(F94) Metodi statistici per l'apprendimento

16 – Boosting

Docente: *Nicolò Cesa-Bianchi* versione 22 maggio 2017

Il boosting è un metodo incrementale per costruire classificatori della forma

$$\operatorname{sgn}\big(f(\boldsymbol{x})\big) \qquad \text{dove} \qquad f(\boldsymbol{x}) = \sum_{i=1}^{T} w_i h_i(\boldsymbol{x}) \ .$$

Qui $\boldsymbol{w}$ è un vettore di coefficienti reali e ciascun $h_i : \mathbb{R}^d \to \{-1, +1\}$ appartiene ad una qualunque famiglia $\mathcal{H}$ fissata di classificatori base.

In pratica, al fine di contenere i costi computazionali si tende a scegliere classificatori base molto semplici. Un tipico esempio per $\mathcal{H}$ è l'insieme di tutti i classificatori della forma $h_{i,\tau} : \mathbb{R}^d \to \{-1, +1\}$, con $i = 1, \dots, d$ e $\tau \in \mathbb{R}$, definiti da $h_{i,\tau}(\boldsymbol{x}) = \operatorname{sgn}(x_i - \tau)$.

In modo simile agli altri algoritmi di classificazione, il boosting minimizza in modo approssimato il training error su un training set fornito in ingresso. A differenza però di molti altri algoritmi, il boosting procede aggiungendo incrementalmente classificatori base h_i fino a soddisfare un dato criterio di stop (per esempio, quando il training error è sceso sotto un certo livello oppure quando il numero di classificatori base utilizzati supera un valore predefinito T).

Presentiamo ora un particolare algoritmo di boosting, detto **AdaBoost** (adaptive boosting). Dato lo schema delineato in precedenza, supponiamo di aver fissato un training set S con m esempi $(\boldsymbol{x}_1, y_1), \dots, (\boldsymbol{x}_m, y_m)$ e una famiglia $\mathcal{H}$ di classificatori $h : \mathbb{R}^d \to \{-1, +1\}$. Per semplicità, supponiamo anche di fissare preliminarmente il numero T di classificatori base che si intendono utilizzare nel classificatore finale f.

Ciò che ci rimane da determinare è la scelta dei coefficienti $\boldsymbol{w} \in \mathbb{R}^T$ e il metodo di scelta incrementale delle funzioni $h_1, \dots, h_T$. Deriveremo queste scelte direttamente dall'analisi dell'errore di f su S.

Ricordiamo che il training error $\widehat{\mathrm{er}}_S(f)$ di f su S è la frazione di elementi del training set che f classifica in modo sbagliato. AdaBoost minimizza un maggiorante convesso alla funzione indicatrice di errore $\mathbb{I}\{\operatorname{sgn}\big(f(\boldsymbol{x}_t)\big) \neq y_t\} = \mathbb{I}\{y_t\, f(\boldsymbol{x}_t) \leq 0\}$, ovvero

$$\widehat{\mathrm{er}}_S(f) = \frac{1}{m} \sum_{t=1}^{m} \mathbb{I}\{y_t f(\boldsymbol{x}_t) \leq 0\} \leq \frac{1}{m} \sum_{t=1}^{m} e^{-y_t f(\boldsymbol{x}_t)} = \frac{1}{m} \sum_{t=1}^{m} e^{-y_t \sum_{i=1}^{T} w_i h_i(\boldsymbol{x}_t)}$$

dove abbiamo usato la disuguaglianza elementare $\mathbb{I}\{z \leq 0\} \leq e^{-z}$ per ogni $z \in \mathbb{R}$. Questa è una tecnica alternativa ad altri metodi, come ad esempio SVM, che maggiorano la funzione indicatrice di errore con la hinge loss.

Introduciamo le funzioni $L_1, \dots, L_T$ definite come $L_i(t) = h_i(\boldsymbol{x}_t) y_t$. Si noti che $L_i(t) \in \{-1, +1\}$ e $L_i(t) = 1$ se e solo se $h_i(\boldsymbol{x}_t) = y_t$. Definiamo lo spazio campionario $\Omega = \{1, \dots, m\}$ e la funzione

di probabilità $\mathbb{P}(t) = 1/m$. Allora ciascuna $L_i : \Omega \to \{-1, 1\}$ è una variabile casuale sullo spazio di probabilità $\langle \Omega, \mathbb{P} \rangle$ e possiamo quindi scrivere

$$\widehat{\mathrm{er}}_S(f) \leq \mathbb{E}\left[e^{-\sum_{i=i}^{T} w_i L_i}\right] = \mathbb{E}\left[\prod_{i=1}^{T} e^{-w_i L_i}\right] = \frac{1}{m}\sum_{t=1}^{m}\prod_{i=1}^{T} e^{-w_i L_i(t)} \ .$$

Quello che vorremo ora fare, per proseguire l'analisi, è scrivere il valore atteso del prodotto come il prodotto dei valori attesi. Come sappiamo, questo passaggio è possibile soltanto nel caso in cui le v.c. $e^{-w_1 L_1}, \ldots, e^{-w_T L_T}$ siano indipendenti, il che è vero se e solo se le v.c. $L_1, \ldots, L_T$ sono indipendenti. Purtroppo, questa richiesta di indipendenza è in contrasto col fatto che noi vorremmo scegliere le funzioni $h_1, \ldots, h_T$ incrementalmente, ovvero ciascuna *in dipendenza* delle precedenti.

Quello che possiamo fare è cambiare lo spazio di probabilità al fine di scrivere

$$\mathbb{E}\left[\prod_{i=1}^{T} e^{-w_i L_i}\right] = \prod_{i=1}^{T} \mathbb{E}_i\left[e^{-w_i L_i}\right] \tag{1}$$

dove ciascun $\mathbb{E}_i\left[e^{-w_i L_i}\right]$ denota il valore atteso di $e^{-w_i L_i}$ rispetto a una funzione di probabilità $\mathbb{P}_i$ su Ω ancora da specificare.

Supponendo di poter eseguire il passo (1), che dimostreremo in seguito, proseguiamo con la derivazione come segue

$$\begin{aligned}
\widehat{\mathrm{er}}_S(f) &\leq \prod_{i=1}^{T} \mathbb{E}_i\left[e^{-w_i L_i}\right] \\
&= \prod_{i=1}^{T} \left(e^{-w_i}\,\mathbb{P}_i(L_i = 1) + e^{w_i}\,\mathbb{P}_i(L_i = -1)\right) \\
&= \prod_{i=1}^{T} \left(e^{-w_i}(1 - \varepsilon_i) + e^{w_i}\varepsilon_i\right) \qquad (2)
\end{aligned}$$

dove abbiamo posto

$$\varepsilon_i \stackrel{\text{def}}{=} \mathbb{P}_i(L_i = -1) = \sum_{t=1}^{m} \mathbb{I}\{L_i(t) = -1\}\mathbb{P}_i(t) \ .$$

Si noti che ε_i è l'errore di h_i rispetto alla funzione di probabilità $\mathbb{P}_i$. Ovvero, ε_i è l'errore di h_i sul training set S i cui elementi sono pesati con la funzione $\mathbb{P}_i$.

Prima di derivare le $\mathbb{P}_i$ terminiamo la costruzione dell'algoritmo **AdaBoost**. Vogliamo ora scegliere i coefficienti $w_1, \ldots, w_T$ in modo da minimizzare (2).

A questo scopo, calcoliamo gli zeri della derivata di $e^{-w}(1 - \varepsilon) + e^{w}\varepsilon$ rispetto a w. Troviamo un unico zero per

$$w = \frac{1}{2}\ln\frac{1 - \varepsilon}{\varepsilon} \ .$$

Studiando il segno della derivata, troviamo che questo zero corrisponde ad un minimo della funzione $e^{-w}(1 - \varepsilon) + e^{w}\varepsilon$. Quindi assegnamo

$$w_i \stackrel{\text{def}}{=} \frac{1}{2}\ln\frac{1 - \varepsilon_i}{\varepsilon_i} \qquad \text{per } i = 1, \ldots, T.$$

Sostituendo in (2) e semplificando otteniamo

$$\widehat{\mathrm{er}}_S(f) \leq \prod_{i=1}^{T} \sqrt{4\varepsilon_i(1-\varepsilon_i)} \ .$$

Si noti che $w_i = 0$ se e solo se $\varepsilon_i = 1/2$, cioè il peso dei punti di D_m (pesati da $\mathbb{P}_i$) dove h_i sbaglia è esattamente $1/2$. Dato che una tale h_i non influenza il valore di f (proprio perché $w_i = 0$) possiamo assumere senza perdita di generalità che $\varepsilon_i \neq 1/2$.

Per comodità poniamo $\gamma_i \stackrel{\text{def}}{=} 1/2 - \varepsilon_i$. Si noti che $\varepsilon_i \neq 1/2$ implica $\gamma_i \neq 0$. Usando la disuguaglianza $1 + x \leq e^x$, valida per ogni x, otteniamo

$$\widehat{\mathrm{er}}_S(f) \leq \prod_{i=1}^{T} \sqrt{4\varepsilon_i(1-\varepsilon_i)} = \prod_{i=1}^{T} \sqrt{1-4\gamma_i^2} \leq \prod_{i=1}^{T} e^{-2\gamma_i^2} = e^{-2\sum_{i=1}^{T}\gamma_i^2} \ .$$

Quest'ultimo risultato ci dà un maggiorante all'errore sul training set di f in funzione dell'errore di ciascuna h_i sul training set pesato da $\mathbb{P}_i$.

Per esempio, se avessimo $\left|\gamma_i\right| > \gamma > 0$ per ciascun $i = 1, \ldots, T$ allora, sotto questa ipotesi,

$$\widehat{\mathrm{er}}_S(f) \leq e^{-2T\gamma^2}$$

ovvero l'errore sul training set scende *esponenzialmente* nel numero T di classificatori base. Se questo è il caso allora, dato che $\widehat{\mathrm{er}}_S(f) = 0$ se e solo se $\widehat{\mathrm{er}}_S(f) \leq 1/m$, un numero

$$T > \frac{\ln m}{2\gamma^2}$$

di classificatori base è sufficiente per portare a zero l'errore di f sul training set.

Passiamo ora a derivare le $\mathbb{P}_i$ che soddisfano la condizione (1). Poniamo $\mathbb{P}_1 = \mathbb{P}$, $\mathbb{E}_1 = \mathbb{E}$ e

$$\mathbb{P}_{i+1}(t) \stackrel{\text{def}}{=} \frac{\mathbb{P}_i(t) e^{-w_i L_i(t)}}{\mathbb{E}_i\left[e^{-w_i L_i}\right]} \qquad \text{per } t = 1, \ldots, m \text{ e } i = 1, \ldots, T-1 \tag{3}$$

dove

$$\mathbb{E}_i\left[e^{-w_i L_i}\right] = \sum_{s=1}^{m} e^{-w_i L_i(s)} \mathbb{P}_i(s) \ .$$

È facile verificare che le $\mathbb{P}_1, \ldots, \mathbb{P}_d$ sono effettivamente delle f. di probabilità su $\Omega = \{1, \ldots, m\}$, in particolare $\mathbb{P}_i(t) > 0$ e $\mathbb{P}_i(1) + \cdots + \mathbb{P}_i(m) = 1$.

Per questa scelta delle f. di probabilità possiamo ora dimostrare la (1) come segue. Prima di tutto risolviamo la (3) per $e^{-w_i L_i(t)}$ ottenendo

$$e^{-w_i L_i(t)} = \mathbb{E}_i\left[e^{-w_i L_i}\right] \frac{\mathbb{P}_{i+1}(t)}{\mathbb{P}_i(t)}$$

quindi otteniamo

$$\begin{aligned}
\mathbb{E}\left[\prod_{i=1}^{T} e^{-w_i L_i}\right] &= \frac{1}{m}\sum_{t=1}^{m}\left(\prod_{i=1}^{T}\mathbb{E}_i\left[e^{-w_i L_i}\right]\frac{\mathbb{P}_{i+1}(t)}{\mathbb{P}_i(t)}\right) \\
&= \frac{1}{m}\sum_{t=1}^{m}\left(\frac{\mathbb{P}_{T+1}(t)}{\mathbb{P}_1(t)}\right)\left(\prod_{i=1}^{T}\mathbb{E}_i\left[e^{-w_i L_i}\right]\right) \\
&= \left(\sum_{t=1}^{m}\frac{\mathbb{P}_{T+1}(t)}{\mathbb{P}_1(t)}\mathbb{P}_1(t)\right)\left(\prod_{i=1}^{T}\mathbb{E}_i\left[e^{-w_i L_i}\right]\right) \\
&= \prod_{i=1}^{T}\mathbb{E}_i\left[e^{-w_i L_i}\right] \ .
\end{aligned}$$

Queste f. di probabilità hanno una semplice intepretazione quando si studi come $\mathbb{P}_{i+1}$ dipende da $\mathbb{P}_i$. Fissata $\mathbb{P}_i$, se $\varepsilon_i < 1/2$ allora $w_i > 0$ e quindi ciascuna $\mathbb{P}_{i+1}(t)$ è ottenuta moltiplicando $\mathbb{P}_i(t)$ per la quantità $e^{-w_i L_i(t)}$ che è maggiore di 1 se e solo se $h_i(\boldsymbol{x}_t) \neq y_t$.

In altre parole, il peso di ciascun elemento $(\boldsymbol{x}_t, y_t)$ del training set viene aumentato passando da $\mathbb{P}_i(t)$ a $\mathbb{P}_{i+1}(t)$ se e solo se h_i sbaglia su $(\boldsymbol{x}_t, y_t)$. Intuitivamente, questo processo concentra il peso su quegli elementi del training set che non sono classificati correttamente dalle ipotesi precedenti. Un ragionamento analogo può essere fatto per il caso $\varepsilon_i > 1/2$.

Siamo finalmente pronti a scrivere l'algoritmo di boosting **AdaBoost**.

Input:	Training set $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_m) \in \mathbb{R}^d \times \{-1, +1\}$.
	Famiglia $\mathcal{H}$ di classificatori base.
	Numero massimo T di classificatori base da utilizzare.
Inizializza	$\mathbb{P}_1(t) \leftarrow 1/m$ per $t = 1, \ldots, m$.

Per $i = 1, \ldots, T$

1. Scegli $h_i \in \mathcal{H}$ tale che $\varepsilon_i \neq 1/2$ (se una tale h_i non può essere trovata, allora STOP).
2. Assegna $w_i \leftarrow \frac{1}{2}\ln\frac{1-\varepsilon_i}{\varepsilon_i}$.
3. Calcola $\mathbb{P}_{i+1}$ con la formula (3).

Output: f tale che $f(\boldsymbol{x}) = \mathrm{sgn}(w_1 h_1(\boldsymbol{x}) + \cdots + w_T h_T(\boldsymbol{x}))$.

Si noti che la scelta delle h_i potrebbe essere fatta utilizzando come subroutine di **AdaBoost** un algoritmo *ad hoc* per la classe $\mathcal{H}$. Per esempio, potremmo usare un Perceptrone nel caso in cui $\mathcal{H}$ sia la famiglia dei classificatori lineari a soglia. In questo caso la scelta di h_i al passo 1. verrebbe eseguita dall'algoritmo del Perceptrone.

Da questo punto di vista, il boosting permette di "pompare" le prestazioni di un qualsiasi algoritmo A che produce un classificatore scelto da $\mathcal{H}$. In teoria, tutto ciò che l'algoritmo di boosting chiede ad A ad ogni passo i è di trovare in $\mathcal{H}$ un classificatore h_i tale che $\varepsilon_i \neq 1/2$. In pratica, se A riesce ogni volta a trovare un h_i tale che ε_i è lontano dal valore "critico" $1/2$ allora **AdaBoost** garantisce che l'errore sul training set descresce esponenzialmente.