

Parte I

Alberi di decisione

Indice

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 paricolare si compone:

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

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

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

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

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

Riassumendo il nostro problema, noti:

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

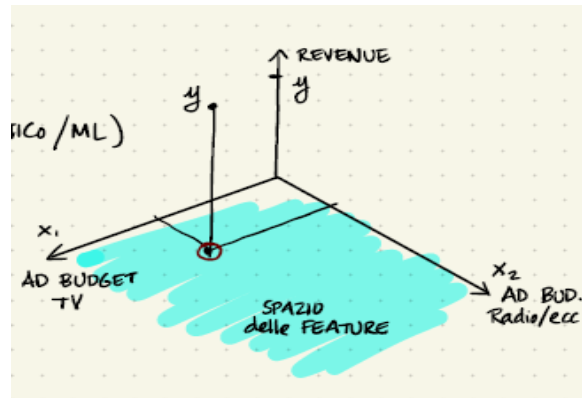
Ottenere:

- $y \in \mathbb{R}$:= 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 corrispondenza 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 tipologie: 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 nostro 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'osservazione 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 un albero 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;
- Ipotezzando 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 feature il test viene detto **univariato** ($|-$); altrimenti **multivariato** ($/$);
 - i **nodi foglia** sono i risultati 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**.

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 dei punti del TS e il nostro obiettivo è quello di produrre degli insiemi più omogenei possibili su cui applicare il test set.

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 portano 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.

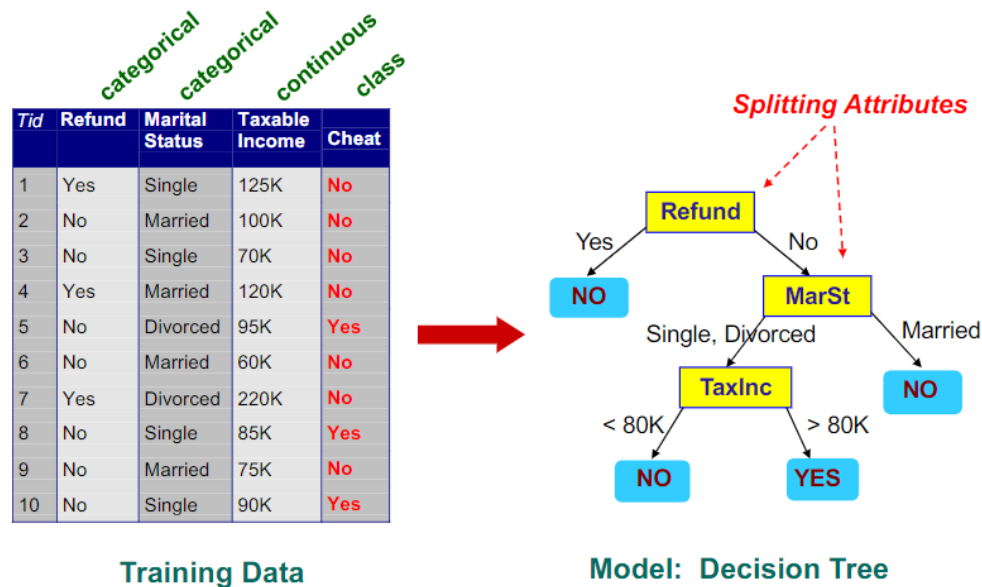


Figura 1:

TDITD Family La **TDITD family** comprende 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 impurità, 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 futuro.

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

Algoritmo di Hunt

Algoritmo

1. **INIT:** $list := \{\text{training set}\}$ $node_list := \{\text{root}\}$, $Droot := list$: insieme delle proprietà che soddisfano
 Choose test $t \in test_pool$
 split di D_t in sottoinsiemi $D_t^1, D_t^2, \dots, D_t^p$
 update $node_list := node_list \cup \{t_1\} \cup \{t_2\} \cup \dots \cup \{t_p\}$
 $node_list := node_list \setminus \{t\}$
 IF $node_list = \emptyset$ THEN STOP ELSE goto 1

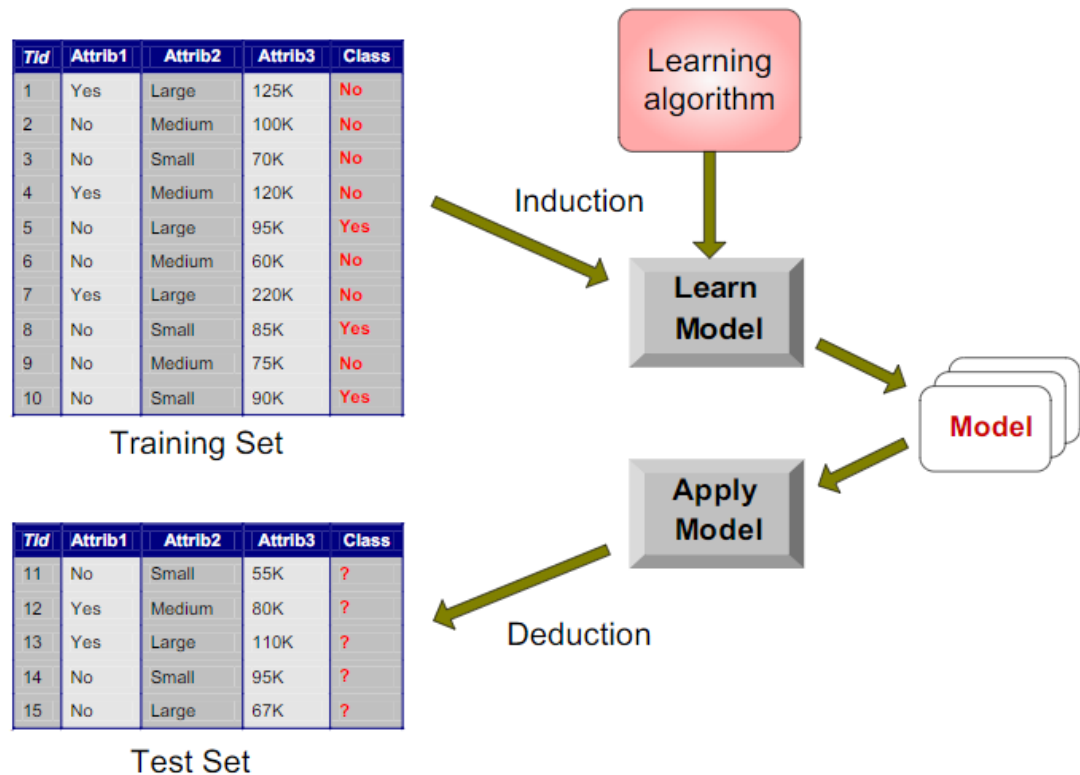


Figura 2:

Algoritmo ID3

Selezione una “window” (sottoinsieme del TS); Costruisci un DT usando i punti della window;
ID3 (idea)

Selezione una “window” (sottoinsieme del TS); Costruisci un DT usando i punti della window;
3.

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 atteso che il DT compie sui punti \notin TS. Viene stimato con il test set.

Indici di (in)purity Nella TDIDT “family” la costruzione dei DT, di norma si sceglie la **strategia greedy**: in ogni passo si sceglie il test che ottimizza 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 compare nell’insieme. Allora:

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

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

Gini Index

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

Entropia

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

Proof

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

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

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

□

Remark 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:

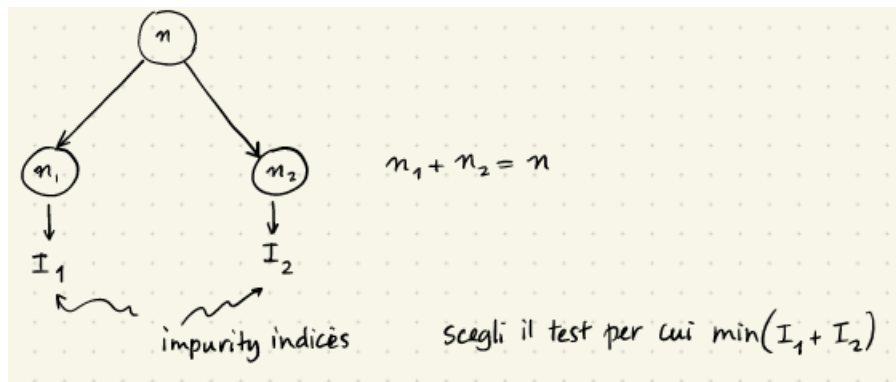


Figura 3:

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

- **ERR_TEST** = $\sum_{j \text{ figlio}} \text{ERR_CLASS}(j)$; si sceglie il test con indice minore

- **GINI_TEST** = $\sum_{j \text{ figlio}} \text{GINI}(j) \frac{n_j}{n}$ 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:

$$\text{INFORMATION_GAIN_TEST} = \text{ENTROPY}(\text{padre}) - \sum_{j \text{ figlio}} \left(\frac{n_j}{n} \right) \text{ENTROPY}(j)$$

Si sceglie il test con indice maggiore;

- **GAIN_RATIO_TEST** = equivale all'information gain test, ma con un andamento uniforme:

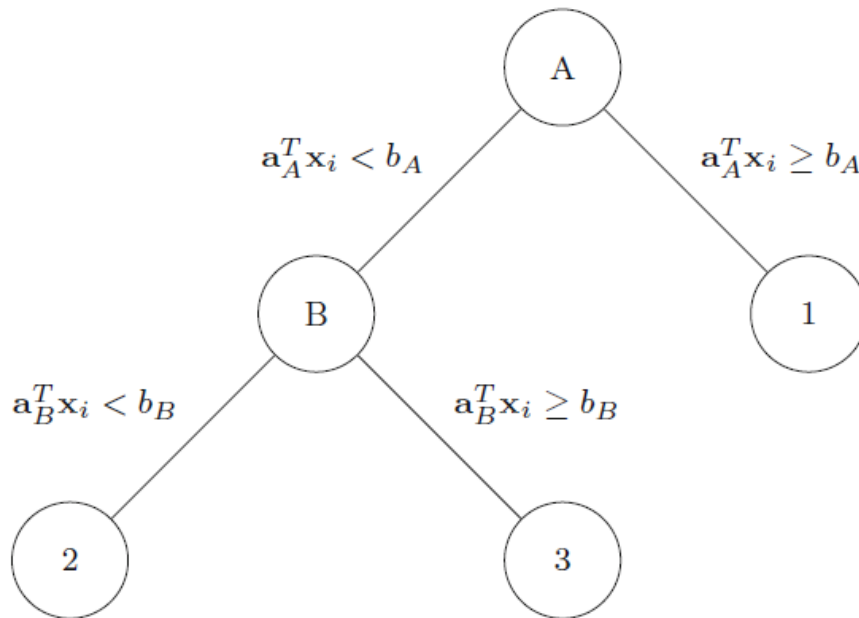
$$\text{GAIN_RATIO_TEST} = \frac{\text{INFORMATION_GAIN_TEST}}{\left[- \sum_{j \text{ figlio}} \left(\frac{n_j}{n} \right) \log \left(\frac{n_j}{n} \right) \right]}$$

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.

Example 1



Assumiamo:

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

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 a e b . Per un dato punto i , $a^T x_i < b$ il punto seguirà la diramazione sinistra del nodo, altrimenti la destra;
- I **nodi foglia** sono assegnati ad una classe che determinerà la predizione di tutti i punti che cadono all'interno della foglia. La classe viene assegnata in base alla moda della foglia.

Decisioni del problema Nella creazione dell'albero, in ogni iterazione, dobbiamo effettuare un numero fissato di decisioni:

- In ogni nodo dobbiamo decidere se dividere o fermarci;

- 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;
- Quando classifichiamo i punti di training 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 massima profondità di D . Dato questo parametro:

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

Notation 1

- $t \in \{T_B \cup T_L\} \setminus \{1\}$;
- $p(t) := \text{nodo padre/genitore}$;
- $A(t)$ è l'insieme degli antenati 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 $t \in T_B = \{1, 2, \dots, \lfloor \frac{T}{2} \rfloor\}$ che applicano la divisione nella forma $a^T x < b$ se lo soddisfano t seguirà il ramo sinistro, altrimenti il destro.

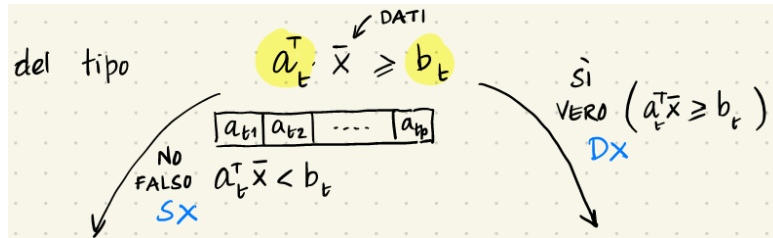


Figura 4:

Leaf Nodes: Sono tutti i nodi $t \in T_L = \{\lfloor \frac{T}{2} \rfloor, \dots, T\}$ 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 $\longrightarrow a_t \in \{0, 1\}^p$ $\bar{x} \in [0, 1]^p$ $\sum_{i=1}^p a_t = 1$
 $(a_t \cdot \bar{x}) \in [0, 1]$
- $b_t \in \mathbb{R} \longrightarrow b_t \in [0, 1]$ è uno scalare
- $d_t \in \{0, 1\}$ variabile che indica se si effettua o no un test $d_t \in \{0, 1\}$

Note 1

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

$$\sum_{i=1}^p a_{t,i} = d_t \quad \forall t \in T_B \quad \text{se effettuo o no il test}$$

$$0 \leq b_t \leq d_t \quad \forall t \in T_B$$

$$d_t \leq d_{p(t)} \quad \forall t \in T_B \quad \text{rappresenta la coerenza con } d_t \text{ se non si effettua lo SPLIT no split .}$$

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 $z_{it} = \{0, 1\}$ $i = 1, \dots, n, t \in T_L$ 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

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

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:

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

Note 2 Il vincolo ??(9) non è supportato dai risolutori. Quindi occorre convertirlo in una forma che non utilizza la disuguaglianza stretta. A tale scopo, aggiungiamo un fattore piccolo ε e il vincolo diventa:

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

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

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

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

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

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

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

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

- Il massimo valore di $a_m^T(x_i + \varepsilon) - b_m$ è $1 + \varepsilon_{\max}$ in cui $\varepsilon_{\max} = \max_j \{\varepsilon_j\}$. Quindi vogliamo:

$$M_1 \geq 1 + \varepsilon_{\max}$$

- Il massimo valore di $b_t - a_t^T x_i$ è 1 e quindi possiamo scegliere:

$$M_2 \geq 1$$

Ottenendo, in conclusione gli ultimi due vincoli:

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

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

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:

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

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

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

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

Task - obiettivo Vogliamo assegnare alle foglie $t \in T_L$ con $l_t=1$ l'etichetta a lui associata. Essa è la moda 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:

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

Al fine 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 sola 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 più 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:

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

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 nell'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:

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

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

Remark 2

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 soluzione di **warm start** il risolutore incrementa notevolmente la velocità con cui è in grado di generare soluzioni fortemente ammissi-

bili. 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 svantaggioso eseguire un problema MIO (Mixed-Integer-Optimization) con una piccola profondità per generare un forte warm start.

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

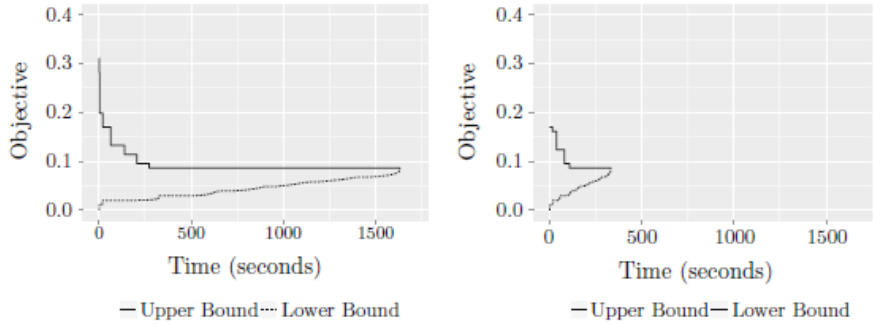


Figura 5: A sinistra senza Warm star; a destra con

Come 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, α per minimizzare l'**errore di training** e la **complessità dell'albero**

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

Remark 3

- α := è il fattore di importanze se è =0 → overfitting

Note 3

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

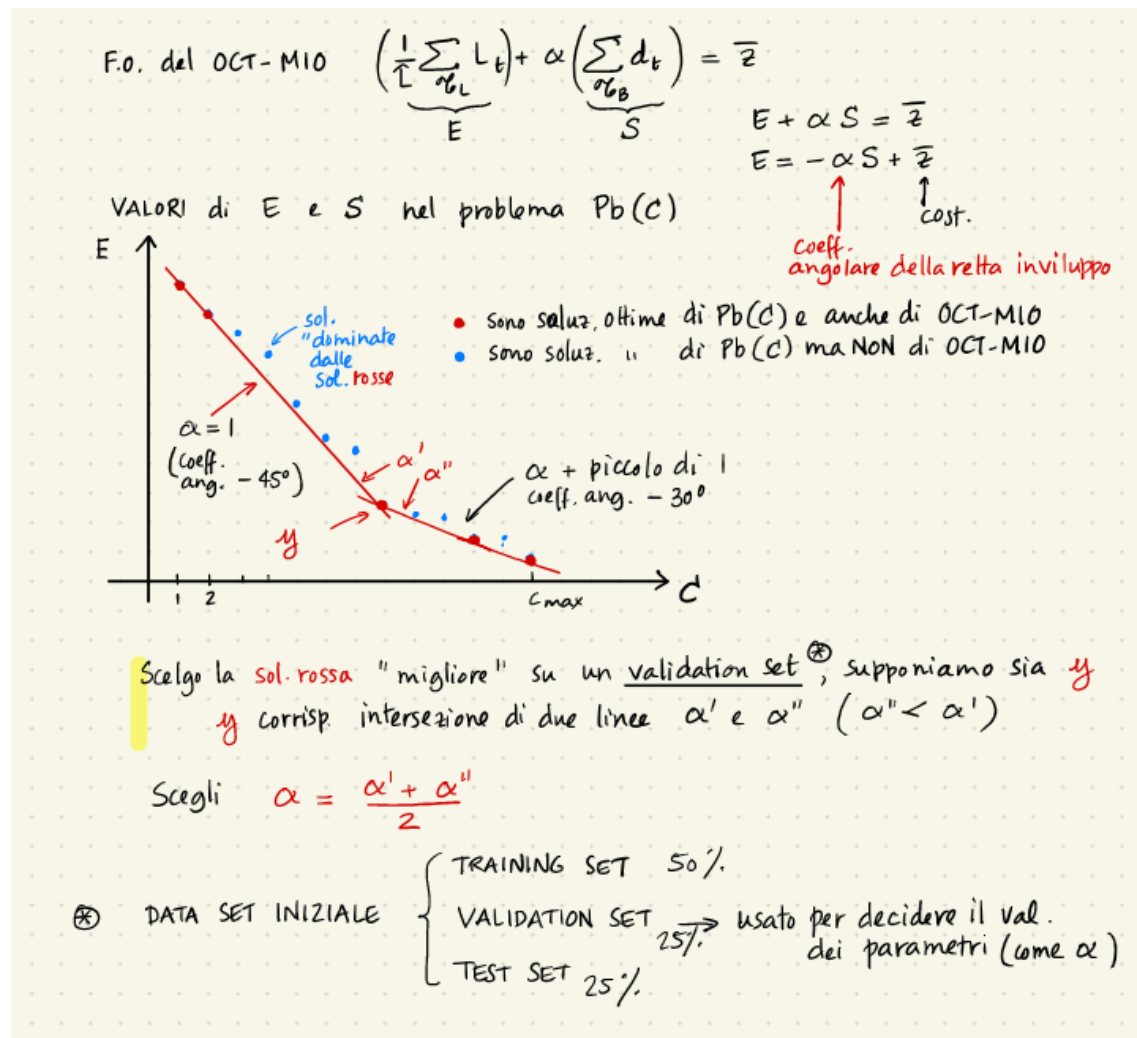
Nell'addestramento di un OCT occorre scegliere la **prodondità** dell'alber D_{\max} da cui si genereranno da $D = 1$ (3 nodi) fino a D_{\max} .

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

$$\min E$$

$$S \leq C$$

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 $D=1, \dots, D_{\max}$ do:

- For $C = 1, \dots, 2^D - 1$ 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 nodo. 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 vincolati a scegliere una singola variabile negli split, 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 ogni nodo ed in particolare $\in [-1, 1]^p$:

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

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

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

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

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

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

Quindi:

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

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

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

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

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

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

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

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

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

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

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

Note 4

Nel caso in cui $a_{jt} \neq 0 \rightarrow s_{jt} = 1$

Nel caso in cui $a_{jt} = 0 \rightarrow s_{jt} = 0$

Linearizzando il vincolo non lineare:

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

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

$$s_{jt} \leq d_t$$

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

Quindi la funzione obiettivo nel caso OCT-H diventa:

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

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 iperparametri si utilizza la stessa procedura di OCT con la differenza che:

$$\begin{aligned}\text{OCT} &\longrightarrow C_{\max} = 2^D - 1 \\ \text{OCT} - H &\longrightarrow C_{\max} = p(2^D - 1)\end{aligned}$$

Classificazione Robusta

7 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 robusto 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**.

8 Robust Classification

Il problema di ottimizzazione è del tipo:

$$\begin{aligned}\max c(x, y) \\ g(x, u) &\leq 0^m \\ x &\in X\end{aligned}$$

In cui:

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

I **parametri** u possono essere:

- fissati \rightarrow problema di ottimizzazione incerti;
- incerti $\rightarrow u \in U$ insieme di incertezza

.Cerchiamo la soluzione ottima nel caso peggiore.

9 Approccio MaxMin

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

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

Definition 1 *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 genera $f = a^T x$

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

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

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

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

10 Uncertainty Set

L'**insieme di incertezza** può essere utilizzato per modellare l'**incertezza delle feature**. Poiché l'incertezza nelle feature è dovuto a mancanza di 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:

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

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

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

A questo punto, possiamo definire un insieme delle perturbazioni:

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

Ottenendo:

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

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

11 Modello MIP (Programmazione Intera Mista)

Se consideriamo un albero binario non completo, esso sarà 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 $k = 1, 2, \dots, \lfloor \frac{T}{2} \rfloor, \lceil \frac{T}{2} \rceil, \dots, T$.

Al posto delle variabili $d_t = \begin{cases} 1 & \text{split nel nodo } t \\ 0 & \text{altrimenti} \end{cases}$, si utilizza:

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

11.1 Vincoli Strutturali

1. $\delta_k = 1 \quad k = \lceil \frac{T}{2} \rceil, \dots, T$;
2. $\delta_k \geq \delta_n \quad k = 1, \dots, T : u \in A(k)$;
3. $\delta_k + \sum_{i=1}^P a_{k,i} = 1 \quad k = 1, 2, \dots, T$

11.2 Allocazione Punti nelle Foglie

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

Remark 4

$$z_{ik} = 1 \iff \text{assegno punto } i \text{ al nodo } k$$

Vincoli

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

11.3 Funzione Obiettivo

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

Variabili

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

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

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

Vincoli:

$$(\alpha) f_k \leq g_k + M[w_k + (1 - l_k)] \quad (\gamma) f_k \geq -M[1 - w_k + (1 - l_k)]$$

$$(\beta) f_k \leq h_k + M[1 - w_k + (1 - l_k)] \quad (\varepsilon) f_k \geq h_k - M[w_k + (1 - l_k)]$$

Se $l_k = 0 \rightarrow$ vincoli soddisfatti

Se $l_k = 1$:

$$w_k \in \{0, 1\} \quad k = 1, \dots, T$$

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

$$h_k \leq f_k \leq g_k$$

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

$$g_k \leq f_k \leq h_k$$

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

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

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

12 Robustezza vs. Incertezza delle feature