I Alberi di decisione

Indice

1 Alberi decisionali per la classificazione

1 Introduzione

Apprendimento statistico

L'**apprendimento statistico** sono delle tipologie di apprendimento che può essere svolta sia in maniera automatica sia tramite ML (Machine Learning).

In parciolare si compone:

$$x^{(i)} \in \mathbb{R}^p \rightarrow \textbf{osservazione}(\text{punto})$$

$$y^{(i)} \in \mathbb{R} \rightarrow \textbf{risposta} \quad \text{con} \quad i = 1, \ldots, n$$

Definiamo il piano/spazio ottenuto dalle osservazione, **spazio delle feature**.

Il nostro obiettivo è quello di ottenere la funzione di uscita in funzione delle osserviazioni date. Questo processo viene detto **inferire**. Formalmente:

$$y = f(x) \quad \text{con } x \in \mathbb{R}^p$$

Riassumendo il nostro problema, noti:

- $(x^{(i)}, y^{(i)})$ con $i = 1, 2, \ldots, n$ osservazioni disponibili$\rightarrow$**Training set**;
- $x^{(i)} \in \mathbb{R}^p :=$ variabili di input indipendenti o anche dette **feature**;

Ottenere:

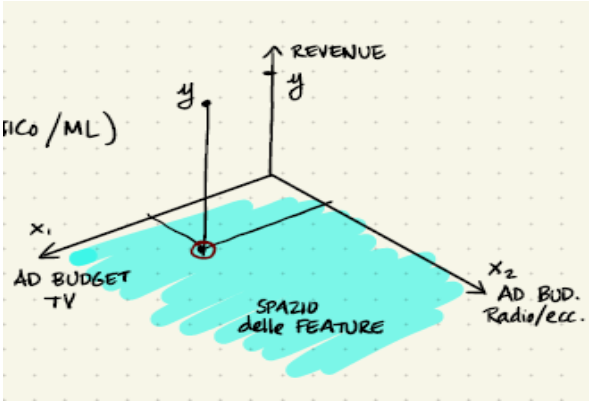
- $y \in \mathbb{R} :=$ variabile di output / **Class** / **etichetta** / **label**.

In altre parole, dato $x \neq x^{(i)}$ qual è la sua uscita? Per rispondere a questa domanda è necessario che la $f(x)$ resistuisca esattamente $y^i = f(x^{(i)})$, cioè che ci sia un **fitting** con i dati del **TS (Training Set)**.

Un modello di questo tipo è un **modello di inferenza/predittivi** poiché permettono di intuire la prossima uscita e producono una y data una certa x.

Apprendimento supervisionato

L'**apprendimento supervisionato** è un tipo di apprendimento per cui l'uscita y è associata un punto dello spazio delle feature. Graficamente:



Apprendimento non supervisionato

L'**apprendimento non supervisionato** è caratterizzato dal fatto che non esiste in corrispondendo di una osservazione nello spazio delle feature la relativa uscita.

Response

Il **response** è l'uscita y che si ottiene dal fitting con il TS.

La response può essere di due tiplogie:

- qualitativa: la **response** è discreta e finita e il suo risultato è una **classificazione** di punti in classi/label. Si cerca quindi di assegnare una classe ad una nuova osservazione conoscendo i dati a disposizione.

  Sono metodi basati su **alberi di decisione.**

- quantitativa: tipicamente è una uscita continua ed i metodi più utilizzati sono la Regressione, Super Vector Machine, Reti Neurali etc..

Quindi il nostre Data Set sarà composto dalle righe che corrispondono alle osservazioni, dalle colonne che sono gli attributi delle varie osservazioni ed a ogni riga, a cui corrispondere un'osservazone con i suoi attributi, è associata una etichetta.

2 Alberi Decisionali

Gli alberi decisionali applicano ricorsivamente metodi euristici per creare delle divisioni nello spazio delle feature per mettere in evidenza un certo aspetto del dataset.

Per ottenere una lbero decisionale dato uno spazio delle feature contenente le osservazioni, occorre:

- **Partizionare/stratificare/segmentare** lo spazio delle feature in modo tale che le partizioni siano il più omogenee possibile in base alla prevalenza o meno di etichette uguali;

- Si associa a ciascuna regione un'etichetta che corrispondendo all'etichetta predominante nella regione anche in presenza di regioni miste;
- Ipotizzando che ci arrivi una nuova osservazione, essa cadrà nello spazio delle feature all'interno di una regione R_i (ottenuta dalle precedenti partizioni) ed assumerà l'etichetta della regione corrispondente;
- Si delimitano i confini delle regioni R_i delle osservazioni e vi si assegnano dei valori;
- L'albero che si formerà sarà costituito da **nodi foglia** e **nodi interni**:
  - i **nodi interni**, sono i **test** che ci permettono di classificare le osservazioni tramite la verifica degli attributi. In particolare se si esamina una sola feauture il test viene detto **univariato (|–)**; altrimenti **multivariato(/);**
  - i **nodi foglia** sono i risulati della nostra classificazione a cui viene associata l'etichetta della regione R_i *a* cui appartengono.

N.B.:

Confronto test univariato vs test multivariato:

- Nel caso di test univariato si potrebbero fare errori di classificazione, ma l'albero risultante è più semplice;
- Nel caso di test multivariato non vi sono errori di classificazione per etichette già note, ma è più soggetto ad errori rispetto al caso univariato(**errore di generalizzazione**), ma l'albero presenta una complessità maggiore data la presenza di numerosi nodi;

Gli attributi di un albero decisionale sono di due tipi:

- **categorici**: cioè qualitativi;
- **continui**: cioè quantitativi;

Dal **training data** cioè l'insieme degli attributi e dell'uscita, si ottiene l'**albero decisionale**.

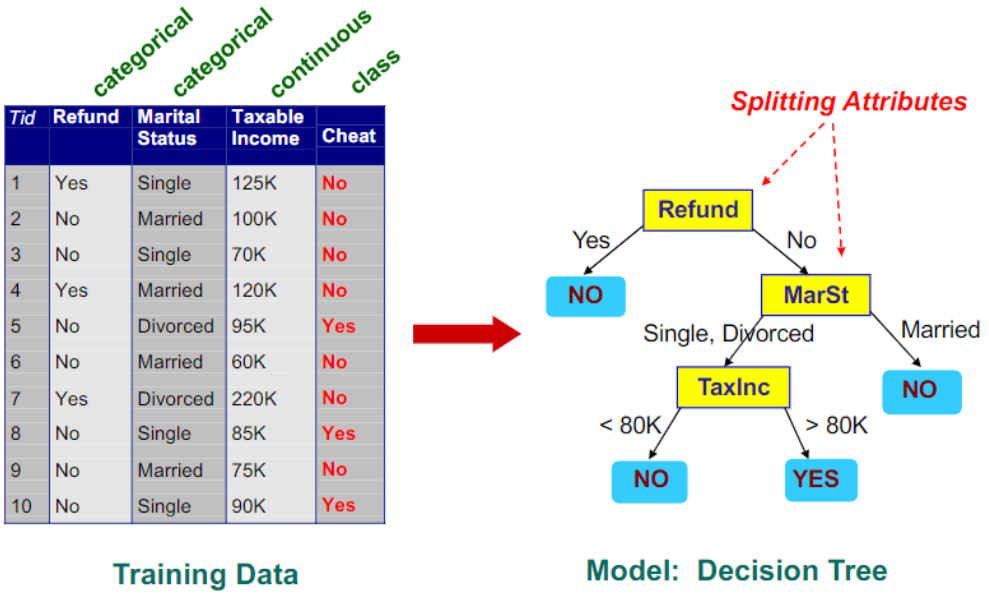


Figura 1.

Ovviamente l'albero risultante varia in base all'ordine in cui i test vengono considerati.

Ogni volta in cui si fa un test, si ha una divisione die punti del TS e il nostro obiettivo è quello di produrre degli insiemi più omogenei possibili su cui applicare il test set.

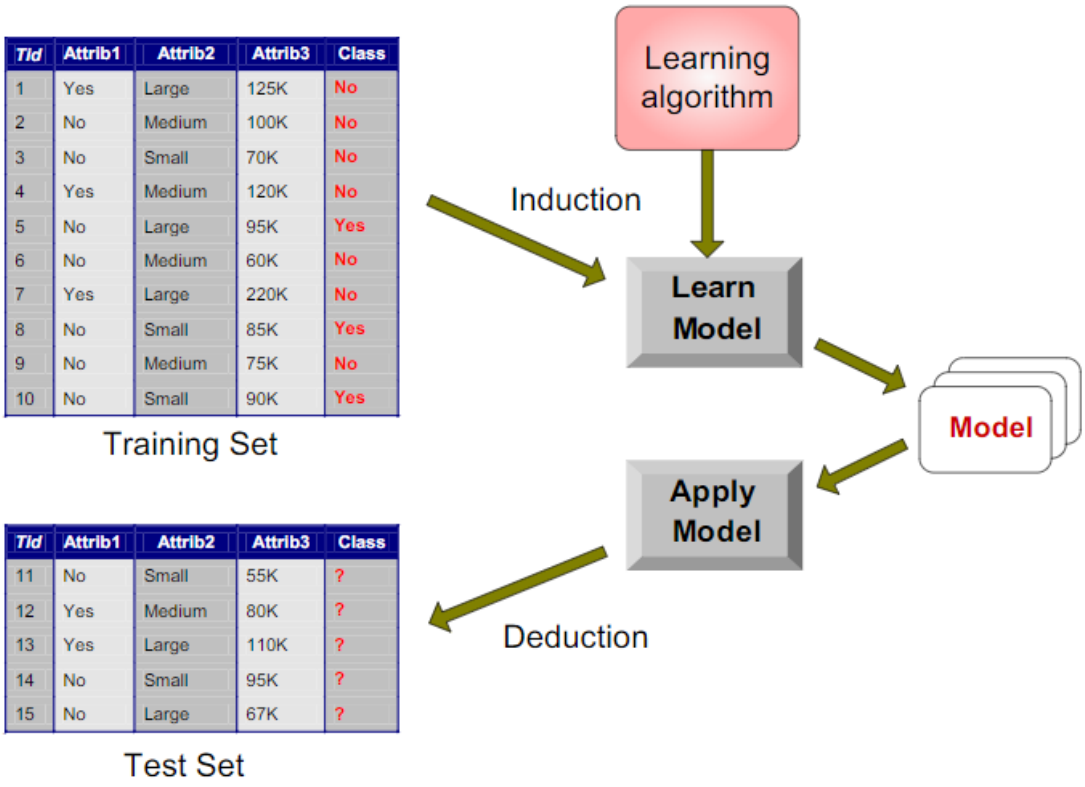


Figura 2.

Pro vs Contro alberi di decisione

- Facilità da spiegare/interpretare e si mima il comportamento umani;
- Facile da manovrare e le variabili sono di tipo qualitativo;
- Strumenti meno accurati;
- ”Nervosismo”: piccole variazioni di dati portanto ad alberi differenti.

Per ovviare a questa ultima problematica si ricorre a dei **metodi di aggregazione** in cui si costruiscono diversi alberi per poi essere uniti tra loro.

TDITD Family

La **TDITD family** compende una serie di Top-Down Induction Decision Trees come il CART e ID3. Questi alberi utilizzando l'induzione, cioè la ricorsione, per creare degli split nello spazio delle feature. Tuttavia, questi potrebbero non rappresentare opportunamente le caratteristiche del dataset.

In particolare, l'albero è usato per classificare i punti del test set secondo gli split e le labels. Questi alberi vengono detti **top-down** poiché, partendndo dal nodo radice, determinano uno split risolvere un problema di ottimizzazione, in genere di minimo rispetto ad una misura di inpurità, per poi applicare la stessa regola alle foglie appena generate.

Questo tipo di classificazione, inoltre, viene detta **greedy**, poiché ogni divisione viene determinata in isolamento senza considerare i possibili impatti che si potrebbero avere nel furuo.

In conclusione, questa famiglia di alberi soffre problemi di complessità.

Algoritmo di Hunt

Algoritmo

1. **INIT:**

list:={training set}
node_list:={root}, Droot:=list: insieme delle proprietà che soddisfano i test
test_pool:={...}

2. **Pick t ∈node list**
3. **Definire $D_t \subseteq$ list/*punti del TS che raggiungono il nodo t*/**
4. **IF $D_t = \varnothing$ THEN t è LEAF label t as "*" (null)**
5. **IF D_t ha elementi tutti di classe K THEN t è LEAF label t as k**
6. **IF D_t ha elementi di classi diverse THEN**
   a. **Choose test ∈test_pool**
   b. **split di D_t in sottoinsiemi $D_t^1, D_t^2, \ldots, D_t^p$**
   c. **update node_list:= node_list$\cup\{t_1\} \cup \{t_2\} \cup \cdots \cup \{t_p\}$**
7. **node_list := node_list$\setminus${t}**
8. **IF node_list=$\varnothing$ THEN STOP ELSE goto 1**

Algoritmo ID3

Algoritmo
ID3 (idea)
1. **Selezione una "window" (sottoinsieme del TS);**
2. **Costruisci un DT usando i punti della window;**
3. **IF $\exists$ elementi $\in$ TS$\setminus${window} che non sono classificati correttamente THEN**
   window:=window $\cup$questi elementi
   ELSE Return DT and STOP

3 Errori negli alberi di classificazione

- **Errore di classificazione:=** sDato un DT, si ha errore di classificazione se il numero di elementi del TS che vengono etichettato in modo corretto. Una sua minimizzazione troppo marcata non porta ad un buon albero poiché soffre dell'**overfitting**. In aggiunta, nel caso di **overfitting**, si eliminano alcuni rami per avere un albero più "semplice", **Tree Pruning**;
- **Errore di generalizzazione:=** è un errore atterso che il DT compie sui punti $\notin$TS. Viene stimato con il test set.

Indici di (in)purity

Nella TDIDT "family" la costruzioni dei DT, di norma si sceglie la **strategia greedy**: in ogni passo si sceglie il test che ottimiza un dato criterio disinteressandosi completamente di quello che può accadere in seguito.

Tuttavia, vogliamo un certo livello di **omogeneità**. A tale scopo si introducono i **purity indices** che indica quanto un sottoinsieme è omogeneo, cioè composto dalle stesse classi.

Errore di classificazione

Sia $p_i \in [0,1]$ frequenza con cui l'etichetta i comprare nell'insieme. Allora:

$$e = 1 - \max_i \{p_i\} \longrightarrow \arg_i \max \{p_i\} := \text{moda}$$

$$0 \leqslant e \leqslant 1 - \frac{1}{k} \quad \text{con } k := \#\text{di etichette}$$

Gini Index

$$g = 1 - \sum_{i=1}^{k} p_i^2 \qquad 0 \leqslant g \leqslant 1 - \frac{1}{k}$$

Entropia

$$E = -\sum_{i=1}^{k} p_i \log_k(p_i) \qquad 0 \leqslant E \leqslant 1$$

Dimostrazione.

Si ha omogeneità massima $p_i = 1, p_j = 0 \qquad \forall j \neq i$

Per $p = \frac{1}{k} \quad \forall i$, si ha che:

$$E = -\sum_{i=1}^{k} \left(\frac{1}{k}\right) \log_k\left(\frac{1}{k}\right) =$$

$$= \sum_{i=1}^{k} \left(\frac{1}{k}\right) (-1) \log_k\left(\frac{1}{k}\right) =$$

$$= \sum_{i=1}^{k} \left(\frac{1}{k}\right) \log_k\left(\frac{1}{k}\right)^{-1} = \sum_{i=1}^{k} \left(\frac{1}{k}\right) 1 = 1$$

□

Importante 1. Noi vogliamo usare gli indici per capire il tipo di test da utilizzare per generare foglie con un grado di omogeneità maggiore. A tale scopo, si sommano gli indici di impurità dei figli:

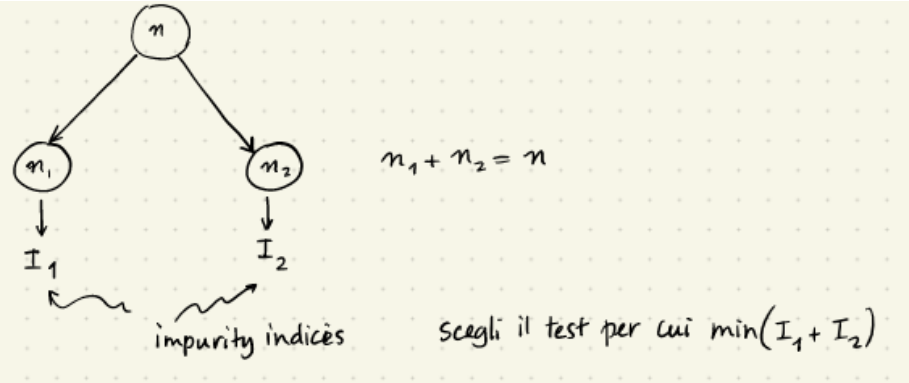


Figura 3.

Test

Si possono effettuare vari tipi di test, i più comuni sono i seguenti:

- **ERR_TEST**$=\sum_{j \quad \text{figlio}}$ ERR_CLASS(j); si sceglie il test con indice minonre
- **GINI_TEST**$=\sum_{j\,\text{figlio}}$ GINI$(j)\,\frac{n_j}{n}$ si sceglie il test con indice minore
  In cui n_j sono i punti nel nodo figlio e n i punti nel nodo padre
- **INFOMATION GAIN_TEST**= misura quando si guadagna in omogeneità negli insieme con un certo tipo di test:

  $$\text{INFORMATION\_GAIN\_TEST} = \text{ENTROPY(padre)} - \sum_{j \quad \text{figlio}} \left(\frac{n_j}{n}\right) \text{ENTROPY}(j)$$

  Si sceglie il test con indice maggiore;
- **GAIN_RATIO_TEST**= equivale all'information gain test, ma con un andamento uniforme:

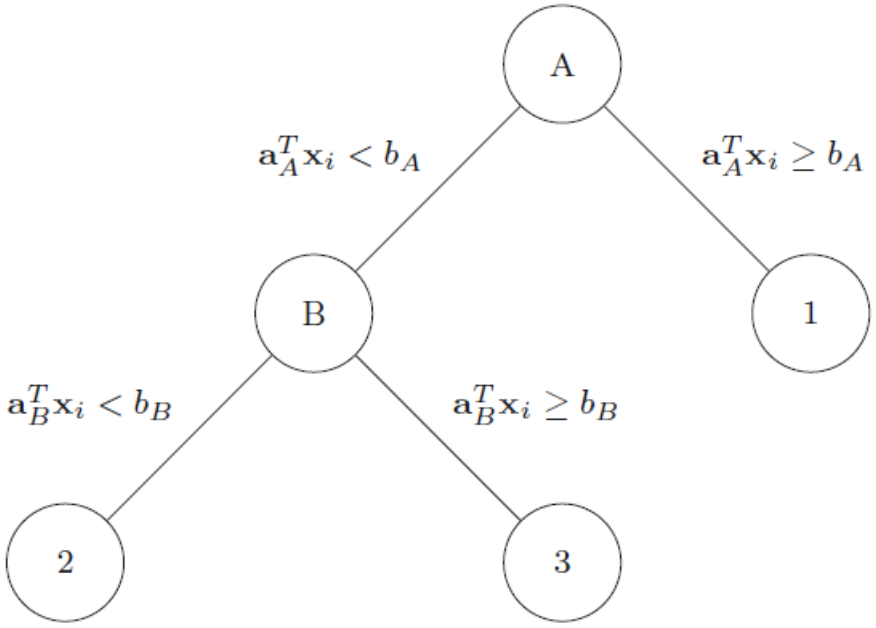
  $$\text{GAIN\_RATIO\_TEST} = \frac{\text{INFORMATION\_GAIN\_TEST}}{\left[-\sum_{j\,\text{figlio}} \left(\frac{n_j}{n}\right) \log\left(\frac{n_j}{n}\right)\right]}$$

4 Albero Ottimo di Classificazione

Introduzione

Il problema di albero decisionale ottimale tenta di risolvere il problema di classificazione di un dataset creando un albero decisione per ottenere l'ottimalità globale.

Esempio 2.



Assumiamo:

- Sia dato un Training Data (X,Y) contenente n osservazioni $(x_i, y_i), i = 1, \ldots, n$ ognuna delle quali con p feature;
- Modello univariato, cioè i test sono effettuati su un attributo alla volta;
- $y_i \in \{1, \ldots, k\}$ K classi/etichette;
- Attributi/feature continue (quantitative) $x_i \in [0, 1]^p$;
- Criterio: min(errore di classificazione + complexity).

I metodi degli alberi decisionali tentano di effettuare una partizione ricorsiva $[0, 1]^p$ per ottenere delle regioni disgiunte che rappresentano l'albero decisionale. L'albero finale sarà formato da nodi foglia e nodi di diramazione:

- I **nodi di diramazione** si dividono con parametri $\boldsymbol{a}$ e $\boldsymbol{b}$. Per nu dato punto i, $a^T x_i < b_i$ il punto seguirà la diramazione sinistra dal noto, altrminenti la destra;
- I **nodi foglia** sono assegnati ad una classe che determinerà la predizione di tutti i punti che cadono all'interno della folgia. La classe viene assegnata in base alla moda della foglia.

Decisioni del problema

Nella creazion dell'albero, in ogni iterazione, dobbiamo effettuare un numero fissato di decisioni:

- In ogni nodo dobbiamo decidere se dividere o fermaci;
- Dopo che abbiamo scelto di fermarci in un nodo, dobbiamo scegliere l'etichetta da assegnare al nuovo nodo foglia;
- Dopo che abbiamo scelto di dividere, dobbiamo scegliere su quali delle variabili effettuare il test;
- Quanndo classifichiamo i punti di traning secondo la costruzione dell'albero, dobbiamo decidere a quali dei nodi foglia assegnare un punto tale che la struttura dell'albero viene rispettata.

Struttura del problema

Consideriamo il problema di costruire un albero ottimale di decisione con la massimo profondità di D. Dato questo parametro:

- Albero binario completo (sotto albero di);
- Profondità fiddasa D$\Longrightarrow$# nodi $2^{D+1}-1=\mathbb{T}$

Notazione 3.

- $t \in \{T_B \cup T_L\} \setminus \{1\}$;
- $p(t) :=$ *nodo padre/genitore;*
- $A(t)$ *è l'****insieme degli antenatii*** *del nodo t;*
- $A_L(t), A_R(t)$ *sono rispettivamente gli* ***insiemi antenati di sinistra e di destra****, presi, partendo dalla radice fino a t, scegliendo rispettivamente la direzione destra o sinistra.;*
- $A(t) = A_L(t) \cup A_R(t)$

Quindi dividiamo i nodi in:

- **Branch Nodes:** sono tutti i nodi $t \in T_B = \left\{1, 2, \ldots, \left\lfloor \frac{T}{2} \right\rfloor\right\}$ che applicano la diisione nella forma $a^T x < b$ se lo soddisfano t seguirà il ramo sinistro, altrimenti il desto.

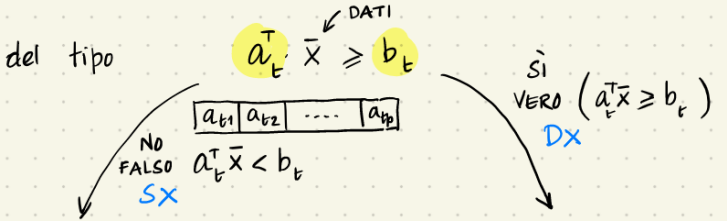


Figura 4.

- **Leaf Nodes**: Sono tutti i nodi $t \in T_L = \left\{\left\lceil \frac{T}{2} \right\rceil, \ldots, T\right\}$ formano una classe di predizione per ogni punti che cade nel nodo foglia.

Variabili

Supponiamo $t \in T_B$

- $a_t \in \mathbb{R}^p$ coefficiente del test $\longrightarrow a_t \in \{0,1\}^p \quad \bar{x} \in [0,1]^p \qquad \sum_{i=1}^{p} a_t = 1$

$$(a_t \cdot \bar{x}) \in [0,1]$$

- $b_t \in \mathbb{R} \longrightarrow b_t \in [0,1]$ è uno scalare
- $d_t \in \{0,1\}$ variabile che indica se si effettua o no un test $d_t \in \{0,1\}$

Nota 4.

L'idea è quella di non permettere la divisione in un branch node. A tale scopo utilizziamo la variabile d_t per sapere su quale branch node si applica la divisione. Se un nodo di diramazione non applica lo split, allora viene modellato con $a_t = 0$ e $b_t = 0$.

Di conseguenza, si forzano tutti a seguire il ramo destro del nodo t. Questo permette all'albero non smettere di crescere.

Vincoli

1. $\sum_{i=1}^{p} a_{t,i} = d_t \qquad \forall t \in T_B$ se effettuo o no il test
2. $0 \leqslant b_t \leqslant d_t \qquad \forall t \in T_B$
3. $d_t \leqslant d_{p(t)} \qquad \forall t \in T_B$ rappresenta la coerenza con d_t se non si effetua lo SPLIT no split.

Allocazione

Il problema di **allocazione** riguarda l'assegnazione dei punti del training set alle foglie dell'albero decisionale e associarli agli errori che sono indotti da questa struttura.

Variabili

- Indichiamo con $z_{it} = \{0,1\} \quad i = 1,\dots,n, t \in T_L$ se il punto x_i appartiene o meglio alla foglia t;
- Per verificare se una foglia possiede dei punti. Utilizziamo:

$$l_t \in \{0,1\} \quad t \in T_L \quad l_t = \begin{cases} 1 & \text{se la foglia } t \text{ ha assegnati dei punti} \\ 0 & \text{altrimenti} \end{cases}$$

- Il numero minimo di punti in ogni foglia $N_{\min}$,

Vincoli

$$\begin{aligned} z_{it} &\in \{0,1\} \quad i = 1,\dots,n \quad t \in T_L \\ \sum_{t \in T_L} z_{it} &= 1 \quad i = 1,\dots,n \quad \text{deve } \exists \text{una foglia per il punto } i \\ \sum_{i=1}^{n} z_{it} &\geqslant N_{\min} l_t \quad t \in T_L \quad \text{Una foglia deve essere sufficientemnte popolata} \end{aligned}$$

A questo punto dobbiamo aggiungere dei vincoli di coerenza che impongono le divisioni richieste fino al raggiungimento della foglia:

- Per i test **non soddisfatti**, deve valere: (9)

$$a_m^T x_i < b_i + M_1(1 - z_{it}) \quad i = 1,\dots,n \quad \forall t \in T_B, \forall m \in A_L(t)$$

- Per i test **soddisfatti**, deve valere:

$$a_m^T x_i \geqslant b_i - M_2(1 - z_{it}) \quad i = 1,\dots,n \quad \forall t \in T_B, \forall m \in A_R(t)$$

Nota 5.

Il vincolo •(9) non è supportato dei risolutori. Quindi occorre convertirlo in una forma che non utilizza la disuguaglianza stratta. A tale scopo, aggiungiamo un fattore piccolo ε e il vincolo diventa:

$$a_m^T x_i + \varepsilon \leqslant b_i + M_1(1 - z_{it}) \quad i = 1,\dots,n \quad \forall t \in T_B, \forall m \in A_L(t)$$

Tuttavia, se ε è troppo piccolo, potrebbe casuare **instabilità** numeriche, quindi occorre renderlo il più grande possibile senza influenzare la soluzione in qualsiasi problema valido.

Occorre definire un e_j per ogni feature j; il più gradnde valore valido è la distanza diversa da 0 più piccola tra i valori adiacenti della feature j:

$$\varepsilon_j = \min\left\{\left|x_j^{(i+1)} - x_j^i\right| \quad x_j^{(i+1)} \neq x_j^i \quad i = 1,\dots,n-1\right\}$$

In cui x_j^i è il più grande valore della feature j-esima.

Dall nota 5 possiamo utilizzare ε nel vincolo, dove ε_j corrisponde alla feature sul quale effettuiamo lo split:

$$a_m^T(x_i+\varepsilon) \leqslant b_m + M_1(1-z_{it}) \quad i=1,\dots,n \quad \forall t \in T_B, \forall m \in A_L(t)$$

Scelta M_1 e M_2

M_1 e M_2 sono dette costanti big-M. Valutando i vincoli, sappiamo che:

- Il masismo valore di $a_m^T(x_i+\varepsilon)-b_m$ è $1+\varepsilon_{\max}$ in cui $\varepsilon_{\max}=\max_j\{\varepsilon_j\}$. Quindi vogliamo:
$$M_1 \geqslant 1+\varepsilon_{\max}$$
- Il massimo valore di $b_t - a_t^T x_i$ è 1 e quindi possiamo scegliere:
$$M_2 \geqslant 1$$

Ottenendo, in conclusione gli ultimi due vincoli:

$$a_m^T x_i + \varepsilon \leqslant b_i + (1+\varepsilon_{\max})(1-z_{it}) \quad i=1,\dots,n \quad \forall t \in T_B, \forall m \in A_L(t)$$

$$a_m^T x_i \geqslant b_i - (1-z_{it})$$

Funzione Obiettivo

La funzione obiettivo che vogliamo determinare si basa su due criteri di minimizzazione riguardo:

- la **complessità dell'albero**: corrisponde alla dimensione dell'albero decisionale ed è fissata a priori da D;
- **Errore di classificazione:** assegnamo ad un'etichetta non corretta il costo di 1 e ad una corretta il costo 0;

Dati

Sia:

$$y_{\text{ik}} = \begin{cases} +1 & \text{se } y_i = k \quad \text{il punto } i \text{ ha etichetta } k \\ -1 & \text{altrimenti} \end{cases} \qquad i=1,\dots,n; k=1,\dots,K$$

Inoltre, $N_{kt} :=$ il numero di punti per l'etichetta k nel nodo t e $N_t :=$ il numero di punti nel nodo t:

$$N_t = \sum_{i=1}^{n} z_{it} \quad t \in T_L$$

$$N_{kt} = \frac{1}{2}\sum_{i=1}^{n}(1+y_{ik})\, z_{ik} \qquad k=1,\dots,k$$

Task - obiettivo

Vogliamo assegnare alle foglioe $t \in T_L$ con l_t=1 l'etichetta a lui associata. Essa è la mode delle etichette assegnate nel nodo t.

Inoltre dobbiamo assegnare un'etichetta ad ogni nodo t nell'albero, definita come:

$$c_t \in \{1,\dots,K\}$$

Ovviamente l'etichetta ottima è quella che corrisponde alla moda di in quel nodo, cioè alla frequenza in cui quella determinata etichetta compare nel nodo:

$$c_t = \arg\max_{k=1,\dots,K}\{N_{\text{kt}}\}$$

Al finte di tenere traccia della predizione di ogni nodo utilizziamo la variabile binaria c_{kt}:

$$c_{kt} = \begin{cases} 1 & \text{se } l'\text{etichetta della foglia } t \text{ è } k \\ 0 & \text{altrimenti} \end{cases}$$

A questo punto ci dobbiamo assicurare che in ogni nodo ci sia una solo predizione. A tale scopo introduciamo il vincolo:

$$\sum_{k=1}^{K} c_{kt} = l_t \quad \forall t \in T_L$$

Dato che sappiamo come scegliere l'etichetta in maniera ottimale in ogni nodo, definiamo L_t come l'**errore di classificazione** in ogni nodo che risulta uguale al numero di punti nel nodo minore del numero di punti dell'etichetta pù comune:

$$L_t = N_t - \max_{k=1,\dots,K} \{N_{kt}\} = \min_{k=1,\dots,K} \{N_t - N_{kt}\} \quad t \in T_L$$

Che può essere linearizzato nel seguente modo:

$$\begin{aligned} L_t &\geqslant N_t - N_{kt} - M(1 - c_{kt}) \quad k = 1,\dots,K \quad \forall t \in T_L \\ L_t &\leqslant N_t - N_{kt} + M c_{kt} \qquad k = 1,\dots,K \quad \forall t \in T_L \\ L_t &\geqslant 0 \qquad \forall t \in T_L \end{aligned}$$

Dove M è una costante sufficientemente grande che rende il vincolo intattivo dipendendo dal valore di c_{kt}. Un possibile valore che possiamo prendere è $M = n$.

Il costo totale dell'errore di classificazione è dato da $\sum_{t \in T} L_t$ e la complessità è il numero di split nelll'albero, dato da $\sum_{t \in T_b} d_t$. A questo punto possiamo normalizzare l'errore di classificazione con una **soglia di accuratezza** $\hat{L}$, ottenuta dalla semplice predizione dell'etichette più popolari dell'intero dataset rendendolo di conseguenza, indipendente dalla quantità α.

Finalmente possiamo dunque scrivere la funzione obiettivo:

$$\min \frac{1}{\hat{L}} \sum_{t \in T_L} L_t + \alpha \sum_{t \in T_B} d_t$$

Riepilogo

Mettendo insieme tutte queste formulazioni delle sezioni precedenti possiamo scrivere il nostro modello OCT:

$$\min \frac{1}{\hat{L}} \sum_{t \in T_L} L_t + \alpha \sum_{t \in T_B} d_t$$

$$\begin{aligned}
L_t &\geqslant N_t - N_{kt} - M(1 - c_{kt}) \quad k = 1,\dots,K \quad \forall t \in T_L \\
L_t &\leqslant N_t - N_{kt} + M c_{kt} \qquad k = 1,\dots,K \quad \forall t \in T_L \\
L_t &\geqslant 0 \qquad \forall t \in T_L \\
N_{kt} &= \frac{1}{2} \sum_{i=1}^{n} (1 + y_{ik}) z_{ik} \qquad k = 1,\dots,k \\
N_t &= \sum_{i=1}^{n} z_{it} \qquad \forall t \in T_L \\
\sum_{k=1}^{K} c_{kt} &= l_t \quad \forall t \in T_L \\
a_m^T x_i &\geqslant b_i - M_2(1 - z_{it}) \quad i = 1,\dots,n \quad \forall t \in T_B, \forall m \in A_R(t) \\
a_m^T x_i + \varepsilon &\leqslant b_i + (1 + \varepsilon_{\max})(1 - z_{it}) \quad i = 1,\dots,n \quad \forall t \in T_B, \forall m \in A_L(t) \\
\sum_{t \in T_L} z_{it} &= 1 \quad i = 1,\dots,n \\
\sum_{i=1}^{n} z_{it} &\leqslant l_t \quad t \in T_L \\
\sum_{i=1}^{n} z_{it} &\geqslant N_{\min} l_t \quad t \in T_L \\
\sum_{i=1}^{p} a_{t,i} &= d_t \qquad \forall t \in T_B \\
0 \leqslant b_t &\leqslant d_t \qquad \forall t \in T_B \\
d_t &\leqslant d_{p(t)} \qquad \forall t \in T_B \\
z_{it}, l_t &\in \{0,1\} \quad i = 1,\dots,n \quad \forall t \in T_L \\
a_{jt}, d_t &\in \{0,1\} \quad j = 1,\dots,p \quad \forall t \in T_B
\end{aligned}$$

Importante 6.

La difficoltà del modello è determinata dal numero di variabili z_{it} in quale sono $n \cdot 2^D$.

Inoltre, occorre specificare a priori tre parametri: la massima profondità dell'albero D, la dimensione minima della foglia $N_{\min}$ e il parametro di complessità. Questa scelta viene detta **tuning** (sincronizzazione).

Il tempo di calcolo di un OCT è nell'ordine dei minuti.

Warm Starts

I risolutori traggono grande beneficio quando gli viene fornito una soluzione intera ammissibili come un **warm start** per il processo di soluzione.

Inserendo una forte soluzone di **warm start** il risolutore incrementa notevolemnte la velocità con cui è in grado di generare soluzioni fortemente ammissibili. Quindi fornisce un upper-bound iniziale sulla soluzione ottima che permette di effettuare l'azione di **pruning** e inoltre fornisce un punto di partenza per la ricerca locale euristica.

Se noi avessimo una soluzione generata per la profondità D, questa soluzione è un valido **warm start** per la profondità $D+1$. Questo è importante poiché la difficoltà del problema aumenta con il crescere della profondità e quindi potrebbe risultare svantaggioro eseguire un problema MIO (Mixed-Integer-Optimization) con una piccola profondità per generare un forte warm start.

Quinsi, data una soluzione iniziale ammissibile è possibile confrontare la velocità del solutore nel determinare una sluzine ottima.

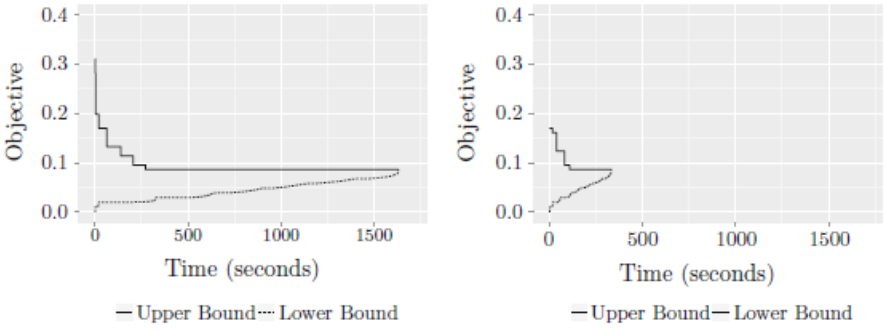


Figura 5. A sinistra senza Warm star; a destra con

Comi si può evincere dall'immagine, si ha convergenza quando upperbound e lower bound coincidono.

Tipicamente il warm start viene fornito da un algoritmo diverso.

5 Addestramento di un OCT

Addestrare una OCT consiste nello scegliere i parametri adatti al modello che vogliamo classificare. Questa azione viene detta **tuning** dei parametri e coinvolge la scelta di $N_{\min}, D, \alpha$ per minimizzare l'**erorre di training** e la **complessità dell'albero**

$$\min \frac{1}{\hat{L}} \sum_{t \in T_L} L_t + \alpha \sum_{t \in T_B} d_t$$

Importante 7.

- $\alpha :=$ è il fattore di importanze se è $=0 \longrightarrow$overfitting

Nota 8.

Un tipico approccio per la scelta di α è quello di discretizzare lo spazio di ricerca e i test di ogni valore.

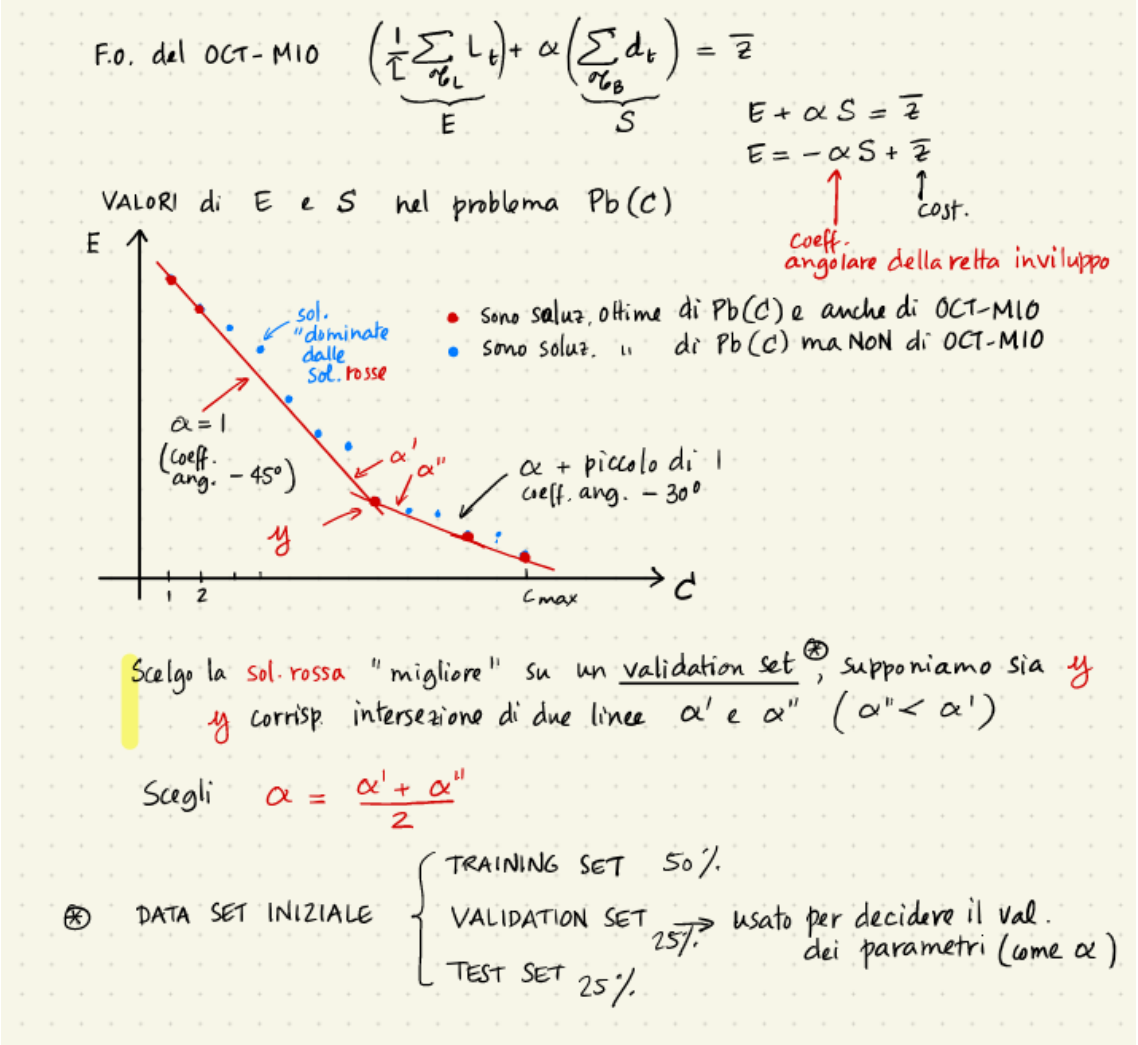
Nell'addestramento di un OCT occorre scegliere la **prodondità** dell'alber $D_{\max}$ da cui si genereranno da $D = 1$(3 nodi) fino a $D_{\max}$.

Possiamo riscrivere il problema di minimizzazione nel seguente modo. Posto $E :=$ errore di training $S :=$ complessità dell'albero:

$$\min E$$

$$S \leqslant C$$

con C è una costante che corrisponde al massimo numero di split $(2^D - 1)$.



Algoritmo 1

1. Scegli $D_{\max}$ e $N_{\min}$;
2. For D=1,..,$D_{\max}$ do:

- For C = 1,...,$2^D - 1$ do:
  - Run TDITD con $\alpha = 0$ e $N_{\min}$. Effettua prune a profondità D,max numero di split = C. Inserisci la soluzione nel pool di warm start:
  - Scegli il candidato più accurato sul validation set nel pool di warm start;
  - Risolvi Pb(C) con profondità D e C split usando il warm start selezionato. Inserisci la soluzione nel pool di warm start;

3. Post-process: rimuovi le soluzioni non ottime per OCT-MIO per alcun valore di α. Determina quindi le soluzioni non dominate e rimuovi le altre.

4. Selezione le soluzioni migliori sul validation set e determina il range di α.

6 OCT: Modelli Multivariati

6.1 Introduzione

Fino ad ora abbiamo considerato alberi di decisione che utilizzano una singola variabile nei loro split ad ogni noto. Questi alberi vengono detti **alberi decisionali univariati.** Ora cerchiamo di estendere questo insieme di alberi decisionali al caso **multivariato**, cioè nel caso in cui negli split di ogni nodo si prendono in considerazione più variabili. Questi alberi decisionali multivariati, vengono detti **OCT-H**.

6.2 Formulazione di OCT-H

Negli alberi decisionali multivariati, non siamo più vincolari a scegliere una singola variabile negli stima, ma possiamo scegliere un generico iperpiano negli split di ogni nodo. Andiamo, ora, a riscrivere i vincoli del nostro problema...

Le variabili a_t sono usate per modellare lo split in ongi nodo ed in particolare $\in[-1,1]^p$:

$$a_{tj} \in [-1,1]$$

$$b_t \in [-1,1]$$

Come nel caso univariato, le variaibli d_t rappresentano la presenza o meno di uno split.

$$\sum_{j=1}^{p} |a_{tj}| \leqslant d_t$$

La problematica di questo vincolo è che è diventato non più lineare. A tale scopo linearizzo tramite delle variabili ausiliarie $\hat{a}_{jt} \geqslant 0 \quad \forall t \in T_B, j = 1,\dots,p$ ottenendo:

$$\hat{a}_{jt} \geqslant a_{jt}; \hat{a}_{jt} \geqslant -a_{jt} \quad j = 1,\dots,p \quad \forall t \in T_B$$

Quindi:

$$\sum_{j=1}^{p} \widehat{a_{jt}} \leqslant d_t$$

Inoltre, abbiamo la variabile b_t che ci indica la presenza di punti dopo la diramazione:

$$-d_t \leqslant b_t \leqslant d_t \quad \forall t \in T_B$$

I **vincoli di consistenza**, mi garantiscono l'assegnamente del punto nel nodo e diventano:

$$\begin{aligned} a_m^T x_i &\geqslant b_m - M(1 - z_{it}) \quad \forall i = 1,\dots,n \quad \forall t \in T_B \quad \forall m \in A_L(t) \\ a_m^T x_i + \mu &\leqslant b_m + M(1 - z_{\text{it}}) \quad \forall i = 1,\dots,n \quad \forall t \in T_B \quad \forall m \in A_R(t) \end{aligned}$$

$\mu = 5 \cdot 10^{-3}$ quantità piccola e di norma si utilizza $M = 2 + \mu$;

$$\begin{aligned} M \leqslant \max\left(a_m^T x_i - b_m\right) &= 2 \\ a_m^T x_i + \mu &\leqslant b_m + (2+\mu)(1 - z_{it}) \quad \forall i = 1,\dots,n \quad \forall t \in T_L \quad \forall m \in A_L(t) \end{aligned}$$

Varia anche la funzione obiettivo poiché la complessità dell'albero varia: vogliamo tenere in considerazione l'uso di test con più attributi.

Introduco $s_{jt} \in \{0,1\} \quad \forall t \in T_B, j = 1,\dots,p$:

$$s_{jt} = \begin{cases} 1 & \text{se al nodo } t \in T_B \text{ utilizzo } j \text{ per la definizione dello split } t \\ 0 & \text{altrimenti} \end{cases}$$

$$s_{jt} \geqslant |a_{jt}|$$

Nota 9.
Nel caso in cui $a_{jt} \neq 0 \longrightarrow s_{jt} = 1$
Nel caso in cui $a_{jt} = 0 \longrightarrow s_{jt} = 0$

Linearizzando il vincolo non lineare:

$$-s_{jt} \leqslant a_{jt} \leqslant s_{jt} \quad \forall t \in T_B, j = 1, \ldots, p$$

Al fine di ottenere una maggiore efficienza si aggiungono altri due vincoli per rendere s_{jt} compatibile con d_{jt}:

$$s_{jt} \leqslant d_t$$

$$\sum_{j=1}^{p} s_{jt} \geqslant d_t$$

Quindi la funzione obiettivo nel caso OCT-H diventa:

$$\min \frac{1}{\hat{L}} \sum_{t \in T_L} L_t + \alpha \sum_{t \in T_B} \sum_{j=1}^{p} s_{jt}$$

6.3 Warm Start OCT-H

Negli OCT-H come Warm start si utilizza un'euristica greedy appartenente alla TDIDT family. Per determinare il migliore split, invece di utilizzare gli impurity index, si utilizza, sui punti "sopravvissuti" al nodo $t \in T_B$ un OCT-H a profondità $D = 1$.

6.4 Tuning degli Iperparametri

Nella scelta degli iperparametrei si utilizza la stessa procedura di OCT con la differenza che:

$$\text{OCT} \longrightarrow C_{\max} = 2^D - 1$$

$$\text{OCT} - H \longrightarrow C_{\max} = p\,(2^D - 1)$$

7 Alberi di Decisione Pro e Contro

La caratteristica degli alberi di decisione è che sono semplici da spiegare/giustificare ai non esperti. Rappresentano in modo più verosimile il processo decisionale umano rispetto ad altri metodi e modellano in maniera immedieta variabili qualitative. Tuttavia, il livello di accuratezza è inferiori e sono poco robusi poiché piccole variazioni nei dati producono cambiamenti significativi nell'albero risultante,

8 Bagging, Random Forests, Boosting

Gli alberi decisionali DT dsoffrono di **varianza elevata:**

- Estrai in modo casuale due o più dataset da una popolazione;
- Ricava alberi di decisione sulla base dei nuovi dataset;
- alberi di decisione molto dissimili;

Per ridurre la varianza si può ricorrere al **bagging** (Bootstrap Aggregation). Questo è un metodo general-purpose utile a ridurre la variazione aumentanto l'accuratezza della predizione tramite l'aggregazione di un insieme di osservazioni.

Idialemnte si ricava un certo numero B di training set della popolazione osservata e se ne considera la media come predizione del modello aggregato.

8.1 Bagging

Si applica l'idea della sezione precedente agli alberi di classificazione. In particolare si costruisce un albero di classificazione a partire da ciascuno dei training set bootstrap $1, \dots, B$.

A questo punto, si classifica sulla base del voto di maggioranza cioè dalla classe di predizione più frequente tra i risultati ottenuti dai B alberi.

In aggiunto, con questo tipo di tecnica non si effettua pruning degli alberi generati a partre dai training set bootstrap. La varianza viene ridotta prendendo il risultato a maggioranza sulle B classificazioni.

A questo punto occorre definire una grandezza per valutare l'errore che stiamo commettendo tramite bagging. Questa grandezza di errore viene detts **stima out-of-bag**.

Un DT nel bagging usa circa $\frac{2}{3}$ dei dati originali. Quindi $\frac{1}{3}$ delle osservazioni non vengono utilizzate per generar eun singolo DT e proprio questi punti/osservazioni rappresentano **Out-Of-Bag**.

L'idea è quella di utilizzare i risultati prodotti dai DT che hanno x_i OOB per ottenere una predizione su x_i. Si ripete questa procedura per tutti gli n punti $x_i \quad i = 1, \dots, n$ e si calcola l'errore di classificazione complessivo.

Ricapitolando il bagging aumenta l'accuratezza della predizione al costo di una minore interpretabilità e utilizza l'indice di Gini come misura dell'importanza di una variabile indipendente j:

Algoritmo 2

Per tutti gli alberi T_b dal training set di bootstrap $b = 1, \dots, B$:
Calcola la diminuzione d_{b_j} dell'indice di Gini associata allo split sulla variabile j
Output$=\frac{1}{B}\sum_b d_{b_j}$ è la misura dell'**importanza relavita della variabile j**

8.2 Random Forests

I random forests sono un miglioramento del bagging basato sulla scelta di alberi meno correlati tra loro. In particolare, si costruiscono B alberi di decisione e ad ogni split:

- si considerano soltanto un **campione casuale** di $m < p$ variabili candidate;
- La scelta di p viene effettuata in cross validation, ma sperimentalmente si è visto che si pone $p \approxeq \sqrt{m}$

La conseguenza della riduzione delle alternative ad ogni split è che si riducono i rischi dovuti alla forte correlazione tra alberi e si aumenta la riduzione della varianza promuovendo la costruizione di alberi poco correlati tra loro.

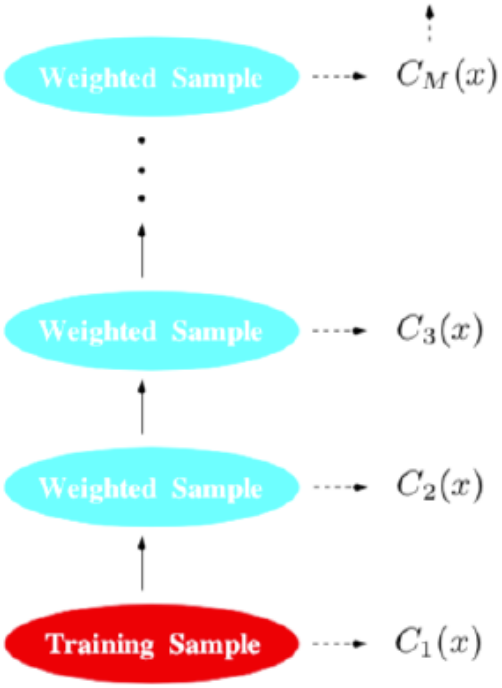
In particolae, ponendo $m = p \longrightarrow$ bagging

Con un numero alto di attributi fortemente correlati, scegliere m piccolo rispettoa p può migliorare le performance di queste teniche rispetto al bagging.

8.3 Boosting

Le tecniche d **boosting** possono essere applicabili a metodi diversi di apprendimento statistico di regressione e classificazione; ogni albero viene creato a partire dalle informazoni sugli alberi creati in precedenza e si combina un numero sufficientemente elevato di B alberi di decisione a partire da B traingin set.

In questo caso, i training set non sono bootstrap, ma sono creati modificando in maniera opportuna i dati originali.



I campioni successivi sono determinati pesando i dati dei campioni precedenti.

Il peso di una osservazione x_i al passo b è maggiore se il DT non ha classificato correttamento x_i al passo $(b-1)$. La classifixazione viene effettuata pesando opportunamente i risultati dei vari DT.

AdaBoost (Freund, Shapire, 1996)

- Init. pesi dei dati $\boldsymbol{x}_i$ del training set: $w_i = (1/n)$, $i = 1, \ldots, n$
- **For** $b = 1, \ldots, B$ **do**:
  - Genera DT(b) su training data b-mo usando pesi w_i
  - Calcola errore pesato di DT(b) : $e(b) = \Sigma\, w_i * \mathrm{I}(i \text{ misclass.}) \,/\, \Sigma\, w_i$
  - Calcola $\alpha(b) = \log[(1 - e(b)) \,/\, e(b)]$
  - Aggiorna: $w_i = w_i * \exp[\alpha(b) * \mathrm{I}(i \text{ misclass.})]$ e rinormalizza
- Output risultato pesato con gli $\alpha(b)$

8.4 Adaboost ST

Assumo che le label $\in\{-1, 1\}$ ed ho M weak learners k_j a stamp:

$$C(x_i) = \sum_{j=1}^{M} \alpha_j k_j(x_i) \quad x_i \in \mathrm{TS} \quad x_i \in \mathbb{R}^p$$

La risposta:

$$y = \frac{C(x_i)}{|C(x_i)|}$$

Il suo segno determina la risposta del sistema e il denominatore è proporzionale alla confidenza, cioè maggiore è il suo valore, maggiore sarà la confidenza della risposta per il punto del TS.

Inoltre:

$$C_m(x_i) = \sum_{y=1}^{M} \alpha_j k_j(x_i) \longrightarrow C(x_i) = C_M(x_i); C_m(x_i) = C_{m-1}(x_1) + \alpha_m k_m(x_i)$$

A questo punto si necessiata di una funzione per determinare l'errore di C_m:**exponential loss**.

8.5 Errore di classificazione al passo m-esimo

$$E = \sum_{i=1}^{n} e^{y_i C_m(x_i)} \equiv \sum_{i=1}^{m} e^{-y_i C_{m-1}(x_i)} e^{\alpha_i k_m(x_i) y_i} = \sum_{i=1}^{m} W_i^{(m)} e^{\alpha_i k_m(x_i) y_i}$$

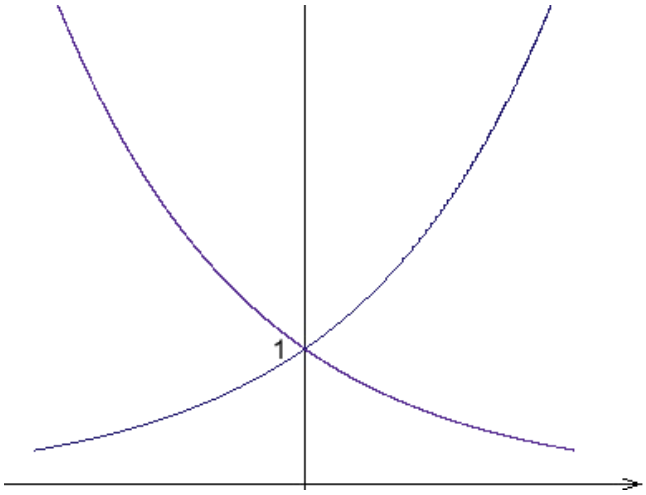


Figura 6.

8.6 Scelta del Weak Learner al passo m

Il mio obiettivo è quello di avere E minimo:

$$E = \sum_{i:K_m(x_i) \neq y_i}^{m} W_i^{(m)} e^{-\alpha_m k_m(x_i) y_i} + \sum_{i:K_m(x_i) = y_i}^{m} W_i^{(m)} e^{\alpha_m k_m(x_i) y_i} =$$

$$= \sum_{I^=} W_i^{(m)} e^{-\alpha_m} + \sum_{I^{\neq}} W_i^{(m)} (e^{\alpha_m} - e^{-\alpha_m})$$

In cui il primo termine non dipenda da $k_m(\cdot)$, mentre il secondo non dipende ne da i ne da $k_m(\cdot)$. Quindi, per minimizzare viene scelto maniera tale che:

$$\sum_{I^{\neq}} W_i^{(m)}$$

sia minimo.

8.7 Scelta del parametro α_m

Voglio minimizzare E come forma di α_m. Quindi per trovare il minimo derivo $\frac{\partial E}{\partial \alpha_m} = 0$:

$$E = \sum_{i \in I^{\neq}}^{m} W_i^{(m)} e^{-\alpha_m k_m(x_i) y_i} + \sum_{i \in I^=}^{m} W_i^{(m)} e^{-\alpha_m k_m(x_i) y_i} = \sum_{i \in I^{\neq}}^{m} W_i^{(m)} e^{\alpha_m} + \sum_{i \in I^{\neq}}^{m} W_i^{(m)} e^{-\alpha_m} =$$

$$= e^{\alpha_m} W^E - e^{-\alpha_m} W^C$$

Quindi:

$$\frac{\partial E}{\partial \alpha_m} = 0 \Longleftrightarrow e^{\alpha_m} W^E = e^{-\alpha_m} W^C$$

$$\alpha_m + \ln W^E = -\alpha_m + \ln W^C$$

$$\alpha_m = \frac{1}{2}\ln\left(\frac{W^C}{W^E}\right) = \frac{1}{2}\ln\left(\frac{1-\varepsilon}{\varepsilon}\right)$$

Rappresenta un errore complessivo pesato che può essere scritto come:

$$\varepsilon = \frac{W^E}{W^E + W^C}$$

2 Classificazione Robusta

1 Introduzione

Nel caso in cui ci fosse incertezza sui dati, si ha un peggioramento delle prestazioni del classificatore. A tale scopo vogliamo delle procedure che rendano il classificatore più robuto rispetto a delle piccole incertezze. L'approccio standard è la **regolarizzazione** che si basa sulle funzioni di penalità per diminuire l'overfitting. UN altro approccio è quello di utilizzare l'**ottimizzazione robusta.**

2 Robust Classification

Il problema di ottimmizzazione è del timo:

$$\max c(x, y)$$

$$g(x, u) \leqslant 0^m$$

$$x \in X$$

In cui:x

- $x :=$ variabili di decisione;
- $u =$ parametri;
- $g(x, u) \leqslant 0^m :=$ vincoli di diseguaglianza.

I **parametri** u possono essere:

- fissati$\longrightarrow$problema di ottimizzazione incerti;
- incerti$\longrightarrow u \in U$ insieme di incertezaò

.Cerchiamo la soluzione ottima nel caso peggiore.

3 Approccio MaxMin

$$\max_{x \in X} \min_{u \in U} \{c(x, u) \colon g(x, u) \leqslant 0^m\}$$

Se $|U| = +\infty$ allora si hanno infiniti vincoli che pososno essere ridotti ad un numero finito.

Definizione 10. *Problema Norma Duale*

La norma duale è un numero reale associato ad una funzione lineare definita sullo spazio vettoriale X.

Se $X = \mathbb{R}^n$: una funzione lineare generia $f = a^T x$

$$z_q = \sup\{|f(x)|: \|x\|_q < 1\}$$

$$z_q = \max\{a^T x: \|x\|_q \leqslant 1, x \in \mathbb{R}^n\}$$

Si può dimostrare che la soluzione di questo problema è:

$$z_q = \|a\|_{q^*} \quad \text{con } q^* = \frac{q}{q-1} \quad \text{se } q \neq 1$$

$$z_q = \|a\|_\infty \quad \text{Se } q = 1$$

Si può estenedere questo problema nella norma di $f(x)$ ristretta da un qualsiasi numero $\rho > 0$. Ottenendo:

$$\begin{aligned} z_q &= \max\{a^T x: \|x\|_q \leqslant \rho: x \in \mathbb{R}^n\} = \\ &= \max\{a^T \rho\, y: \|y\|_q \leqslant 1, y \in \mathbb{R}^n\} = \\ &= \rho \|a\|_{q^*} \end{aligned}$$

4 Uncertainty Set

L'**insieme di incertezza** può essere utilizzato per modellare **l'incertezza delle feautre.** Poiché l'incertezza nelle feature è dovuto a mancanza id dati, errori di manipolazioni e errori di misura vogliamo costruire un modello di ottimizzazione per il classificatore che sia una **controparte robusta** del modello OCT.

A tale scopo, supponiamo che:

$$x_i \in \mathbb{R}^p \text{ è il valore esatto}$$

$$(x_i + \Delta x_i) \quad \forall i = 1, \ldots, n \quad \text{Training data effettivo}$$

con $\Delta x_i :=$ perturbazione dell'$i-$esimo dato.

A questo punto, possiamo definire un insieme delle perturbazioni:

$$\Delta X = \{\Delta x_1, \ldots, \Delta x_n\} \in \mathbb{R}^{p \times n}$$

Ottenendo:

$$U_x = \{\Delta x \in \mathbb{R}^{p \times n}: \|\Delta X_i\|_q \leqslant \rho, i = 1, \ldots, n\}$$

Il termine ρ viene detto **parametro di magnitudo(incertezza)** e viene scelto tramite cross validation.

5 Modello MIP (Programmazione Intera Mista)

Se consideriamo un albero binario non compelto, esso sara formato da $\left\lceil \frac{\#D}{2} \right\rceil$. Fissata la struttura, il numero di nodi T è dispari ed abbiamo:

$$\#\text{Foglie} = \left\lceil \frac{T}{2} \right\rceil$$

I nodi sono numerati in modo che $k = 1, 2, \ldots, \left\lfloor \frac{T}{2} \right\rfloor, \left\lceil \frac{T}{2} \right\rceil, \ldots, T$.

Al poso delle vairabili $d_t = \begin{cases} 1 \text{ split nel nodo } t \\ 0 \text{altirmenti} \end{cases}$, si utilizza:

$$\delta_k = \begin{cases} 1 \text{ se non si ha lo split al nodo } k \\ 0 \text{ altrimenti} \end{cases}$$

5.1 Vincoli Strutturali

1. $\delta_k = 1 \quad \text{k}=\left\lceil \frac{T}{2} \right\rceil, \ldots, T;$
2. $\delta_k \geqslant \delta_n \quad k = 1, \ldots, T : u \in A(k);$
3. $\delta_k + \sum_{i=1}^{P} a_{k,i} = 1 \qquad k = 1, 2, \ldots, T$

5.2 Allocazione Punti nelle Foglie

$$z_{i,k} \in \{0,1\} \quad i = 1, \ldots, n \quad k = 1, \ldots, T$$

Importante 11. $z_{\text{ik}} = 1 \Longleftrightarrow$ assegno punto i al nodo k

Vincoli

- $\sum_{k=1}^{T} z_{ik} = 1 \qquad \text{i=1,..,n}$
- $z_{ik} \leqslant \delta_k \qquad i = 1, \ldots, n \quad k = 1, \ldots, T$
- $\sum_{i=1}^{n} z_{ik} \geqslant N_{\min} l_k \quad l_k \in \{0,1\} \quad k = 1, \ldots, T \quad l_k = 1$ se la foglia popolata da punti
- $z_{ik} + \delta_u \leqslant 1 \quad i = 1, \ldots, n; k = 1, \ldots, T; u \in A(k)$
- $l_k + \sum_{u \in A(k)} \delta_u \geqslant \delta_k \qquad k = 1, \ldots, T.$

5.3 Funzione Obiettivo

Assumiamo 2 classi $y_i \in \{-1; +1\} \quad \forall i = 1, \ldots, n$

Variabili

$g_k, h_k \in \mathbb{Z}_{\geqslant 0} (\mathbb{R} \geqslant 0)$ contano il numero di punti assegnati $a\, k$ che assumo ± 1

$$(\text{numero di} -1 \text{ in } k) \qquad g_k = \frac{1}{2} \sum_{i=1}^{n} z_{ik}(1 - y_i) \qquad \forall k \in T$$

$$(\text{numero di } 1 \text{ in } k) \quad h_k = \frac{1}{2} \sum_{i=1}^{n} z_{ik}(1 + y_i)$$

Vincoli:

$$(\alpha)\, f_k \leqslant g_k + M\,[w_k + (1 - l_k)] \qquad (\gamma)\, f_k \geqslant -M[1 - w_k + (1 - l_k)]$$

$$(\beta)\, f_k \leqslant h_k + M[1 - w_k + (1 - l_k)] \qquad (\varepsilon)\, f_k \geqslant h_k - M[w_k + (1 - l_k)]$$

Se $l_k = 0 \longrightarrow$ vincoli soddisfatti

Se $l_k = 1$:

$$w_k \in \{0,1\} \qquad k = 1, \ldots, T$$

Se $w_k = 0 \quad \beta e \gamma$ sono soddisfatti banalmente, ma i vincoli attivi α, ε impongono:

$$h_k \leqslant f_k \leqslant g_k$$

Se $w_k = 1$ $\alpha.\varepsilon$ vengono soddisfatti e rimangono attivi gli altri due e impongono:

$$g_k \leqslant f_k \leqslant h_k$$

Quindi l'insieme f_k contiene tutti i punti malclassificati e quindi può rappresentare la funzione da minimizzare:

$$\min \left[\sum_{k=1}^{T} f_k + \alpha \sum_{k=1}^{T} (1 - \delta_k) \right]$$

Coerenza dei Test

$$a_u^T x_i \leqslant b_u + M\,(1 - z_{ik}) \quad u \in A_l(k) \quad \forall i = 1, \ldots, l$$

$$a_u^T x_i \geqslant b_u - M\,(1 - z_{ik}) \quad u \in A_R(k) \quad \forall i = 1, \ldots, l$$

6 Robustezza vs. Incertezza delle feauture

$$U_x = \{\Delta x \in \mathbb{R}^{p\times n} : \|\Delta x_i\|_q \leqslant \rho . i = 1, \dots, n\}$$

Nei vincoli 5.3 della coerenza dei testi, **controparte robusta**:

$$a_u^T(x_i + \Delta x_i) + \varepsilon \leqslant b_u + M(1 - z_{ik}) \quad \forall i, k, u \in A_L(k) \quad \forall \Delta x \in U_x$$

Possiamo riscriverlo come:

$$a_u^T \Delta x_i \leqslant -a_u^T x_i - \varepsilon + b_u + M(1 - z_{ik})$$

Quindi:

$$\max_{\Delta x \in U_x} \{a_u^T \Delta x_i\} \leqslant -a_u^T - \varepsilon + b_u + M(1 - z_{ik})$$

$$\Updownarrow$$

$$\max_{\|\Delta x = \rho\|} \{a_u^T \, \Delta x : \|\Delta x_i\| \leqslant \rho\} = \rho \|a_u\|_q = \rho$$

La versione robustificata diventa:

$$a_u^T x_i + \varepsilon + \rho \leqslant b_u + M(1 - z_{ik})$$

7 Robustezza vs. Incertezza nelle label

7.1 Dualita PL

Il problema primale di minizzazione in un contesto di classificazione è il seguente:

$$z = \min c^T x$$

$$A\, x \leqslant b$$

$$x \in \mathbb{R}^n_{\geqslant 0}$$

Il suo duale:

$$w = \max u^T b$$

$$u^T A \leqslant c^T$$

$$u \in \mathbb{R}^n_{\leqslant 0^m}$$

Se $\overline{x} \in \mathbb{R}^n_{\geqslant 0}$ tale che $A\,\overline{x} \leqslant b$, cioè ammissibile per il primale e $\overline{u} \in \mathbb{R}^m_{\leqslant 0}$ tale che $\overline{u}^T A \leqslant c^T$, ammissibile per il problema duale allora:

$$c^T \overline{x} \geqslant z = w \geqslant \overline{u}^T b$$

7.2 Robustezza nelle label

Se vi è incertezza nelle label, esse sono affette da errore e quindi occorre robustificare il modello e definire l'insieme di incertezza. Sostanzialmente esse sono dei **flip**: la label da 1 diventa -1 e viceversa. Quindi definiamo:

$$\Delta y_i = \begin{cases} 0 & \text{Se non vi è flip} \\ 1 & \text{altrimenti} \end{cases}$$

Posso definire l'insieme di incertezza come:

$$U_y = \left\{ \Delta y_i \in \{0,1\}^n : \sum_{i=1}^{n} \Delta y_i \leqslant \Gamma \right\} \quad \Gamma \in \mathbb{Z}_+$$

La funzione obiettivo, tenendo conto dellìincertezza sulle label diventa:

$$g_k = \frac{1}{2}\sum_{k=1}^{n} z_{ik}(1 - y_i(1 - 2\,\Delta y_i))$$

$$h_k = \frac{1}{2}\sum_{i=1}^{n} z_{ik}(1 + y_i(1 - 2\,\Delta y_i))$$

I vincolI:

$$(\alpha)\, f_k \leqslant g_k + M\,[w_k + (1 - l_k)] \qquad (\gamma)\, f_k \geqslant -M[1 - w_k + (1 - l_k)]$$

$$(\beta)\, f_k \leqslant h_k + M[1 - w_k + (1 - l_k)] \qquad (\varepsilon)\, f_k \geqslant h_k - M[w_k + (1 - l_k)]$$

Considerando $\alpha + \beta + \gamma + \varepsilon \longrightarrow \text{Se } l_k = \begin{cases} h_k \leqslant f_k \leqslant g_k & \text{se } w_k = 0 \\ g_k \leqslant f_k \leqslant h_k & \text{se } w_k = 1 \end{cases}$

Prendiamo in considerazione solo il primo vincolo, le restanti si svolgono allo stesso modo $\forall k = 1, \ldots, T, \Delta y_i \in U_y$

$$f_k \leqslant \frac{1}{2}\sum_{i=1}^{n} z_{ik}(1 - 2\,\Delta y_i) + M[w_k + (1 - l_k)]$$

$$\frac{1}{2}\sum_{i=1}^{n} (1 - y_i)\, z_{ik} + \sum_{i=1}^{n} z_{ik} y_i\, \Delta y_i \geqslant f_k - M[w_k + (1 - l_k)]$$

$$\frac{1}{2}\sum_{i=1}^{n} (1 - y_i) z_{ik} + \min_{\Delta y \in U} \{z_{ik} y_i\, \Delta y_i\} \geqslant f_k - M[w_k + (1 - l_k)]$$

La funzione di minimo è un problema di programmazione lineare PL, che può essere rilassato:

$$\xi_k = \min_{\Delta y \in U} \{z_{ik} y_i\, \Delta y_i\}$$

$$\sum_{i=1}^{n} \Delta y_i \leqslant \Gamma$$

$$\Delta y_i \in \{0, 1\} \quad i = 1, \ldots, n$$

Che rilassando il vincolo binario:

$$\xi_k = \min_{\Delta y \in U} \{z_{ik} y_i\, \Delta y_i\}$$

$$\sum_{i=1}^{n} \Delta y_i \leqslant \Gamma$$

$$0 \leqslant \Delta y_i \leqslant 1 \quad i = 1, \ldots, n$$

Quindi, risolto questo problema, otteniamo ξ_k:

$$\frac{1}{2}\sum_{i=1}^{n} (1 - y_i) z_{ik} + \xi_k \geqslant f_k - M[w_k + (1 - l_k)]$$

7.3 Proprietà

$$\xi_k^R \leqslant \xi_k \quad \forall k = 1, \ldots, T$$

Allora posso considerare il duale del rilassamento:

$$\xi_k = \min_{\Delta y \in U} \{z_{ik} y_i\, \Delta y_i\} \qquad \tau_k = \max\left(\Gamma \nu_k + \sum_{i=1}^{n} \mu_{ik}\right)$$

$$\sum_{i=1}^{n} \Delta y_i \leqslant \Gamma \iff \mu_{ik} + \nu_k \leqslant z_{ik}\, y_i$$

$$0 \leqslant \Delta y_i \leqslant 1 \quad i = 1, \ldots, n \qquad \mu_{ik} \leqslant 0; \nu_k \leqslant 0$$

Se sono ammissibili per il duale allora:

$$\left(\Gamma\overline{\nu_k} + \sum_{i=1}^{n} \overline{\mu_{ik}}\right) \leqslant \tau_k \leqslant \xi_k^R \leqslant \xi_k$$

Allora 7.2 diventa:

$$\frac{1}{2}\sum_{i=1}^{n}(1-y_i)z_{ik} + \left(\Gamma\overline{\nu_k} + \sum_{i=1}^{n} \overline{\mu_{ik}}\right) \geqslant f_k - M[w_k + (1-l_k)]$$

$$\mu_{ik} + \nu_k \leqslant z_{ik}\, y_i \quad \forall i = 1, \ldots, n$$

$$\mu_{ik} \leqslant 0 \qquad \forall k = 1, \ldots, T$$

$$\nu_k \leqslant 0$$

Avendo così, in conclusione, $n \cdot T$ vincoli invece di T.