

Modelli ad albero e combinazione di modelli

Vincenzo Bonifaci

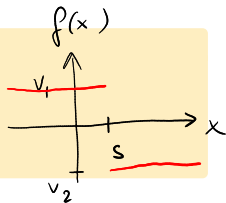
IN550 – Machine Learning

Decision stump

Uno dei modi più semplici di associare un valore o una decisione ad un input x consiste nel confrontarlo con una *soglia* \underline{s} :

 \mathbb{R}^1

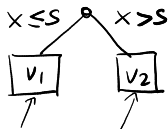
$$f(x) = \begin{cases} v_1 & x \leq \underline{s} \\ v_2 & x > \underline{s} \end{cases}$$



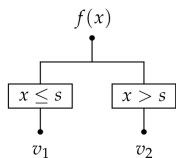
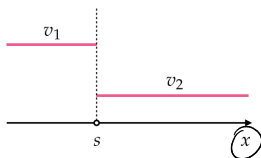
f è una funzione con 3 parametri interni (v_1, v_2, \underline{s})

f è detta *decision stump* [ceppo di decisione], in quanto rappresentabile con una struttura ramificata

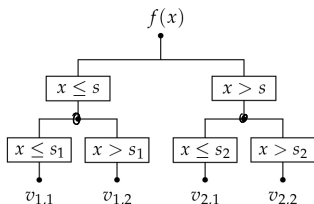
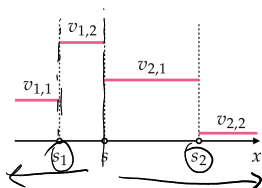
v_1 e v_2 sono i valori associati alle due *foglie* dello stump



Rappresentazioni di un decision stump



Profondità 1



Prof. 2

Rappresentazioni di un modello ad albero di profondità due

Decision stump per input d -dimensionali

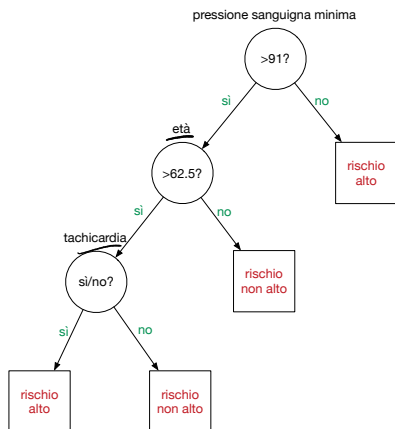
Se $x = (x_0, x_1, \dots, x_d) \in \mathbb{R}^d$, un decision stump si focalizza su una particolare componente x_k dell'input:

$$f(x) = \begin{cases} v_1 & x_k \leq s \\ v_2 & x_k > s \end{cases}$$

Applicando ricorsivamente i decision stump alle foglie, otteniamo un *albero di decisione* di profondità due, tre, ...

Esempio 1

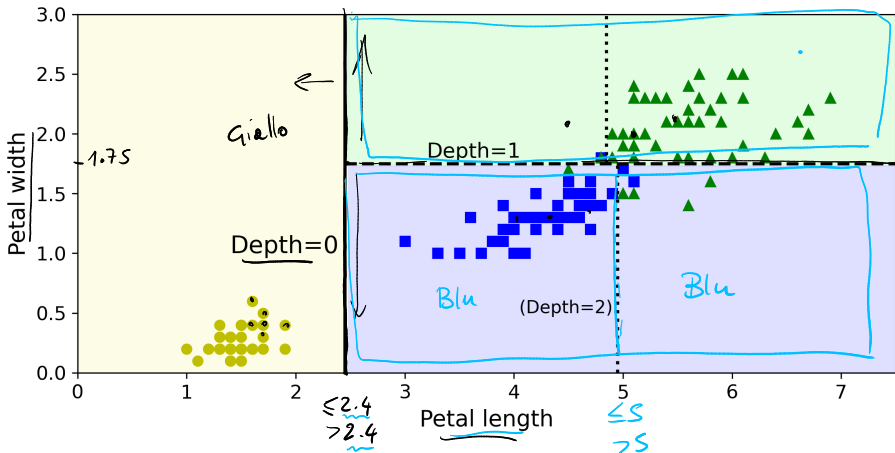
Identificazione di pazienti a rischio immediato dopo un attacco di cuore



Modelli ad albero di profondità non eccessiva sono **facili da interpretare** per gli esseri umani

Esempio 2

Iris



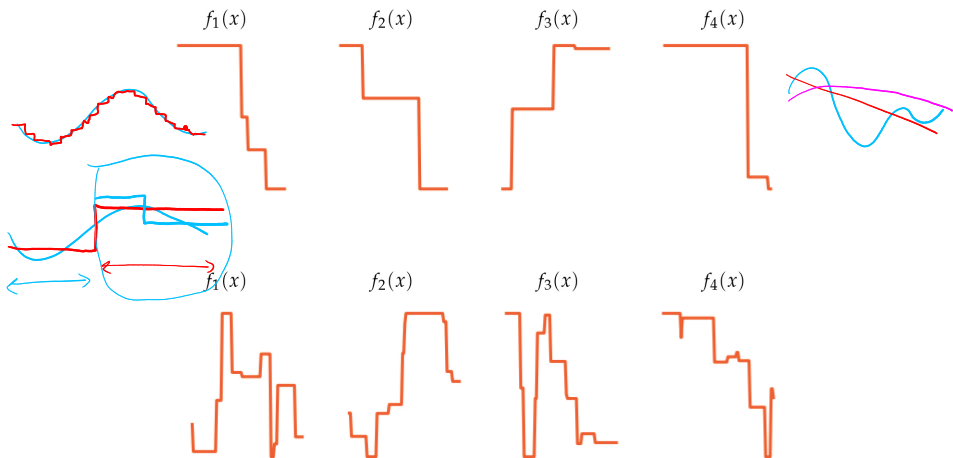
Regressione e classificazione

I modelli ad albero possono essere utilizzati per:

- Regressione (*alberi di regressione*)
- Classificazione (*alberi di classificazione*)

Chiamati anche genericamente *alberi di decisione* [*decision trees*]

Modelli ad albero come approssimatori universali

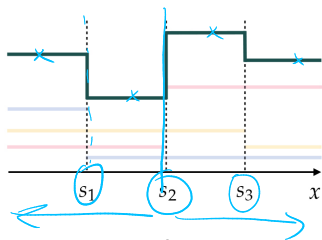
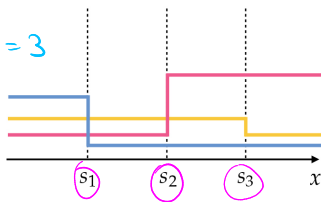


C'è una differenza con gli approssimatori a forma fissa e approssimatori neurali: le unità degli alberi di decisione sono definite in modo **locale**

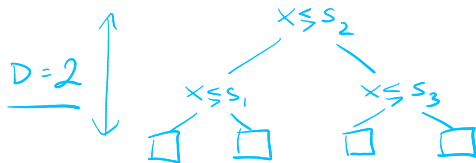
Somma di alberi

La **somma** di alberi è ancora rappresentabile con un albero, più profondo

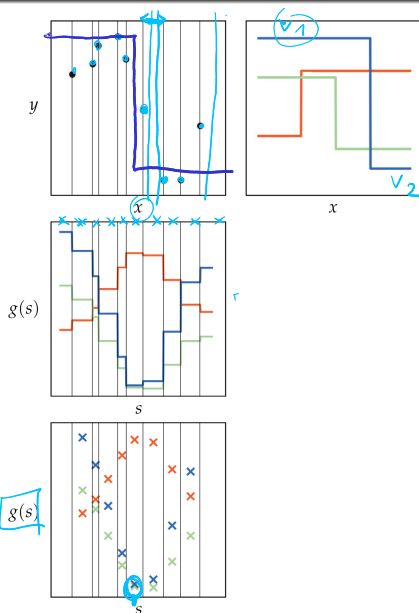
$$2^D - 1 = 3$$



In generale, $2^D - 1$ decision stump possono essere combinati in un albero binario di profondità D



Ottimizzare la soglia (a foglie fisse)

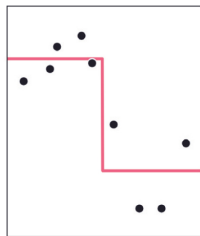
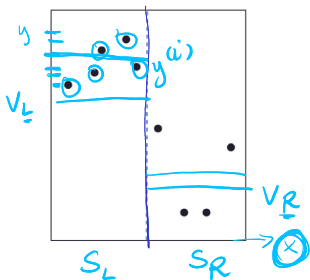


(S, v_1, v_2)

$$(v - y)^2$$

$m - 1$ valori
possibili per le
soglie s

Ottimizzare le foglie (a soglia fissa)

S insieme di esempi

$$g(v_L) = \frac{1}{|S_L|} \sum_{i \in S_L} (v_L - y^{(i)})^2$$

$$v_L^* = \frac{1}{|S_L|} \sum_{i \in S_L} y^{(i)}$$

$$g(v_R) = \frac{1}{|S_R|} \sum_{i \in S_R} (v_R - y^{(i)})^2$$

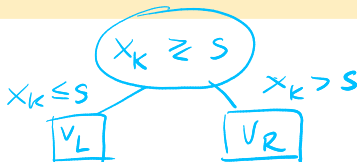
$$v_R^* = \frac{1}{|S_R|} \sum_{i \in S_R} y^{(i)}$$

Costruzione di un albero di regressione di profondità 1



Costruzione di un albero di regressione di profondità 1

- Per ogni variabile di input $x_k \in \{x_1, \dots, x_d\}$: d
 - Per ognuno degli $m-1$ possibili valori intermedi di x_k : $m-1$
 - Fissa la soglia s a quel valore intermedio
 - Ottimizza i valori v_L, v_R delle foglie (con s fissata)
 - Memorizza il rischio empirico risultante $L(x_k, s, v_L, v_R)$
- Restituisci la combinazione (k, s, v_L, v_R) che minimizza il rischio empirico





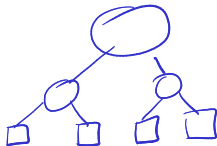
Costruzione di alberi di regressione di profondità > 1

Alberi di profondità maggiore possono essere creati attraverso uno schema di *dicotomizzazione iterativa*

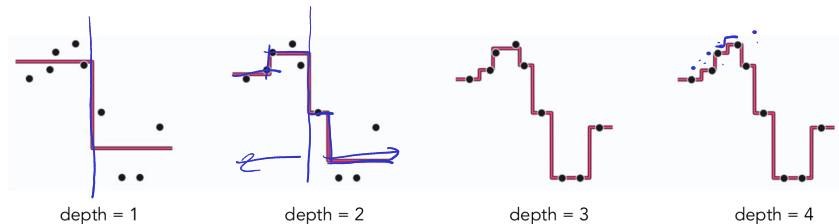
Costruzione di un albero di regressione

- Costruisci un albero di regressione di profondità 1
- Sostituisci ricorsivamente ciascuna foglia con un albero di profondità $D - 1$

Nota. Si tratta di un'euristica. Non vi è garanzia che l'albero costruito in tal modo abbia rischio empirico minimo. In effetti, trovare un albero dal rischio empirico minimo è un problema NP-arduo.



Esempio



Con una profondità sufficientemente alta, il rischio empirico sarà zero

Alberi di classificazione

- Ottimizzare le foglie (a soglia fissa):
due possibili approcci:
 - basato su una **funzione costo**
 - basato su un **voto di maggioranza** (pesato)
- Ottimizzare la soglia (a foglie fisse) e la variabile di decisione:
basato su una misura di qualità (**purezza**) delle foglie risultanti:
 - accuratezza bilanciata
 - altre misure (entropia, coefficiente di Gini)

Ottimizzare le foglie (a soglia fissa) – Approccio 1

In questo caso, l'ottimizzazione è guidata da una qualche **funzione di costo**

Esempio: con etichette ± 1 e funzione costo Softmax,

$$\underset{v_L}{\text{minimize}} \rightarrow g(v_L) = \frac{1}{|S_L|} \sum_{i \in S_L} \log(1 + \exp(-y_{\pm 1}^{(i)} v_L)) \quad \rightsquigarrow v_L^*$$

$$g(v_R) = \frac{1}{|S_R|} \sum_{i \in S_R} \log(1 + \exp(-y^{(i)} v_R)) \quad \rightsquigarrow v_R^*$$

dove S_L ed S_R sono i sottoinsiemi di esempi a “sinistra” e a “destra” della soglia

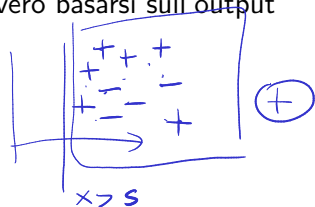
Si applica un algoritmo di ottimizzazione convessa per trovare v_L^* e v_R^* (il problema di ottimizzazione è unidimensionale)

Ottimizzare le foglie (a soglia fissa) – Approccio 2

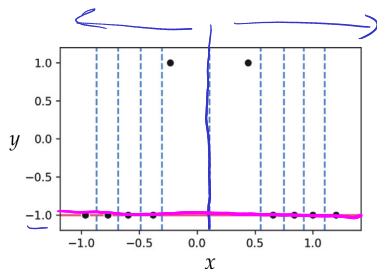
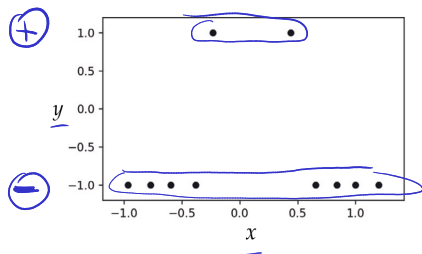
In questo caso si combinano direttamente gli output in ognuna delle due regioni (sopra e sotto la soglia)

Restituire la media ora non ha più senso, perché siamo in un problema di classificazione

La possibilità più semplice è utilizzare la **moda**, ovvero basarsi sull'output di **maggioranza** nella regione



Output a maggioranza e sbilanciamento tra classi



In questo esempio, qualunque sia la soglia, prendendo una maggioranza pura ogni decision stump sarebbe completamente piatto

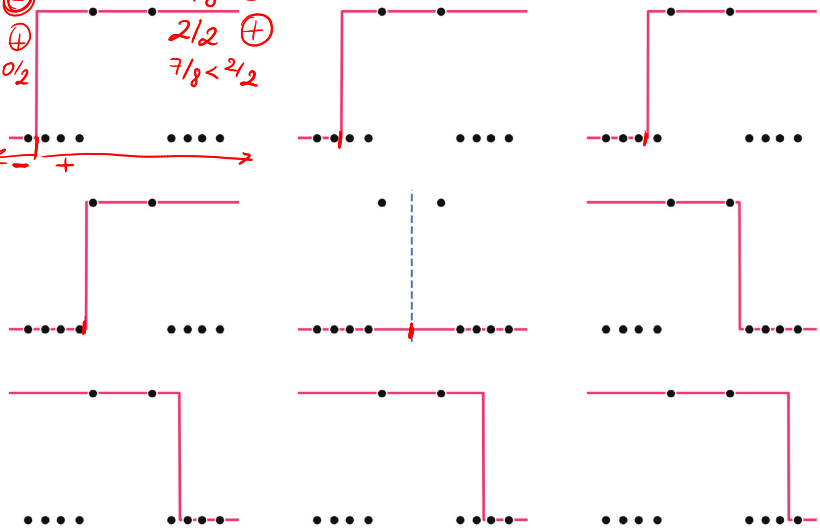
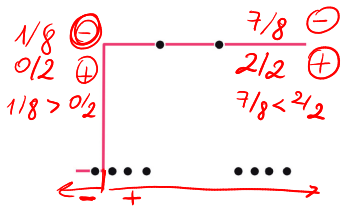
Output a maggioranza pesata



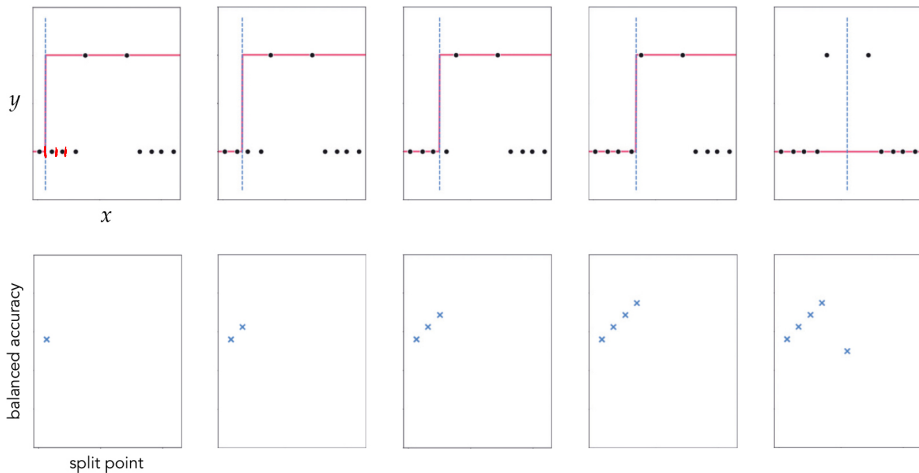
L'output a maggioranza può essere problematico se c'è sbilanciamento tra le classi, quindi può convenire calcolare una **maggioranza pesata**:

$$\operatorname{argmax}_{k \in \{1, \dots, K\}} \frac{\text{numero di esempi di classe } k \text{ della } \text{foglia}}{\text{numero di esempi di classe } k \text{ in entrambe le foglie}}$$

Output a maggioranza pesata

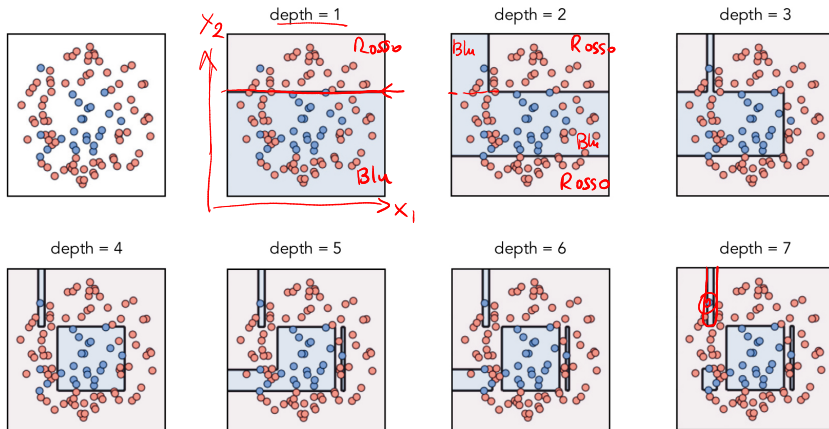


Scelta della soglia con l'accuratezza bilanciata

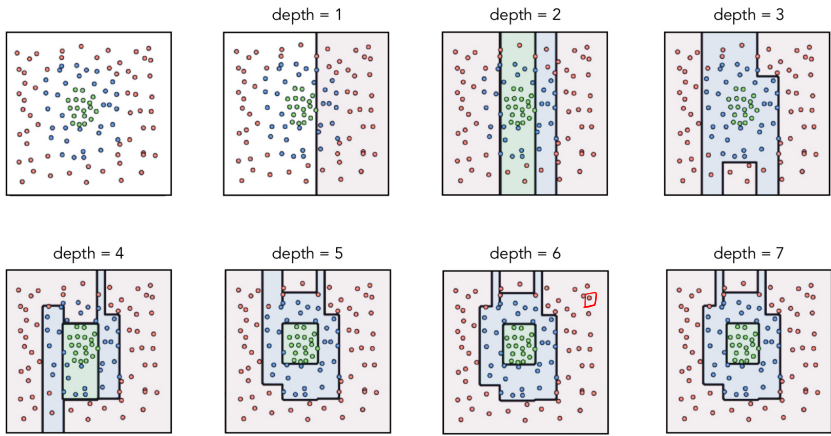


Costruzione di alberi di classificazione a profondità > 1

Si ottiene **ricorsivamente** (per dicotomizzazione iterativa) come nel caso degli alberi di regressione



Esempio con 3 classi ($K = 3$)



Vari criteri di terminazione (purezza sufficiente, albero con troppi nodi...)



Vantaggi e svantaggi dei modelli ad albero



- Applicabili a dati sia numerici che categorici
- Applicabili a regressione o classificazione (anche multiclasse)
- Alta interpretabilità
- Sono approssimatori universali
- Come tutti gli approssimatori universali, possono dare **overfitting**

Limitare l'overfitting nei modelli ad albero

Per limitare la possibilità di overfitting, si può:

- Terminare la costruzione dell'albero quando le foglie sono sufficientemente pure
- Terminare la costruzione dell'albero quando l'albero raggiunge una certa grandezza o profondità
- Costruire l'albero completamente e poi "potarlo" utilizzando un validation set (si veda ad es. *cost-complexity pruning* in `scikit-learn`)



Combinazione di modelli (*ensemble methods*)

Supponiamo di avere un certo numero di modelli (funzioni ipotesi)

$$h_1, h_2, \dots,$$

dalle prestazioni non eccelse

Possiamo **combinarli** per ottenere un modello H migliore di tutti i precedenti?

Due approcci generali:

- *Boosting*
- *Bagging*

Li discuteremo nel contesto di ceppi/alberi per la classificazione binaria

Classificatori deboli

Consideriamo un problema di classificazione binaria

$$0.5$$

$$0.5 - \gamma$$

Classificatore debole

Una funzione ipotesi $h : \mathcal{X} \rightarrow \mathcal{Y}$ è un *classificatore debole* se $\exists \gamma > 0$:

inaccuratezza

$$\Pr_{(x,y) \in \mathcal{D}} [h(x) \neq y] \leq \frac{1}{2} - \gamma$$

Un *apprendista debole* è un algoritmo di apprendimento in grado di generare classificatori deboli

Come incrementare [dare un *boost*] la qualità di un apprendista debole?

Schema del boosting

0.5
~~1~~

1.5
~~1~~

1.5
~~1~~

↓ ↓
h₁ h₂
⋮ h₃
⋮
h_T

Dato: un dataset $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(m)}, y^{(m)})$

→ ■ Inizialmente, tutti gli esempi hanno lo stesso peso

■ Ripeti per $t = 1, 2, \dots$:

■ Fornisci il dataset pesato all'apprendista debole, ottenendo un classificatore debole h_t

■ Ri-pesa gli esempi dando enfasi agli esempi misclassificati da h_t

■ Combina tutti i modelli h_t linearmente

h₁
α₁

h₂
α₂

h₃
α₃

h_T
α_T

$$H(x) = \text{sgn}(\alpha_1 h_1(x) + \alpha_2 h_2(x) + \dots + \alpha_T h_T(x))$$

Un algoritmo di boosting: AdaBoost

AdaBoost

Dato: dataset $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(m)}, y^{(m)}), y^{(i)} \in \{-1, +1\}$

1 Inizializza $D_1(i) = 1/m$ per ogni $i = 1, 2, \dots, m$

2 Per ogni $t = 1, 2, \dots, T$:

■ Fornisci (D_t) all'apprendista debole, ottieni $(h_t): \mathcal{X} \rightarrow [-1, 1]$

■ Calcola il margine di correttezza di h_t :

Misura di qualità $\rightarrow r_t = \sum_{i=1}^m D_t(i) \underbrace{y^{(i)} h_t(x^{(i)})}_{\pm 1 \times \pm 1} \in [-1, 1]$

$\rightarrow 1$ se corretto
 $\rightarrow -1$ se scorretto

$$\alpha_t = \frac{1}{2} \log \frac{1+r_t}{1-r_t}$$

\bar{e} crescente per r_t crescente

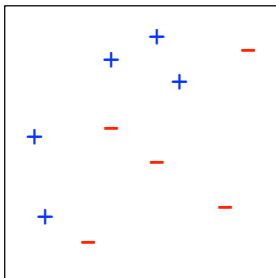
■ Aggiorna i pesi: $D_{t+1}(i) \propto D_t(i) \exp(-\alpha_t y^{(i)} h_t(x^{(i)}))$

\rightarrow se > 0
allora
 $D_{t+1}(i) < D_t(i)$

3 Classificatore complessivo: $H(x) = \text{sgn} \left(\sum_{t=1}^T \alpha_t h_t(x) \right)$

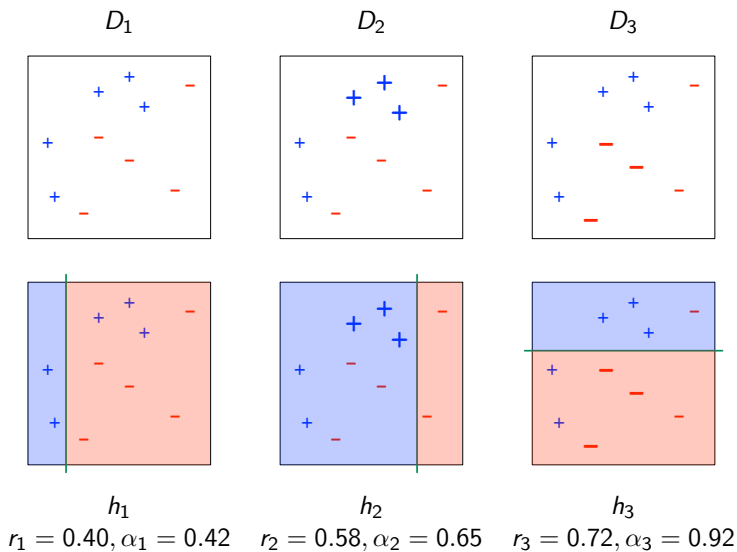
Esempio

Training set:

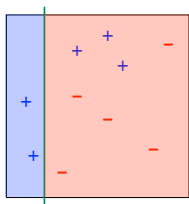


Come classificatori deboli, usiamo dei ceppi di decisione

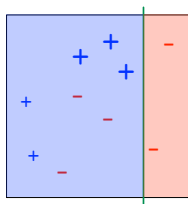
Esempio



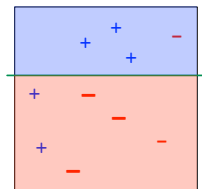
Esempio



$$h_1$$

$$\alpha_1 = 0.42$$


$$h_2$$

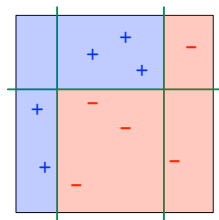
$$\alpha_2 = 0.65$$


$$h_3$$

$$\alpha_3 = 0.92$$

Final classifier:

$$\text{sign}(0.42h_1(x) + 0.65h_2(x) + 0.92h_3(x))$$



Proprietà del boosting

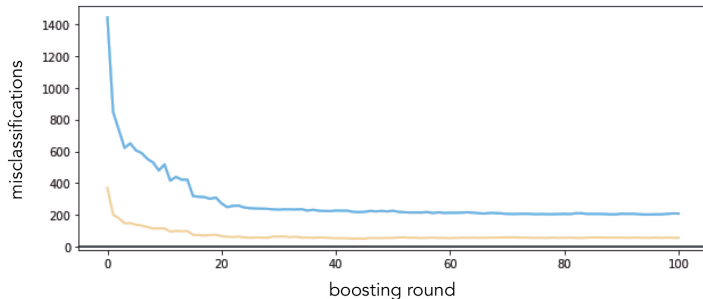
Teorema (performance di AdaBoost)

Si supponga che ad ogni iterazione t , il modello h_t restituito dall'apprendista debole abbia un errore sulla distribuzione D_t che è $\leq 1/2 - \gamma$. Allora, dopo T iterazioni, l'errore di training della regola combinata

$$H(x) = \text{sgn} \left(\sum_{t=1}^T \alpha_t h_t(x) \right)$$

è al più $\exp(-\gamma^2 T/2)$.

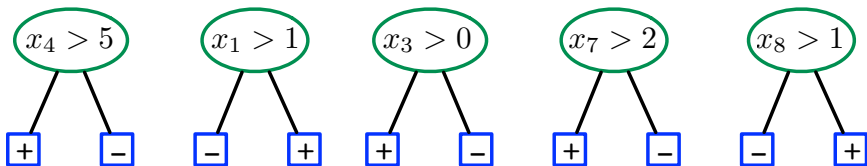
Evoluzione dell'errore durante il boosting



Curva blu: errori sul training set

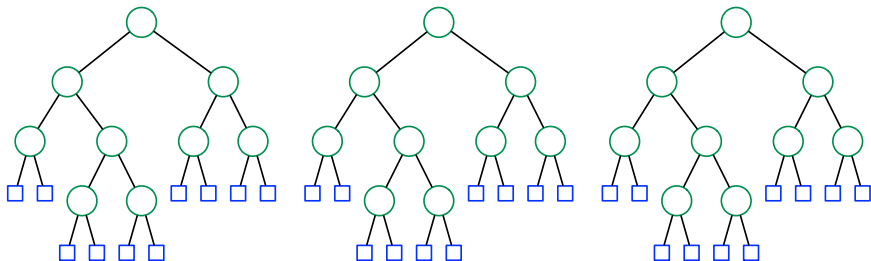
Curva gialla: errori sul validation set

Boosting di ceppi di decisione vs. un albero



Possiamo vedere una combinazione lineare di ceppi come un singolo albero (perché la somma di ceppi è rappresentabile da un albero)

Boosting di alberi



Anziché boosting di ceppi, si può fare il boosting di interi alberi, ottenendo una *foresta*

Il procedimento (come nel boosting di ceppi) è **sequenziale**

Random forest

Per velocizzare il processo si possono costruire gli alberi **in parallelo** anziché sequenzialmente

Random forest

Dato un dataset S di m esempi etichettati:

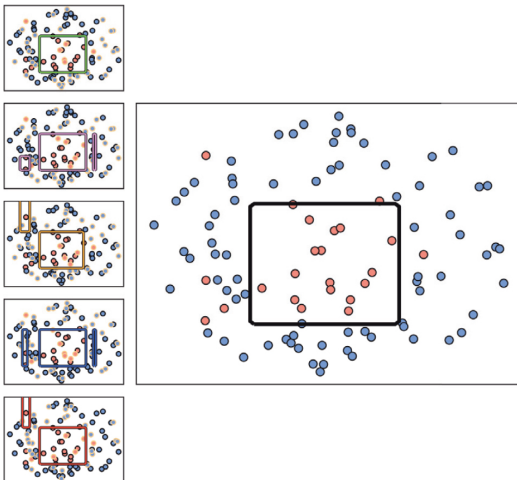
- Per $t = 1, \dots, T$:
 - Seleziona m' esempi casualmente, con rimpiazzo, da S
 - Costruisci un albero di decisione h_t su questi punti
 - Ogni nodo si basa sul valore di una feature tra k scelte a caso

Esempio di impostazioni:

- $m' = m$
- $k = \sqrt{d}$ per input d -dimensionali

Classificatore finale: voto a maggioranza di h_1, \dots, h_T

Esempio



Bagging

In generale, l'idea di **ricampionare** un dataset per ottenere un insieme di dataset correlati ma diversificati è detta *bagging*

Sia il **boosting** che il **bagging** sono concetti generali che ci permettono di ottenere combinazioni di modelli più performanti dei modelli singoli

Svantaggio: un modello **combinato** è anche **meno interpretabile** dei modelli di partenza