_new[j, c] + X_j(e)$ 
```

```
     $stable \leftarrow (fs\_new = fs) \wedge (cc\_new = cc)$ 
```

```
     $fs \leftarrow fs\_new$ 
```

```
     $cc \leftarrow cc\_new$ 
```

```
  until  $stable$ 
```

```
  return  $cluster, pred$ 
```

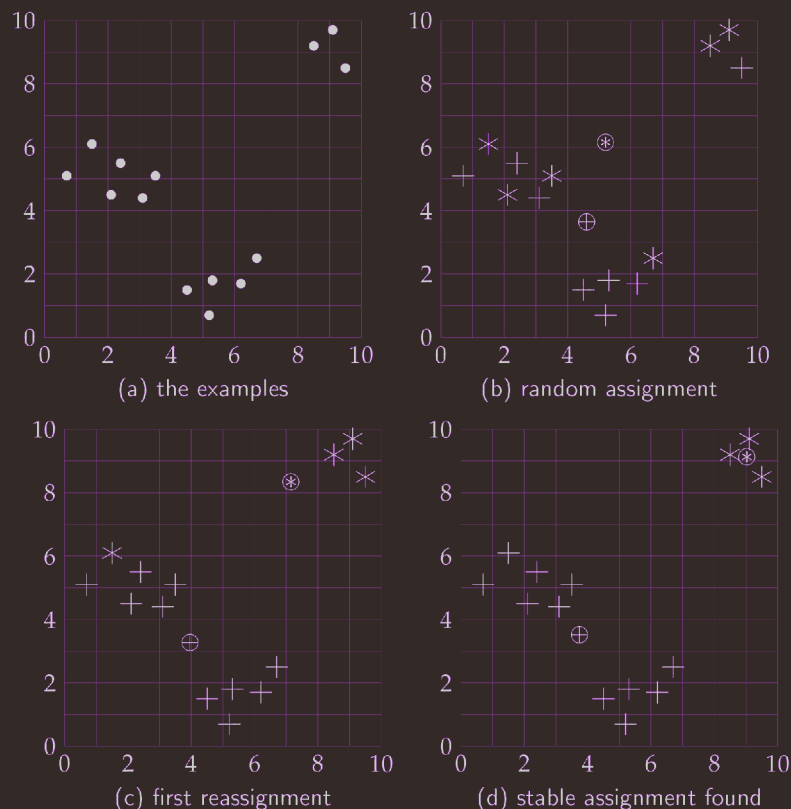
L'algoritmo converge verso un *minimo locale* stabile: l'errore tende a ridursi mentre c'è solo un numero *finito* di possibili riassegnazioni. Spesso bastano poche iterate.

Le *statistiche sufficienti* a calcolare la media (centroide) di ogni cluster (ossia i valori per ciascuna feature) sono $cc[c]$, il numero di esempi nel cluster c , e $fs[j, c]$, la frequenza del valore $X_j(e)$ per gli esempi nel cluster c ; per calcolare $pred(j, c)$, stima più recente di $pred(j, c)$ in $cluster(e)$, si usano i valori correnti di fs e cc per determinare i successivi, fs_new e cc_new .

Nell'*inizializzazione* l'assegnazione casuale degli esempi ai cluster si possono selezionare a caso k esempi come prototipi dei cluster (medie, *centroidi*) oppure si possono calcolare le statistiche sufficienti iniziali su *pochi esempi*, evitando così un passaggio sull'intero dataset.

Si raggiunge la *stabilità* del partizionamento quando le assegnazioni di *cluster*, e quindi *fs* e *cc*, non cambiano: ciò richiede che *argmin* si comporti in modo *coerente* per ogni esempio nei casi in cui più di un cluster minimizzi l'errore.

Esempio — Dato un dataset con feature $\langle X_1, X_2 \rangle$ per $k = 2$ cluster, i diversi passi sono indicati in figura [PM23]



Sono possibili diverse assegnazioni iniziali quindi si possono ottenere diversi partizionamenti. Un clustering che emerge dal dataset indica che i punti più in basso ($Y < 3$) vanno in un cluster, i restanti nell'altro.

Usando $k = 3$ si dovrebbero separare i dati in un cluster in alto a destra, uno a sinistra verso il centro e uno inferiore. Ma si possono raggiungere altre assegnazioni stabili. Ad esempio un partizionamento con i 3 punti più in alto in due cluster diversi e gli altri in un cluster diverso. È anche possibile che un cluster non contenga esempi.

Esempi — Altre demo per visualizzare l'algoritmo **k-MEANS**: A, B.

Ottenute diverse assegnazioni stabili da più esecuzioni si può scegliere il partizionamento migliore confrontandoli in termini dell'errore, ossia della distanza dai centroidi dei cluster assegnati ai diversi esempi. Si può pensare all'applicazione del **RANDOM RESTART** all'algoritmo con diverse più assegnazioni iniziali, per poi scegliere l'assegnazione stabile con minimo errore. L'etichettatura precisa è poco importante: qualsiasi permutazione andrà bene.

È importante notare la sensibilità dell'errore (ossia della distanza euclidea) alle diverse *scale* delle feature (ad esempio, feature come *altezza*, *età* e *genere*). Pertanto conviene lavorare su valori preventivamente *riscalati* (su scala comune, ad esempio standardizzati) per poter essere poi confrontati.

È importante anche la scelta del(l'iper-)parametro k che dovrebbe rappresentare un compromesso fra complessità e adattamento ai dati. È possibile usare il *BIC* score, la *ricerca del gomito/elbow* (o *ginocchio/knee*) nella *curva* dell'errore al variare dei diversi valori di k : curva decrescente al crescere di k . Il valore *naturale* per k sarebbe quello con maggiore *riduzione* dell'errore rispetto a $k - 1$. Si noti che il clustering ottimale per k può spesso risultare molto diverso da quello ottimale per $k + 1$.

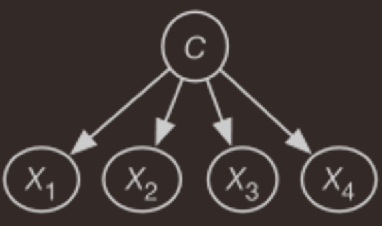
3.2 EXPECTATION MAXIMIZATION e Soft Clustering

I classificatori bayesiani possono costituire una base per l'*apprendimento non-supervisionato* se si considera la classe come variabile *latente*, ossia non osservata (senza valori nel dataset).

L'algoritmo **EXPECTATION MAXIMIZATION** (**EM**) consente di apprendere modelli probabilistici con variabili latenti, quindi può essere utilizzato per problemi di *soft clustering* se combinato con un classificatore probabilistico come **NB**. Il suo schema è analogo a quello di **K-MEANS**, con gli stessi argomenti in input, l'insieme di esempi E_s e il numero di cluster k , ma gli esempi saranno assegnati ai cluster in senso probabilistico.

Dato E_s , **EM** costruisce un modello **NB** con uno nodo genitore con la variabile *latente* C che indicherà la classe/cluster, quindi con *dominio* $\{1, \dots, k\}$, e come nodi figli uno per ogni feature X_i osservata nei dati in E_s . Si devono apprendere le distribuzioni $P(C)$, $P(X_i|C)$ che si adattino meglio a E_s .

Esempio — Problema con 4 feature binarie:

Dati				Modello	Probabilità
X_1	X_2	X_3	X_4		$P(C)$ $P(X_1 C)$ $\rightarrow P(X_2 C)$ $P(X_3 C)$ $P(X_4 C)$
t	f	t	t		
f	t	t	f		
f	f	t	t		
...		

Rispetto a **K-MEANS** si può considerare l'estensione_ (*augmentation*) della tabella-dati con due colonne (virtuali): una colonna C per la classe/cluster, ogni esempio originario replicato per k righe della tabella, una per classe/cluster; una colonna *Count* per il *contatore* normalizzato, con le k assegnazioni dell'esempio alle classi di C , che insieme rappresentano la distribuzione $P(C|e)$.

Iterativamente **EM** ricalcola i conteggi dal modello e poi il modello dai conteggi.

Esempio — Problema con $k = 3$ classi:

X_1	X_2	X_3	X_4	C	<i>Count</i>
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots
t	f	t	t	1	.4
t	f	t	t	2	.1
t	f	t	t	3	.5
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots

← E

M →

$P(C)$
 $P(X_1 | C)$
 $P(X_2 | C)$
 $P(X_3 | C)$
 $P(X_4 | C)$

Passi (simili a quelli di **K-MEANS**):

- **EXPECTATION**: aggiorna *Count* in base alle distribuzioni correnti del modello
 - per ogni esempio originario $\langle X_1 = v_1, \dots, X_n = v_n \rangle$, il contatore associato a $\langle X_1 = v_1, \dots, X_n = v_n, C = c \rangle \quad \forall c = 1, \dots, k$ viene aggiornato con $P(C = c \mid X_1 = v_1, \dots, X_n = v_n)$;
 - il *valore atteso* va calcolato tramite inferenza probabilistica;
- **MAXIMIZATION**: inferisce le probabilità del modello dagli esempi *estesi*
 - in tabella: esempi completi con valori per *tutte* le variabili, come nell'apprendimento supervisionato di probabilità (con il **NB**);
 - per la stima delle probabilità del modello si considera la *massimizzazione* ML (o MAP).

EM parte con distribuzioni o conteggi casuali e punta a convergere a un *massimo* locale della *likelihood* dei dati fornendo come output un *modello probabilistico*, utile anche a classificare nuovi esempi:

$$P(C = c \mid X_1 = v_1, \dots, X_n = v_n) = \frac{P(C = c) \cdot \prod_{i=1}^n P(X_i = v_i \mid C = c)}{\sum_{c'} P(C = c') \cdot \prod_{i=1}^n P(X_i = v_i \mid C = c')}$$

Per il calcolo delle probabilità tramite *statistiche sufficienti*

- a ogni iterata, si calcolano dai dati:
 - *cc*: *contatori-classe*, vettore *k*-dimensionale, con *cc[c]* somma dei contatori degli esempi per la classe *c* nella tabella dei dati estesa;
 - *fc*: *contatori-feature*, matrice *tridimensionale* (feature categoriche) in cui per *i* tra 1 e *n*, per ogni $v \in \text{dom}(X_i)$ e ogni classe/cluster *c*: *fc[i, v, c]* è la somma dei contatori per ogni esempio *t* per *c*;
 - le statistiche sono usate per derivare quelle dell'iterata successiva; si noti che *cc* può essere anche calcolato da *fc*, ma è più comodo tenere un contatore separato;
- la stima delle *probabilità* può essere fatta in base a *cc* e *fc*: $P(C = c) = cc[c]/|Es|$, con $|Es|$ numero di esempi nel dataset originario, o somma dei conteggi in quello esteso. Inoltre $P(X_i = v \mid C = c) = fc[i, v, c]/cc[c]$.

```

procedure EM(Xs, Es, k)
  Input
    Xs insieme di feature,  $Xs = \{X_1, \dots, X_n\}$ 
    Es insieme di esempi di training
    k numero di classi
  Output
    statistiche sufficienti per il modello probabilistico su Xs
  Local
    real cc[c], cc_new[c] // conteggi di classe/cluster vecchi e nuovi
    real fc[i, v, c], fc_new[i, v, c] // conteggi di feature vecchi e nuovi
    real dc // distribuzione per esempio e classe/cluster corrente
    Boolean stable
  repeat
    azzerare cc_new[c] e fc_new[i, v, c]
    for each  $\langle v_1, \dots, v_n \rangle \in Es$  do
      for each  $c \in [1, k]$  do
        dc  $\leftarrow P(C = c \mid X_1 = v_1, \dots, X_n = v_n)$  // dai conteggi
        cc_new[c]  $\leftarrow cc\_new[c] + dc$ 
        for each  $i \in [1, n]$  do
          fc_new[i, vi, c]  $\leftarrow fc\_new[i, v_i, c] + dc$ 
        stable  $\leftarrow (cc \approx cc\_new) \wedge (fc \approx fc\_new)$ 
        cc  $\leftarrow cc\_new$ 
        fc  $\leftarrow fc\_new$ 
  until stable
  return cc, fc

```

Si osservi che per valutare *dc*, ossia $P(C = c \mid X_1 = v_1, \dots, X_n = v_n)$, ci si basa su *cc* e *fc*.

Per l'inizializzazione (non indicata) si può partire con una *distribuzione casuale* (alla prima iterata) poi si calcolano i conteggi dai dati. In alternativa si possono considerare conteggi casuali *prima* di vedere i dati.

Si ottiene la *convergenza* quando *cc* e *fc* cambiano poco in una data iterata. La *soglia* può essere regolata trovando un compromesso tra accuratezza e velocità. In alternativa si può considerare un *numero fissato* di iterazioni.

Esempio — Nel problema precedente, per classificare $\langle x_1, \neg x_2, x_3, x_4 \rangle$, si calcola (normalizzando) la distribuzione:

$$\begin{aligned} & P(C = c \mid x_1 \wedge \neg x_2 \wedge x_3 \wedge x_4) \\ & \propto [P(X_1 = 1 \mid C = c) \cdot P(X_2 = 0 \mid C = c) \\ & \quad \cdot P(X_3 = 1 \mid C = c) \cdot P(X_4 = 1 \mid C = c)] \\ & \quad \cdot P(C = c) \\ & = \frac{fc[1, 1, c]}{cc[c]} \cdot \frac{fc[2, 0, c]}{cc[c]} \cdot \frac{fc[3, 1, c]}{cc[c]} \cdot \frac{fc[4, 1, c]}{cc[c]} \cdot \frac{cc[c]}{|Es|} \\ & \propto \frac{fc[1, 1, c] \cdot fc[2, 0, c] \cdot fc[3, 1, c] \cdot fc[4, 1, c]}{(cc[c])^3} \end{aligned}$$

Supponendo che i valori siano: 0.4 ($C = 1$), 0.1 ($C = 2$) e 0.5 ($C = 3$) si avrà che:

- *cc_new*[1] viene incrementato di 0.4, *cc_new*[2] di 0.1, ecc.
- anche *fc_new*[1, 1, 1] e *fc_new*[2, 0, 1] sono incrementati di 0.4, poi *fc_new*[1, 1, 2], *fc_new*[2, 0, 2] incrementati di 0.1, ecc.

La similarità con **K-MEANS**: il passo E assegna gli esempi alle classi (probabilisticamente) mentre il passo M (ri)determina quanto viene predetto dalle classi.

Un *problema* tipico (con $k > 1$) è che **EM** ha pressoché sempre massimi locali e globali *moltiplici*: con qualsiasi permutazione delle etichette delle classi stesse probabilità. Per trovare il massimo globale si possono effettuare più *restart* e infine scegliere il modello con log-likelihood massima.

Formulazioni alternative di **EM** si possono trovare in [McK05, DHS01, Mur12, RN20].

4 Apprendimento di Belief Network

Una BN definisce una distribuzione su un insieme di variabili. Dato che è difficile per un esperto fornire modelli accurati da codificare come rete bayesiana, si può cercare di *imparare* la rete da dati disponibili.

La complessità di tale problema dipende dalla disponibilità di conoscenza a priori sulla struttura e sulle variabili coinvolte e dalla completezza del dataset disponibile.

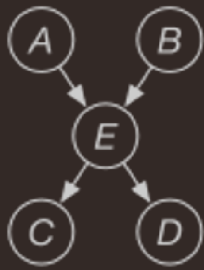
L'*obiettivo* dell'apprendimento della rete bayesiana varia dal caso *più semplice* in cui la struttura è nota e tutte le variabili note e *osservabili*, disponendo di un dataset completo, fino al caso *più difficile* in cui la struttura non è data, le variabili non siano tutte note e il dataset sia incompleto, ovvero ci siano dati mancanti (senza possibilità di fare l'assunzione che siano *missing at random*).

L'*obiettivo* si riduce a imparare, per ogni X_i , la distribuzione condizionata $P(X_i \mid \text{parents}(X_i))$. Questo si può vedere come un problema di apprendimento *supervisionato* con feature-target X_i e feature di input $\text{parents}(X_i)$.

Essendo, solitamente, i genitori in numero limitato ogni probabilità potrà essere appresa *separatamente* sfruttando gli esempi di training e altra conoscenza a priori comunicata dagli esperti, come ad esempio pseudo-conteggi (cfr. sezione dedicata) che migliorino la stima delle probabilità.

Esempio — Tipica istanza del caso semplice (variabili booleane):

Modello



Dati

A	B	C	D	E
t	f	t	t	f
f	t	t	t	t
t	t	f	t	f
...

Probabilità

$P(A)$
 $P(B)$
 $\rightarrow P(E | A, B)$
 $P(C | E)$
 $P(D | E)$

Ad esempio, nel determinare $P(E | A, B)$:

$$P(E = t | A = t \wedge B = f) = \frac{n_t + c_t}{n_f + n_t + c_f + c_t}$$

dove:

- n_t è il numero di casi in cui $E = t \wedge A = t \wedge B = f$ con $c_t \geq 0$ pseudo-conteggio corrispondente, noto prima di osservare i dati;
- n_f è numero di casi in cui $E = f \wedge A = t \wedge B = f$ con $c_f \geq 0$ pseudo-conteggio corrispondente.

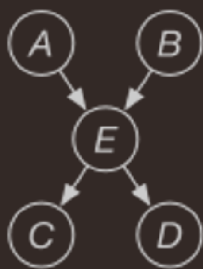
4.1 Variabili Latenti

Un caso un po' più difficile è quello in cui il modello contenga anche variabili *latenti* ossia variabili i cui valori non siano non osservabili per nessun esempio. Nella tabella dei dati non ci saranno colonne per tali variabili. Senza queste variabili nel modello aumentano i parametri per cui si hanno maggiori rischi di *sovradattamento*. Ad esempio escludendo la variabile **E** dal modello precedente occorrerebbe determinare le distribuzioni $P(A)$, $P(B)$, $P(C | A, B)$, $P(D | A, B, C)$, passando quindi da 10 a 14 parametri.

Esercizio — Spiegare il numero dei parametri nei diversi problemi citati

Esempio — BN con **E** variabile *nascosta*:

Modello



Dati

A	B	C	D
t	f	t	t
f	t	t	t
t	t	f	t
...

Probabilità

$P(A)$
 $P(B)$
 $\rightarrow P(E | A, B)$
 $P(C | E)$
 $P(D | E)$

L'obiettivo sarà quello di apprendere i parametri del modello, ossia 10 valori di probabilità nelle diverse distribuzioni.

Per imparare le probabilità di BN con variabili latenti si può usare **EM**:

- nel *passo E*, tramite inferenza probabilistica, per ogni esempio, si deriva la distribuzione delle variabili latenti, date quelle osservate;
- nel *passo M*, si deriva le probabilità del modello dai dati estesi, come visto in precedenza (si noti che, in tal caso, i conteggi non sono necessariamente valori interi).

Esempio — (cont.)

<i>A</i>	<i>B</i>	<i>C</i>	<i>D</i>	<i>E</i>	cont.		Probabilità
⋮	⋮	⋮	⋮	⋮	⋮		$P(A)$
t	f	t	t	t	.71	← E	$P(B)$
t	f	t	t	f	.29		$P(E A, B)$
f	f	t	t	f	4.2	M →	$P(C E)$
⋮	⋮	⋮	⋮	⋮	⋮		$P(D E)$
f	t	t	t	f	2.3		

- **E** calcola $P(E | A, B, C, D)$ per ogni esempio;
- **M** impara le probabilità dai dati completi.

4.2 Dati Mancanti

Oltre al caso delle variabili latenti, i dati potrebbero risultare *incompleti* per altre cause. Ci potrebbero essere *valori mancanti* per alcune variabili in qualche tupla. Serve cautela nel trattamento di tali esempi: la mancanza potrebbe essere non essere casuale ma risultare *correlata* con il fenomeno d'interesse.

Nel caso più semplice vi sono valori mancanti per variabili categoriche senza genitori da interrogare. Per poter interrogare altre variabili, la distribuzione potrebbe essere modellata aggiungendo il valore **missing** al dominio delle variabili con valori mancanti oppure considerando una variabile indicatrice binaria per ogni valore del dominio della variabile: con **0** a indicare un valore mancante nelle variabili indicatrici.

Esempio — Fornitura volontaria di dati personali.

Se non si hanno sorelle o fratelli, si tende a non dichiararlo. Inoltre si tende a non menzionare problemi di cui non si soffre, ad esempio l'ansia. È più facile che si indichino altre problematiche di cui si soffre.

Si pensi a un modello di regressione logistica riguardante la *paura di isolamento* (variabile-target **F**) in dipendenza del fatto di avere sorelle o fratelli (variabile **S** booleana) e dell'essere ansiosi ansietà (variabile **A** booleana):

$$P(F) = \text{sigmoid}(w_0 + w_1 \cdot S_t + w_2 \cdot S_f + w_3 \cdot A_t + w_4 \cdot A_f)$$

dove S_t, S_f, A_t, A_f sono tutte variabili indicatrici (binarie) con $S_t = 1$ se **S** è vera, $S_f = 1$ se **S** è falsa, ed entrambe nulle se il valore è mancante (analogamente per A_t e A_f). Il bias w_0 rappresenterebbe il valore per il caso di valore mancante per entrambe le variabili **S** e **A**.

I casi di dati mancanti non vanno ignorati. Si può ricorrere ai *modelli causali* (non in programma).

Esempio — Test di una *terapia* per una malattia: si supponga che la terapia sia inefficace e che, peggio ancora, faccia aggravare i pazienti affetti dalla malattia.

Scegliendo a caso i pazienti da trattare: i più gravi potrebbero essere esclusi molto presto dai test clinici perché si aggraverebbero troppo per rimanere sotto trattamento. Quindi ci sarebbero *malati trattati* esclusi ad una velocità più alta dei *malati non trattati*. Se si decidesse di ignorare pazienti con dati mancanti considerando solo gli altri, si potrebbe concludere *erroneamente* che la terapia funzioni essendoci meno malati tra i pazienti trattati rimasti sotto test.

4.3 Apprendimento della Struttura

Si supponga di conoscere le variabili della rete da costruire e che nessuna sia latente ma non si conosca la *struttura*, disponendo però di un dataset completo.

In tal caso gli approcci principali all'**apprendimento della struttura** sono i seguenti:

- si definisce la struttura in termini di *indipendenza condizionata*: dato un *ordinamento totale* delle variabili, occorre determinare i sottoinsiemi *parents*(\cdot) di predecessori in base a relazioni di *indipendenza condizionata* quindi occorre prima determinare il *miglior* ordinamento totale e in seguito verificare l'effettiva indipendenza, il che risulta difficile a causa della limitatezza dei dati disponibili.
- si conduce una ricerca basata su una misura di *valutazione* (*score*) delle diverse reti con l'obiettivo di trovare una struttura che minimizzi l'errore, ad esempio usando una misura basata sul criterio MAP, quindi facendo un compromesso tra adattamento ai dati e complessità.

Assumendo la disponibilità di un dataset di esempi Es completo, si punta a trovare un modello m che massimizzi:

$$P(m \mid Es) \propto P(Es \mid m) \cdot P(m)$$

con $P(Es \mid m)$ calcolabile come prodotto delle probabilità per ogni esempio, ulteriormente fattorizzabili in ragione delle distribuzioni di ogni X_i dati i genitori $par(X_i, m)$ in m :

$$\begin{aligned} P(Es \mid m) \cdot P(m) &= \left(\prod_{e \in Es} P(e \mid m) \right) \cdot P(m) \\ &= \left(\prod_{e \in Es} \prod_{X_i} P_m^e(X_i \mid par(X_i, m)) \right) \cdot P(m) \end{aligned}$$

dove $P_m^e(\cdot)$ distribuzione relativa a e secondo m .

Passando ai logaritmi (stessi punti di massimo) si può scrivere:

$$\log P(Es \mid m) + \log P(m) =$$

$$\begin{aligned} &= \left(\sum_{e \in Es} \sum_{X_i} \log P_m^e(X_i \mid par(X_i, m)) \right) + \log P(m) \\ &\stackrel{(*)}{=} \left(\sum_{e \in Es} \sum_{X_i} \log P_m^e(X_i \mid par(X_i, m)) \right) + \sum_{X_i} \log P(m(X_i)) \\ &= \sum_{X_i} \left(\sum_{e \in Es} \log P_m^e(X_i \mid par(X_i, m)) \right) + \sum_{X_i} \log P(m(X_i)) \\ &= \sum_{X_i} \left(\sum_{e \in Es} \log P_m^e(X_i \mid par(X_i, m)) + \log P(m(X_i)) \right) \end{aligned}$$

dove in $(*)$ si assume che si possa fattorizzare $P(m) = \prod_i P(m(X_i))$ con $m(X_i)$ *modello locale* per X_i . L'obiettivo finale diventa:

$$\sum_{X_i} \left(\sum_{e \in Es} \log P_m^e(X_i \mid par(X_i, m)) + \log P(m(X_i)) \right)$$

in cui ogni variabile può essere ottimizzata separatamente purché si mantenga la rete aciclica: va trovato un buon ordinamento totale sulle variabili tramite ricerca *locale* o *branch-and-bound*. Risultano diversi *problemi supervisionati* indipendenti da risolvere determinando per ogni variabile le distribuzioni $P_m^e(X_i \mid par(X_i, m))$ e una possibile approssimazione di $\log P(m(X_i))$, ad esempio usando il BIC.

4.4 Caso Generale

Se la struttura non è data, alcune variabili sono latenti (potrebbero anche essere state trascurate altre variabili necessarie) e vi sono anche dati mancanti ci sono diversi problemi da risolvere: oltre a quello dei dati *mancanti* c'è la *complessità* computazionale data dalla ricerca in uno spazio circoscritto ma troppo esteso per pensare di testare tutte le combinazioni possibili di ordinamenti e variabili latenti, anche laddove si considerino solo variabili latenti che semplifichino il modello. Lo *spazio* sarà finito ma è *vastissimo*.

Possibili *soluzioni* a tali problemi sono la selezione del miglior modello, ad esempio adottando il criterio MAP oppure effettuare predizioni *mediando* (*averaging*) su tutti i modelli (*approccio bayesiano* [HTF09]). Questa seconda soluzione fornisce in genere predizioni migliori ma risulta più difficile da spiegare / giustificare.

Riferimenti Bibliografici

- [DHS01] R. O. Duda, P. E. Hart and D. G. Stork: *Pattern classification*. 2nd edn., Wiley-Interscience (2001)
- [HTF09] T. Hastie, R. Tibshirani, J. Friedman: *The Elements of Statistical Learning* Springer. 2nd ed. 2009 (online)
- [McK05] D.J.C. MacKay: *Information Theory, Inference, and Learning Algorithms* Cambridge University Press. 2005 (online)
- [Mur12] K.P. Murphy: *Machine Learning: A Probabilistic Perspective*. MIT Press. 2012
- [PM23] D. Poole, A. Mackworth: *Artificial Intelligence: Foundations of Computational Agents*. Cambridge University Press. 3a ed. 2023
- [RN20] S.J. Russell, P. Norvig: *Artificial Intelligence*. Pearson. 4th Ed. 2020
- [Wol96] D. H. Wolpert: *The lack of a priori distinctions between learning algorithms*. Neural Computation, 8(7):1341–1390. (1996) doi

Link

- [ML_guide] Machine Learning — The Complete Guide: su wikipedia
- [ML_playground] Simulazioni di vari algoritmi ML: <https://ml-playground.com>

Note

- ¹ Qui indicato con \mathbf{x} , i.e. come tupla di valori delle feature di input, $\mathbf{X} = \mathbf{x}$, anziché un solo come in [PM23]
- ² Altre maniere per combinare evidenze (credenze) nella Dempster-Shafer Theory, estensione della teoria Bayesiana sulle probabilità soggettive.
- ³ MDL su Wikipedia, Scholarpedia.
- ⁴ BIC su it.wikipedia. Si veda anche AIC [HTF09].
- ⁵ La tupla dei valori della classe per ogni feature $\hat{\mathbf{X}}(c) = (\hat{X}_1(c), \hat{X}_2(c), \dots, \hat{X}_n(c))$ viene detta **prototipo** della classe (cfr. *centroide*, *medoide*) [DHS01, Mur12, RN20].