(F94) Metodi statistici per l'apprendimento

Predittori ad albero

Docente: *Nicolò Cesa-Bianchi* versione 6 marzo 2017

Come accennato nella lezione introduttiva, mentre alcuni tipi di dati (come immagini e documenti) si prestano bene ad essere rappresentati da vettori $\boldsymbol{x} \in \mathbb{R}^d$, altri invece (come cartelle mediche e dati strutturati in generale) non si prestano affatto. Consideriamo infatti un problema di diagnosi medica in cui i dati sono costituiti da cartelle sanitarie contenenti i campi seguenti

$$\begin{aligned}
&\text{età} \in \{12, \dots, 90\} \\
&\text{fumatore} \in \{\text{sì}, \text{no}, \text{ex}\} \\
&\text{peso} \in [10, 120] \\
&\text{sesso} \in \{\text{M}, \text{F}\} \\
&\text{terapia} \in \{\text{antibiotici}, \text{cortisonici}, \text{nessuna}\}
\end{aligned}$$

È piuttosto chiaro che l'algoritmo k-NN non è adatto per classificare dati di questo tipo. Infatti, anche qualora volessimo rappresentare con valori numerici valori come M o F per il campo sesso, oppure sì, no, ex per il campo fumatore, misurare la distanza euclidea fra i vettori così ottenuti —come richiesto da k-NN— è poco naturale.

Al fine di poter apprendere dati i cui attributi variano in insiemi eterogenei $\mathcal{X}_1, \dots, \mathcal{X}_d$ (ovvero, insiemi per i quali non ha senso confrontare $a_i \in \mathcal{X}_i$ con $a_j \in \mathcal{X}_j$ per $i \neq j$), introduciamo una nuova famiglia di predittori: i predittori ad albero.

La struttura di un predittore ad albero è quella di un albero ordinato con radice. Ricordiamo che un albero ordinato è un albero dove i figli di ogni nodo interno sono numerati progressivamente. Quindi, se il nodo interno v ha k figli, distingueremo il figlio 1, il figlio 2 e così via fino al figlio k.

Fissiamo $\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_d$ dove $\mathcal{X}_i$ è l'insieme di valori che può assumere l'i-esimo attributo x_i. Un **predittore ad albero** $h_T : \mathcal{X} \to \mathcal{Y}$ è un predittore associato ad un albero ordinato T i cui nodi interni sono etichettati da *test* e le cui foglie sono etichettate da elementi di $\mathcal{Y}$. Un test per un nodo interno con k figli è una funzione $f : \mathcal{X}_i \to \{1, \dots, k\}$ dove i è l'indice di un attributo. Quindi f mappa ciascun valore dell'i-esimo attributo in un figlio del nodo. Per esempio, se $\mathcal{X}_i \equiv \{a, b, c, d\}$ e $k = 3$ allora f potrebbe essere definita come

$$f(x_i) = \begin{cases} 1 & \text{se } x_i = c, \\ 2 & \text{se } x_i = d, \\ 3 & \text{se } x_i \in \{a, b\}. \end{cases}$$

Un esempio con $\mathcal{X}_i = \mathbb{R}$ e $k = 3$ è il seguente

$$f(x_i) = \begin{cases} 1 & \text{se } x_i \in (-\infty, \alpha], \\ 2 & \text{se } x_i \in (\beta, +\infty), \\ 3 & \text{se } x_i \in (\alpha, \beta] \end{cases}$$

dove $\alpha < \beta$ sono valori arbitrari. Si veda la Figura 1 per un esempio.

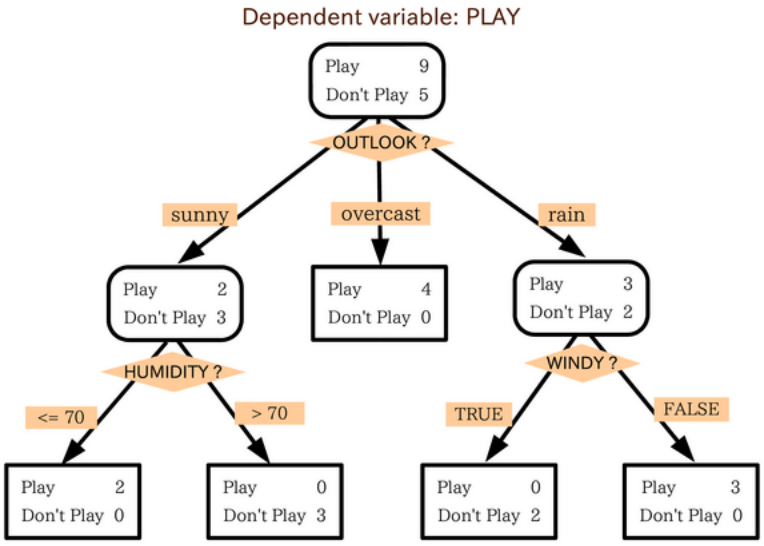

Figura 1: Un esempio classico di albero di decisione per un problema di classificazione binaria. Gli attributi sono: OUTLOOK, HUMIDITY e WINDY. Le possibili etichette sono PLAY e DON'T PLAY (con PLAY s'intende giocare a golf). All'interno dei nodi viene raffigurata la ripartizione di un insieme di 14 esempi etichettati (9 con PLAY e 5 con DON'T PLAY). Dal modo in cui questi esempi sono ripartiti nelle foglie, se ne deduce che esiste un'etichettatura delle foglie che classifica correttamente tutti i 14 esempi.

Il valore $h_T(\boldsymbol{x})$ viene calcolato nel modo seguente:

Inizio assegnando $v \leftarrow r$, dove r è la radice di T;

1. se v è una foglia ℓ, allora mi fermo assegnando a $h_T(\boldsymbol{x})$ l'etichetta $y \in \mathcal{Y}$ associata a ℓ;
2. altrimenti, se $f : \mathcal{X}_i \to \{1, \ldots, k\}$ è il test associato a v eseguo l'assegnamento $v \leftarrow v_j$ dove $j = f(x_i)$ e v_j indica il j-esimo figlio di v;
3. ritorno al passo 1.

Diciamo che ℓ è la *foglia finale di* $\boldsymbol{x}$ se terminiamo il calcolo di $h_T(\boldsymbol{x})$ nella foglia ℓ.

Come facciamo a costruire un classificatore ad albero dato un training set S? Per semplicità, consideriamo il caso di classificazione binaria $\mathcal{Y} = \{-1, +1\}$ e focalizzamoci su **alberi binari**, dove tutti i nodi interni hanno due figli. Fissato un classificatore ad albero h_T, calcoliamo per prima cosa i contributi di ciascuna foglia al training error $\widehat{\text{er}}(h_T)$. Per ogni foglia ℓ, definiamo $S_\ell \equiv \{(\boldsymbol{x}_t, y_t) \in S : \ell \text{ è la foglia finale di } \boldsymbol{x}_t\}$. Ovvero, S_ℓ è il sottoinsieme di esempi del training set che hanno ℓ come foglia finale. Definiamo inoltre $S_\ell^+ \equiv \{(\boldsymbol{x}_t, y_t) \in S_\ell : y_t = +1\}$ e $S_\ell^- \equiv \{(\boldsymbol{x}_t, y_t) \in S_\ell : y_t = -1\}$.

Per ogni foglia ℓ definiamo $N_\ell^+ = \big|S_\ell^+\big|$, $N_\ell^- = \big|S_\ell^-\big|$ e $N_\ell = \big|S_\ell\big| = N_\ell^- + N_\ell^+$. Per minimizzare il training error $\widehat{\mathrm{er}}(h_T)$ conviene assegnare a ℓ l'etichetta

$$y_\ell = \begin{cases} +1 & \text{se } N_\ell^+ \geq N_\ell^-, \\ -1 & \text{altrimenti.} \end{cases}$$

Così la foglia ℓ sbaglierà a classificare esattamente $\min\big\{N_\ell^-, N_\ell^+\big\}$ esempi in S_ℓ. Possiamo quindi scrivere il training error come somma dei contributi delle singole foglie,

$$\widehat{\mathrm{er}}(h) = \frac{1}{m} \sum_\ell \min\left\{\frac{N_\ell^-}{N_\ell}, \frac{N_\ell^+}{N_\ell}\right\} N_\ell = \frac{1}{m} \sum_\ell \psi\left(\frac{N_\ell^+}{N_\ell}\right) N_\ell$$

dove abbiamo introdotto la funzione $\psi(a) = \min\{a, 1-a\}$ definita su $[0,1]$ —si rammenti che $\big(N_\ell^+ + N_\ell^-\big)/N_\ell = 1$, e quindi l'argomento di ψ è un numero fra zero e uno.

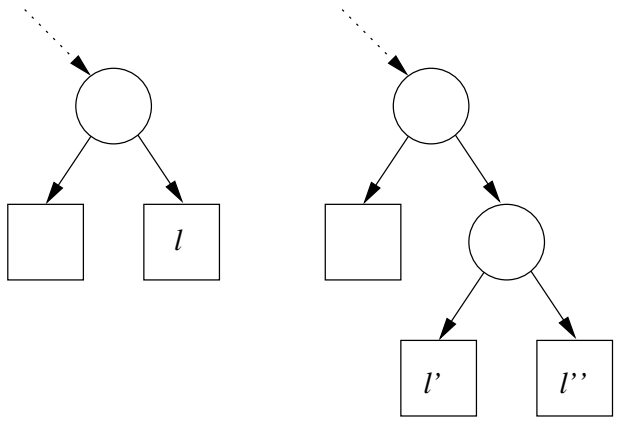

Figura 2: Un passo di costruzione dell'albero di decisione: una foglia ℓ viene sostituita da un nodo interno e da due nuove foglie ℓ' e ℓ''. Il test associato al nodo interno determina da quale delle due foglie viene classificato ciascun esempio.

Supponiamo ora di sostituire una foglia ℓ di T con un nodo interno, avente un test associato, e con due nuove foglie ℓ' e ℓ'' —si veda la Figura 2. Il training error del nuovo albero potrà essere maggiore del training error di T? Per rispondere a questa domanda, è sufficiente osservare che ψ è una funzione concava (con la concavità rivolta verso il basso, come la funzione logaritmo). Quindi vale la disuguaglianza di Jensen, che stabilisce la relazione $\psi\big(\alpha a + (1-\alpha)b\big) \geq \alpha\psi(a) + (1-\alpha)\psi(b)$, per ogni $a, b \in \mathbb{R}$ e per ogni $\alpha \in [0,1]$.

Notiamo quindi che, utilizzando la disuguaglianza di Jensen, possiamo studiare come cambia il

training error quando alla foglia ℓ vengono sostituite le due foglie ℓ' e ℓ'',

$$\begin{aligned}\underbrace{\psi\left(\frac{N_\ell^+}{N_\ell}\right)N_\ell}_{\text{contributo di }\ell} &= \psi\left(\frac{N_{\ell'}^+}{N_{\ell'}}\frac{N_{\ell'}}{N_\ell}+\frac{N_{\ell''}^+}{N_{\ell''}}\frac{N_{\ell''}}{N_\ell}\right)N_\ell \\ &\geq \psi\left(\frac{N_{\ell'}^+}{N_{\ell'}}\right)\frac{N_{\ell'}}{N_\ell}N_\ell+\psi\left(\frac{N_\ell^+}{N_{\ell''}}\right)\frac{N_{\ell''}}{N_\ell}N_\ell \\ &= \underbrace{\psi\left(\frac{N_{\ell'}^+}{N_{\ell'}}\right)N_{\ell'}}_{\text{contributo di }\ell'}+\underbrace{\psi\left(\frac{N_\ell^+}{N_{\ell''}}\right)N_{\ell''}}_{\text{contributo di }\ell''}\end{aligned}$$

cioè uno split non aumenta l'errore campionario (infatti in pratica lo diminuisce)

Descriviamo ora un metodo generico di costruzione dell'albero a partire dal training set S.

1. **Inizializzazione:** Creo T con la sola radice ℓ. Associo alla radice l'insieme $S_\ell = S$. Come etichetta della radice scelgo la maggioranza delle etichette degli esempi in S_ℓ.

2. **Loop principale:** scelgo una foglia ℓ e la trasformo in nodo interno creando due figli ℓ' (primo figlio) e ℓ'' (secondo figlio). Scelgo un attributo i e un test $f : \mathcal{X}_i \to \{1, 2\}$. Associo il test f alla ex-foglia ℓ e partiziono l'insieme S_ℓ nei due sottoinsiemi

$$S_{\ell'} = \{(\boldsymbol{x}_t, y_t) \in S_\ell \,:\, f(x_{t,i}) = 1\} \qquad \text{e} \qquad S_{\ell''} = \{(\boldsymbol{x}_t, y_t) \in S_\ell \,:\, f(x_{t,i}) = 2\} \ .$$

   Come etichetta di ℓ' scelgo la maggioranza delle etichette degli esempi in $S_{\ell'}$ e come etichetta di ℓ'' scelgo la maggioranza delle etichette degli esempi in $S_{\ell''}$.

La scelta del nodo da espandere avviene scegliendo la coppia foglia/test che approssimativamente massimizza la discesa del training error. In pratica, per valutare tale discesa non si usa la funzione $\psi(p) = \min\{p, 1-p\}$ perché può presentare dei problemi in determinate circostanze. Per esempio, si consideri uno split dove $p = \frac{N_\ell^+}{N_\ell} = 0.8$, $q = \frac{N_{\ell'}^+}{N_{\ell'}} = 0.6$, $r = \frac{N_{\ell''}^+}{N_{\ell''}} = 1$ e $\alpha = \frac{N_{\ell'}}{N_\ell} = 0.5$. In questo caso, quando $\psi(p) = \min\{p, 1-p\}$ si ha che

$$\psi(p) - \Big(\alpha\psi(q) + (1-\alpha)\psi(r)\Big) = 0.2 - \big(0.5 \times 0.4 + 0.5 \times 0\big) = 0 \ . \tag{1}$$

Ovvero, lo split non diminuisce il training error e quindi qualsiasi altro split gli sarebbe preferito. D'altra parte, questo è uno split eccellente: metà degli esempi vengono etichettati correttamente! La funzione $\psi(p) = \min\{p, 1-p\}$ non ha riconosciuto un potenziale guadagno poiché esso non riduceva il training error. Per rimediare a questo problema, si usano funzioni ψ che, come min sono simmetriche attorno a $\frac{1}{2}$ e tali che $\psi(0) = \psi(1) = 0$, ma —a differenza di min— hanno una curvatura non nulla (derivata seconda strettamente negativa), si veda la Figura 3. La curvatura serve nei casi come quello dell'esempio (1) quando p, q, r sono tutti e tre dalla stessa parte rispetto a $\frac{1}{2}$ e inoltre $p = \alpha\, q + (1-\alpha)r$. In tal caso, $\psi(p) - \big(\alpha\psi(q) + (1-\alpha)\psi(r)\big) = 0$ perché fra 0 e $\frac{1}{2}$ la funzione $\psi(a) = \min\{a, 1-a\}$ è una retta.

Alcuni esempi di tali funzioni ψ utilizzate in pratica sono

- **Funzione di Gini:** $\psi_2(p) = 2p(1-p)$.
- **Funzione entropia scalata:** $\psi_3(p) = -\frac{p}{2}\ln(p) - \frac{1-p}{2}\ln(1-p)$.
- $\psi_4(p) = \sqrt{p(1-p)}$.

Valgono le disuguaglianze $\min\{p, 1-p\} \leq \psi_2(p) \leq \psi_3(p) \leq \psi_4(p)$.

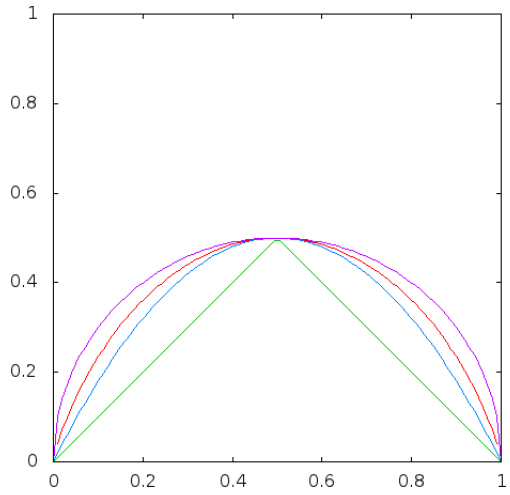

Figura 3: Grafici delle curve $\min\{p, 1-p\}$ e ψ_2, ψ_3, ψ_4.

Come i classificatori costruiti dall'algoritmo k-NN, anche gli alberi di decisione possono subire il fenomeno dell'overfitting. In questo caso, il parametro rilevante risulta essere il numero di nodi nell'albero. Consideriamo un training set affetto da rumore. Se l'albero viene fatto crescere fino ad azzerare il training error, il classificatore associato tenderà a mostrare un test error più elevato di un altro albero la cui crescita venga arrestata prima. Quindi, come in k-NN esiste il problema di scegliere il valore del parametro k in rapporto ai dati, così negli alberi di decisione esiste il problema di scegliere un numero massimo di nodi da usare per costruire l'albero.

Una caratteristica interessante dei classificatori basati su alberi di decisione è che possiamo rappresentarli come una formula di logica proposizionale in forma normale disgiuntiva (DNF). Questa si ottiene facendo la disgiunzione delle clausole (congiunzioni di predicati) ottenute dai cammini che dalla radice conducono a tutte le foglie etichettate con +1. Per esempio, il classificatore corrispondente all'albero di Figura 1 può essere rappresentato come

$$\begin{aligned}(\text{outlook} = \text{sunny}) \wedge (\text{humidity} \leq 70\%) \ \vee \ &(\text{outlook} = \text{overcast}) \\ \vee \ &(\text{outlook} = \text{rainy}) \wedge (\text{windy} = \text{false}) \ .\end{aligned}$$

Questa rappresentazione "a regole" del classificatore è molto intuitiva e si presta ad essere manipolata con gli strumenti della logica proposizionale, per esempio per ottenere formulazioni più

compatte dello stesso classificatore. Inoltre, essa fornisce una descrizione interpretabile di ciò che l'algoritmo di apprendimento ha appreso dal training set.

Per concludere, accenniamo al fatto che gli alberi di decisione possono essere utilizzati in modo naturale per risolvere problemi di classificazione multiclasse (non binaria) e di regressione. Nel primo caso, le etichette di ogni foglia ℓ saranno scelte ancora in base al maggioranza delle etichette degli esempi che hanno ℓ come foglia finale. Nel secondo caso, regressione con $\mathcal{Y} = \mathbb{R}$, il valore associato ad ogni foglia sarà la media dei valori y_t degli esempi $\boldsymbol{x}_t$ che hanno ℓ come foglia finale.