

# Introduzione ad Alberi di Regressione

Programmazione di Applicazioni Data Intensive

Laurea in Ingegneria e Scienze Informatiche  
DISI - Università di Bologna

Alberi di Decisione per la regressione, test t-Student,  
Foreste di Decisione, XGBoost

Gianluca Moro

Dipartimento di Informatica – Scienza e Ingegneria  
Università di Bologna  
Via dell’Università, 50 – I-47522 Cesena (FC)  
[nome.cognome@unibo.it](mailto:nome.cognome@unibo.it)

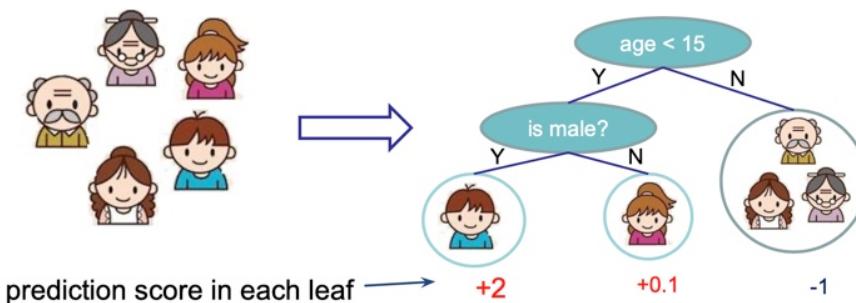
(draft)



*Introduzione ad Alberi di Regressione*

## Alberi di Regressione: Esempio

- per ogni persona sappiamo *età, genere, occupazione* etc. (variabili di input X) e quanto apprezza (Y output) i videogame
- L’obiettivo è creare una struttura ad albero dove
  - ogni nodo intermedio contiene un predicato con una variabile di input
  - il predicato divide le istanze di input in 2 sotto insiemi a cui corrispondono 2 nodi figli con relativi sotto alberi e così via ricorsivamente (e.g. binario)
  - ogni nodo foglia contiene tutte le istanze con apprezzamento Y più simile possibile tra esse, i.e. minimizzando l’errore di regressione in ogni foglia



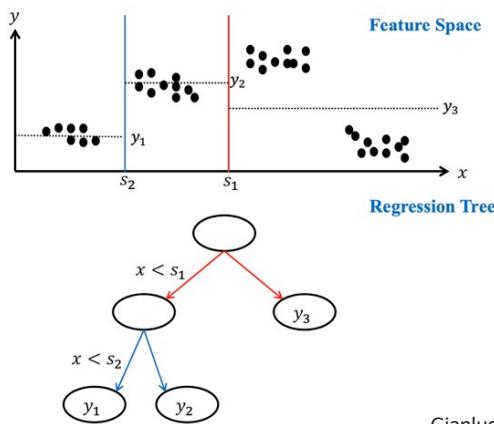
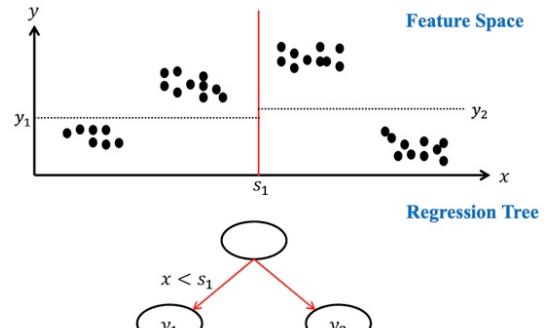
Come scegliere il predicato per ogni nodo ?

Qual è il criterio di arresto della divisione per ogni nodo ?



# Alberi di Regressione: Divisione dello Spazio (i)

- limitiamoci ad un dataset con una sola variabile  $X$  di input ed albero bin.
- la divisione della radice, retta rossa, data dal predicato  $x < s_1$  divide in 2 lo spazio in modo t.c. l'errore totale di regressione delle istanze rispetto a  $y_1$  e  $y_2$  sia minimo



- i sottospazi in cui l'errore è insoddisfacente si dividono ricorsivamente con lo stesso criterio;
- la figura di sinistra mostra la divisione del sottospazio di sinistra con  $x < s_2$ , per cui di nuovo la somma dell'errore di regressione rispetto a  $y_1$  e  $y_2$  nel sottospazio diviso sia inferiore all'errore prima della divisione

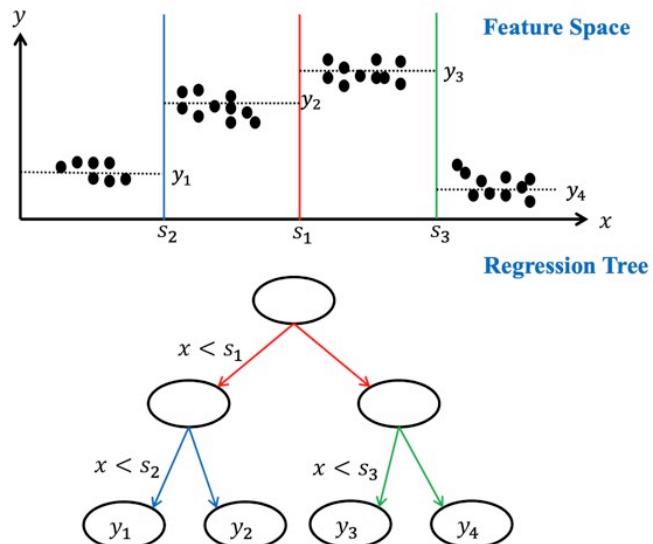
Gianluca Moro - DISI, Università di Bologna

3



# Alberi di Regressione: Divisione dello Spazio (ii)

- Analogamente viene diviso il sottospazio di destra, retta verde, con  $x < s_3$
- scegliendo  $s_3$  in modo tale che diminuisca l'errore dato dalla somma dell'errore di regressione nei due sottospazi divisi dalla retta verde, rispetto ai valori  $y_3$  e  $y_4$
- Nell'esempio  $y$  è l'intervallo di apprezzamento per i videogame ed  $x$  è una variabile delle istanze, e.g. l'età delle persone.



Gianluca Moro - DISI, Università di Bologna

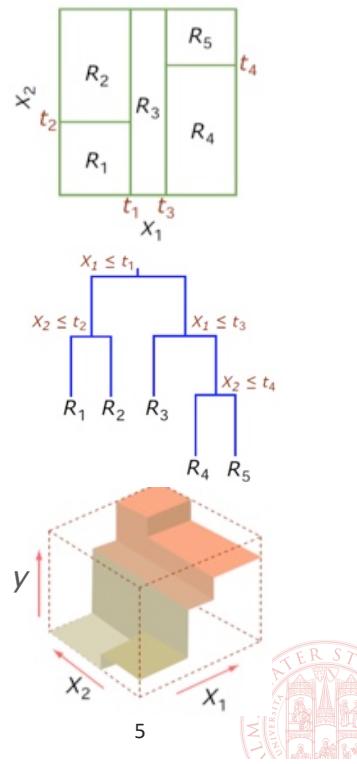
4



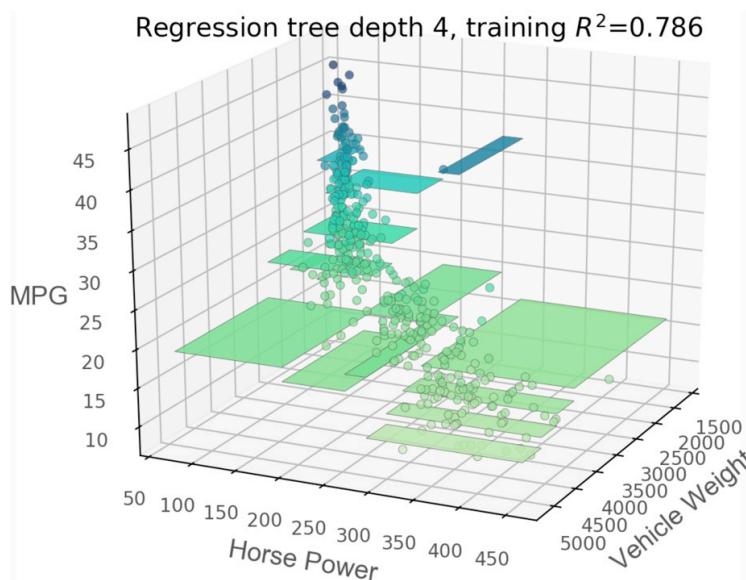
# Alberi di Regressione: Esempio in 2 Variabili (i)

- Esempio di partizionamento dello spazio dei dati con due variabili di input  $x_1, x_2$ 
  - la partizione  $t_1$  su  $x_1$  con  $x_1 \leq t_1$  crea 2 sottoalberi
  - $t_2$  eseguita su  $x_2$  nel sottoalbero sinistro con  $x_2 \leq t_2$  crea 2 sottospazi foglie  $R_1$  ed  $R_2$
  - partizione  $t_3$  su  $x_1$  del sottospazio destro di  $t_1$  con  $x_1 \leq t_3$  crea il sottoalbero con la foglia sinistra  $R_3$
  - infine partizione  $t_4$  del sottospazio destro di  $t_3$  su  $x_2$  con  $x_2 \leq t_4$  crea 2 sottospazi foglie  $R_4$  ed  $R_5$
- Visualizzazione del partizionamento in 3D
  - con i valori di regressione  $y$  predetti, per ciascuna partizione in base ai dati raccolti in ciascuna di esse
  - valori che corrispondono ad iperpiani ortogonali ad  $y$

Gianluca Moro - DISI, Università di Bologna



# Alberi di Regressione: Esempio in 2 Variabili (ii)



- previsione del consumo di carburante miles/gallon di veicoli dal peso e dalla potenza con albero di regressione

Gianluca Moro - DISI, Università di Bologna



# Alberi di Regressione: Algoritmo greedy di base

- sia  $k = 0$ ,  $S = \mathcal{R}^d$  lo spazio iniziale dei dati,  $d = \text{num. variabili di input}$

- def **RegressionTree**( $k, S$ )

$k += 1$

per ogni variabile di input  $j = 1, \dots, d$

divide lo spazio  $S$  scegliendo per  $j$  il valore di separazione  $v_j \in \mathcal{R}$  t.c.

$$S_{j,<} = \{ i \mid x_{i,j} < v_j \}; \quad S_{j,\geq} = \{ i \mid x_{i,j} \geq v_j \}$$

$$\bar{y}_{<} = \frac{\sum_{i \in S_{j,<}} y_i}{|S_{j,<}|}; \quad \bar{y}_{\geq} = \frac{\sum_{i \in S_{j,\geq}} y_i}{|S_{j,\geq}|}$$

$$MSE_j = \min \sum_{i \in S_{j,<}} (y_i - \bar{y}_{<})^2 + \sum_{i \in S_{j,\geq}} (y_i - \bar{y}_{\geq})^2$$

sia  $p$  la variabile in  $1, \dots, d$  che ha ottenuto il minimo  $MSE_p$

se la condizione\_di\_stop è insoddisfatta

restituisce **RegressionTree**( $k, S_{p,<}$ )  $\cup$  **RegressionTree**( $k, S_{p,\geq}$ );

altrimenti restituisce  $[(p, v_p)]$

- La condizione di stop è data da iperparametri: e.g. max profondità  $k$ , num. foglie, num. min di istanze per foglia, un valore minimo di  $MSE$  etc.



## Complessità del Modello ed Overfitting

- La complessità del modello aumenta all'aumentare della profondità dell'albero di regressione
  - Regolare la complessità è necessaria per adattare il modello alla complessità dei dati, ma la complessità può causare overfitting
  - Regolare la complessità con la regolarizzazione, come abbiamo già visto nelle diverse soluzioni precedenti
- Aggiungiamo la dimensione dell'albero alla loss con peso dato dall'iperparametro  $C$  da ottimizzare con nested cross validation
  - $\min$  errore-di-traiing(Tree) +  $C \times$  dimensione(Tree)
- Strategie alternative di creazione dell'albero
  - lasciare crescere l'albero da zero e fermarlo quando la loss aumenta
  - lasciare crescere pienamente l'albero e poi potarlo partendo dalle foglie fino a quando la loss inizia ad aumentare
  - meglio la seconda perché meno sensibile a scelte errate di predicit



# Regole di Potatura (Pruning)

- Stop quando ogni foglia contiene una sola istanza
- Stop quando il numero di foglie è inferiore ad una soglia o quando l'errore nella foglia è inferiore ad una soglia
- Stop quando il numero di istanze in ogni foglia è inferiore ad una soglia
- Stop quando il p-value nella divisione di un nodo in 2 foglie è maggiore di una soglia (e.g. 0.05) in base ad un test statistico
  - E.g. student t-test dei due gruppi, per stabilire se le due partizioni di dati del nodo sono diversi con significatività statistica data dal p-value, e.g. < 0.05
- Albero di classificazione: Stop quando tutte le istanze in ogni foglia hanno la stessa label



## Alberi: Potatura con t-test Student su 2 insiemi

- test d'ipotesi statistico per decidere se la differenza tra gruppi di dati è significativa o frutto del caso
  - ipotesi da testare, i.e. ipotesi nulla: le medie di 2 gruppi, i.e. insiemi, di dati sono equivalenti
  - t-score (i.e. t-value) è il rapporto tra la differenza tra i due gruppi di dati e la differenza interna a ciascun gruppo
  - maggiore è t-score, minore è la probabilità, i.e. p-value, che la differenza tra i due gruppi sia frutto del caso
  - p-value è calcolato con t-test Student
- fissiamo una soglia di accettazione dell'ipotesi nulla, e.g. 0.05
  - se il corrispondente p-value è superiore alla soglia di accettazione allora assumiamo che la differenza tra i due gruppi sia frutto del caso
  - quindi accettiamo l'ipotesi nulla che i due gruppi siano equivalenti e potiamo il sottoalbero riunendo le due foglie in un nodo foglia



## t-test di Student su 2 insiemi: t-score e p-value

- siano 2 foglie  $S_{j,<}, S_{j,\geq}$  date dalla suddivisione rispetto a  $v_j \in \mathcal{R}$

$$S_{j,<} = \{ i \mid x_{i,j} < v_j \}; \quad S_{j,\geq} = \{ i \mid x_{i,j} \geq v_j \}; \text{ medie } \bar{y}_{<} = \frac{\sum_{i \in S_{j,<}} y_i}{|S_{j,<}|}; \quad \bar{y}_{\geq} = \frac{\sum_{i \in S_{j,\geq}} y_i}{|S_{j,\geq}|}$$

$$\text{varianze } \sigma_{\bar{y}_{<}}^2 = \frac{\sum_{i \in S_{j,<}} (y_i - \bar{y}_{<})^2}{|S_{j,<}|-1}; \quad \sigma_{\bar{y}_{\geq}}^2 = \frac{\sum_{i \in S_{j,\geq}} (y_i - \bar{y}_{\geq})^2}{|S_{j,\geq}|-1}$$

distrib.  
normale

$$\text{t-score} = \frac{\bar{y}_{<} - \bar{y}_{\geq}}{\sqrt{\frac{\sigma_{\bar{y}_{<}}^2}{|S_{j,<}|} + \frac{\sigma_{\bar{y}_{\geq}}^2}{|S_{j,\geq}|}}} \quad \text{p-value} = P(Z > |\text{t-score}|) \text{ con } Z \sim \mathcal{N}(0, 1)$$

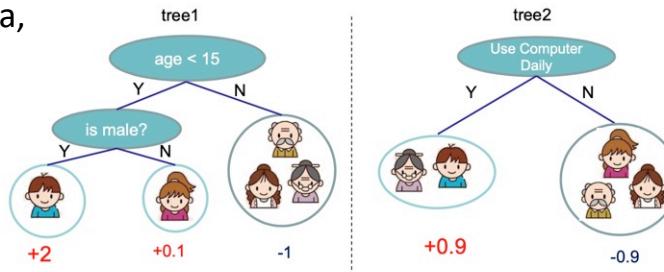
$$\text{Gradi di libertà } v = |S_{j,<}| + |S_{j,\geq}| - 2$$

- t-score segue approssimativamente la t-distrib. con  $v$  gradi di libertà
- E.g. siano 2 foglie con  $|S_{j,<}| = |S_{j,\geq}| = 30$ , t.c.  $\bar{y}_{<} = -1.2$ ;  $\bar{y}_{\geq} = -0.5$ ;  $\sigma_{\bar{y}_{<}}^2 = 23.2$ ;  $\sigma_{\bar{y}_{\geq}}^2 = 10.7$ ; **t-score** = -0.66, **p-value** = 0.51 > 0.05, i 2 gruppi sono equivalenti
- in python: `from scipy import stats ... t_score, p_value = stats.ttest_ind(S_{j,<}, S_{j,\geq})`



## Foreste di Regressione: Esempio

- in genere l'errore diminuisce combinando più alberi semplici creati sullo stesso dataset (*Ensemble Learning*), varie strategie:
  - utilizzando per ciascuno diversi sottoinsiemi casuali di dati non disgiunti, **bagging**, e sottoinsiemi casuali di variabili di input e.g. **random forest**
  - alberi in sequenza da errori residui del precedente: **gradient boosting**, **xgboost** ...
  - l'output è ottenuto combinando in varie modalità i singoli output
  - Esempio precedente in figura, 2 alberi con feature diverse
  - La previsione per ogni istanza è la somma delle regressioni predette da ciascun albero
  - Aumenta la complessità, diminuisce l'interpretabilità

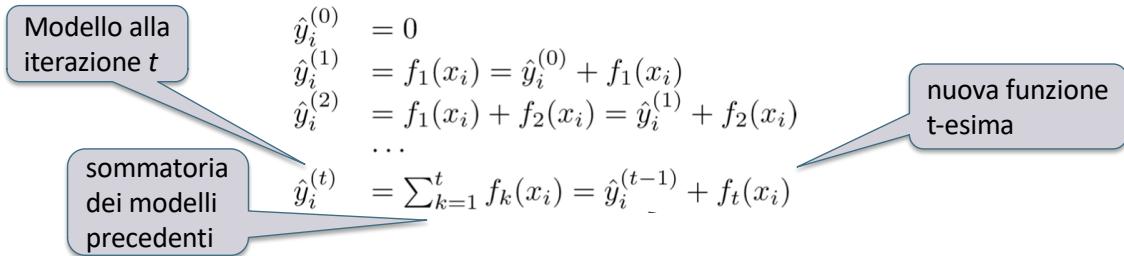


$$f(\text{boy}) = 2 + 0.9 = 2.9 \quad f(\text{old man}) = -1 - 0.9 = -1.9$$



# Gradient Boosting Machine: Learning Additivo di più Modelli

- Training additivo di modelli, da qui *boosting*
  - inizia con un modello a previsione costante, e.g. che produce sempre 0, e ad ogni iterazione  $k$  somma un nuovo modello  $\hat{y}_i^{(k)} = f_k(x_i)$  al precedente



- ogni albero aggiunto è addestrato dal dataset degli errori residui del precedente, i.e. dalle coppie  $(x_i, y_i - \hat{y}_i^{(k-1)})$  con  $y_i$  il valore da predire
- Loss da minimizzare  $L(y, \hat{y}^{(t)}) = \frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i^{(t)})^2$   $N$  num. istanze

Gianluca Moro - DISI, Università di Bologna

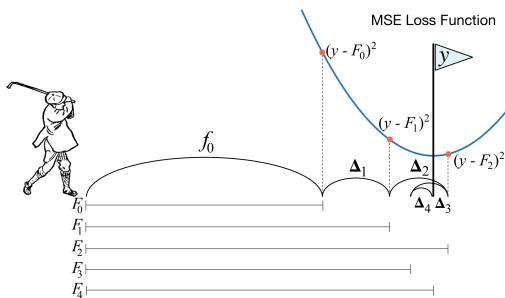
13



## Gradient Boosting vs Discesa del Gradiente

- minimizziamo  $L(y, \hat{y}^{(t)})$  rispetto ad uno specifico  $\hat{y}_j^{(t)}$ 
  - $\frac{\partial L(y, \hat{y}^{(t)})}{\partial \hat{y}_j^{(t)}} = \frac{\partial \sum_{i=1}^N (y_i - \hat{y}_i^{(t)})^2}{\partial \hat{y}_j^{(t)}}$  Eliminato  $N$  dalla sommatoria, il num. di istanze, poiché è costante e  $\text{argmin}_x c \cdot f(x) = f(x)$
  - $= \frac{\partial}{\partial \hat{y}_j^{(t)}} (y_j - \hat{y}_j^{(t)})^2$  Eliminata la sommatoria poiché le derivate parziali di  $L$  con  $i \neq j$  valgono zero
  - $= -2(y_j - \hat{y}_j^{(t)})$
  - $\frac{\partial L(y, \hat{y}^{(t)})}{\partial \hat{y}^{(t)}} = y - \hat{y}^{(t)}$

gradiente rispetto al modello t-esimo  $\hat{y}^{(t)}$  per tutte le istanze, con la costante 2 analogamente eliminata



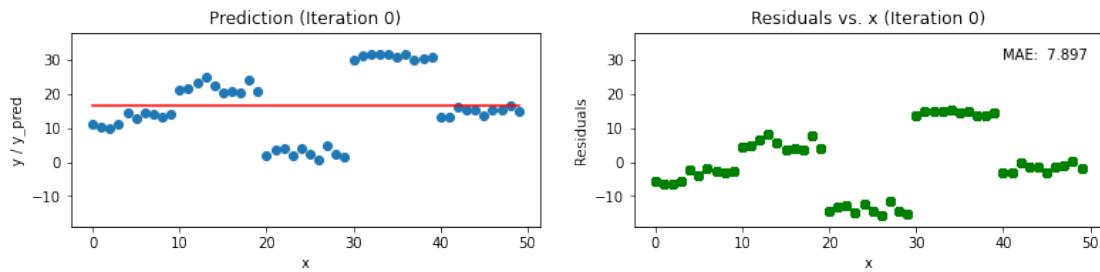
il modello GBM è una funzione ricorrente  
 $\hat{y}^{(t)} = \hat{y}^{(t-1)} + f_t(x)$   
l'aggiunta di un nuovo modello  $f_t(x)$  ad ogni iterazione  $t$  riduce il residuo, i.e. gradiente,  $y - \hat{y}^{(t)}$   
infatti la loss è  $L(y, \hat{y}^{(t-1)} + f_t(x))$   
con discesa del gradiente con  $\eta$  learning rate  
 $\hat{y}^{(t)} = \hat{y}^{(t-1)} + \eta(-\nabla L(y, \hat{y}^{(t-1)} + f_t(x)))$

Gianluca Moro - DISI, Università di Bologna

14



# Esempio di Gradient Boosting: Stato Iniziale (i)



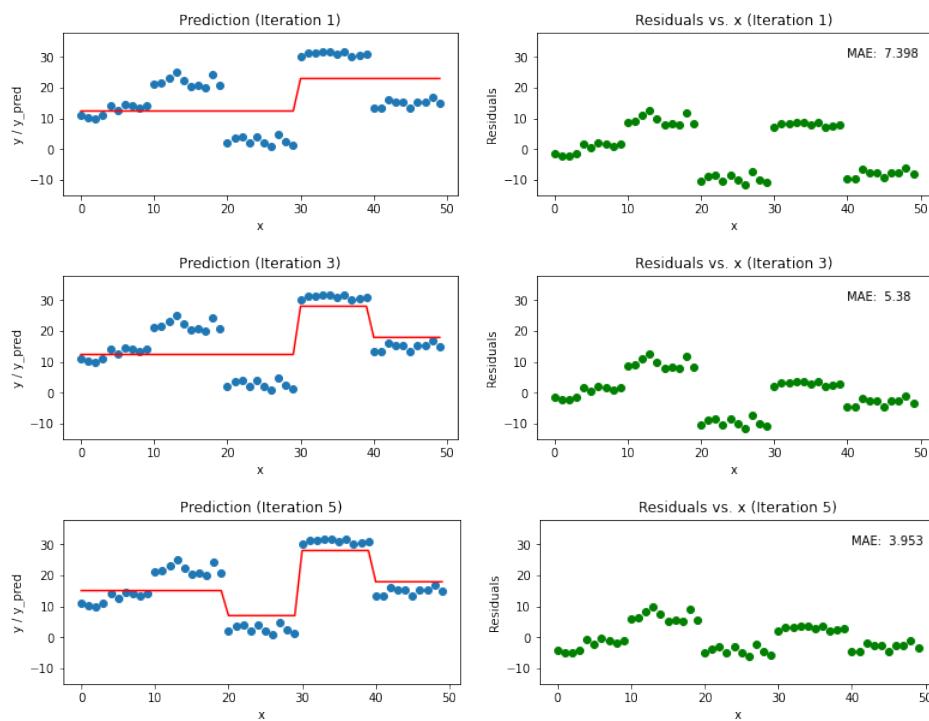
- La figura di sinistra mostra il dataset univariato con istanze  $(x_i, y_i)$ , dove  $x_i$  è la variabile di input ed  $y_i$  la variabile target da predire
- Inizialmente, prima dello split, il modello di previsione  $\hat{y}_i^{(0)} = \bar{y}_i$  i.e. valor medio delle istanze  $y_i$  (retta rossa), ma può essere qualsiasi valore, anche 0
- La figura di destra mostra gli errori residui  $y_i - \hat{y}_i^{(0)}$  per ogni istanza  $i$ : l'errore assoluto medio MAE è 7.897, idealmente dovrebbe essere 0
- L'iterazione successiva genera un albero  $\hat{y}_i^{(1)}$  dalle istanze  $(x_i, y_i - \hat{y}_i^{(0)})$ , i.e. dagli errori residui, ed il nuovo modello di previsione diventa  $\hat{y}_i^{(0)} + \hat{y}_i^{(1)}$

Gianluca Moro - DISI, Università di Bologna

15



# Esempio: Iterazioni e Alberi da 1 a 5 (ii)



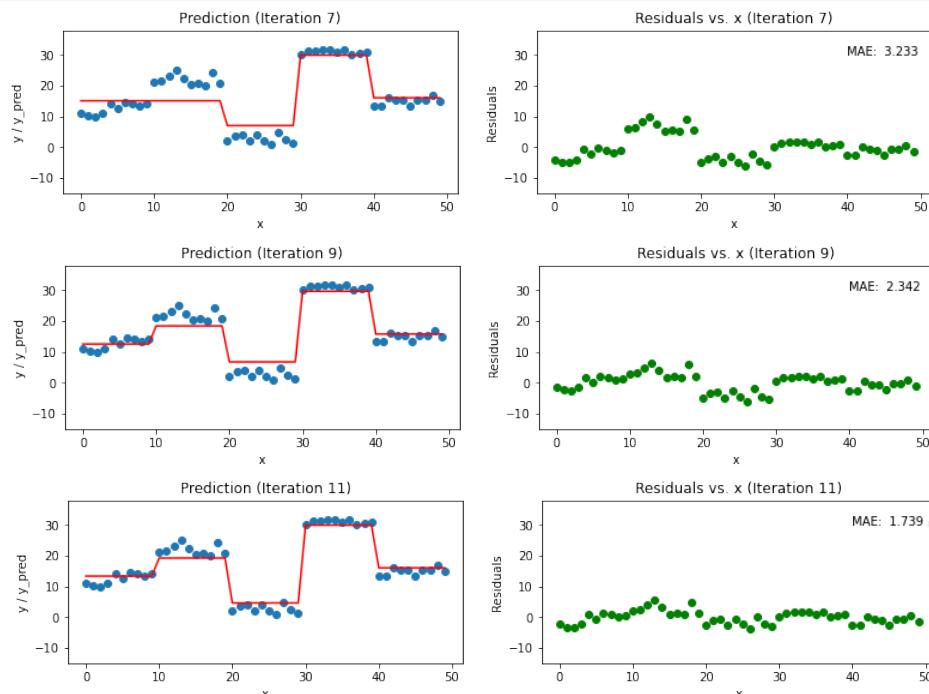
I residui si distribuiscono intorno allo zero in modo NON casuale, quindi c'è margine per creare nuovi alberi che modellino la loro distribuzione non casuale

Gianluca Moro - DISI, Università di Bologna

16



## Esempio: Iterazioni e Alberi da 7 a 11 (iii)

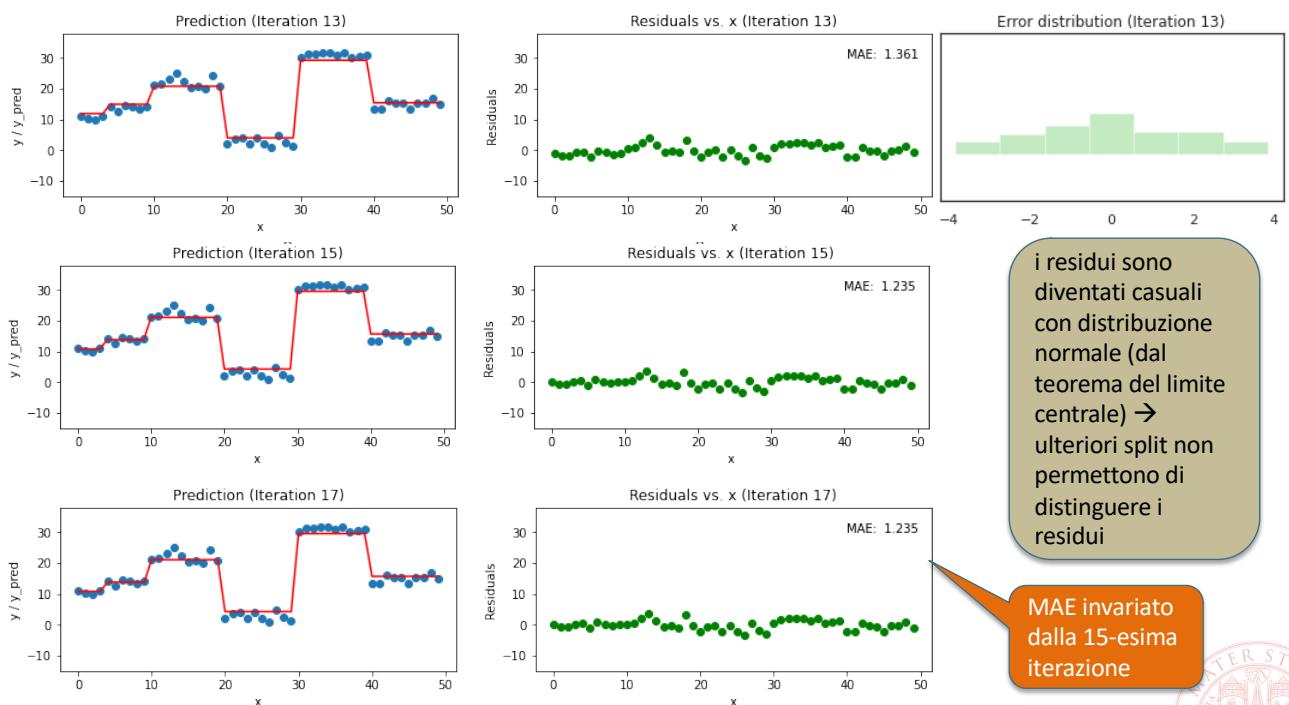


Gianluca Moro - DISI, Università di Bologna

17



## Esempio: Iterazioni e Alberi da 13 a 17 (iv)



Gianluca Moro - DISI, Università di Bologna

18



# Supervised Learning di un Singolo Modello

- obiettivo del supervised learning di un singolo modello
  - determinare dai dati  $x_i$  una funzione  $f$  parametrica di regressione per predire i corrispondenti  $\hat{y}_i$   $\hat{y}_i = f(x_i)$
  - con un modello lineare con parametri  $w_j$  si ha  $\hat{y}_i = \sum_j w_j x_{ij}$
  - con  $d$  parametri  $w_j$   $\Theta = \{w_j | j = 1, \dots, d\}$
- funzione obiettivo da minimizzare: loss  $L$  + la regolarizzazione  $\Omega$

$$Obj(\Theta) = L(\Theta) + \Omega(\Theta)$$

- dove  $L(\Theta) = \sum_i (\hat{y}_i - y_i)^2$ ,  $\Omega(\Theta) = \lambda \|w\|_2^2$
- In  $L$  manca la divisione per  $N$ , il numero di istanze, perché  $\operatorname{argmin}_x c \cdot f(x) = f(x)$  quando  $c$  è costante
- loss  $L$  definita con la tecnica dei minimi quadrati
  - per determinare i parametri  $w$  che minimizzano l'errore di regressione
- regolarizzazione  $\Omega$  definita dal quadrato della norma L2 pesata per l'iperparametro lambda
  - per ridurre, in valore assoluto, il valore dei  $w$  e semplificare il modello

Gianluca Moro - DISI, Università di Bologna

19



# Supervised Learning Additivo di più Alberi

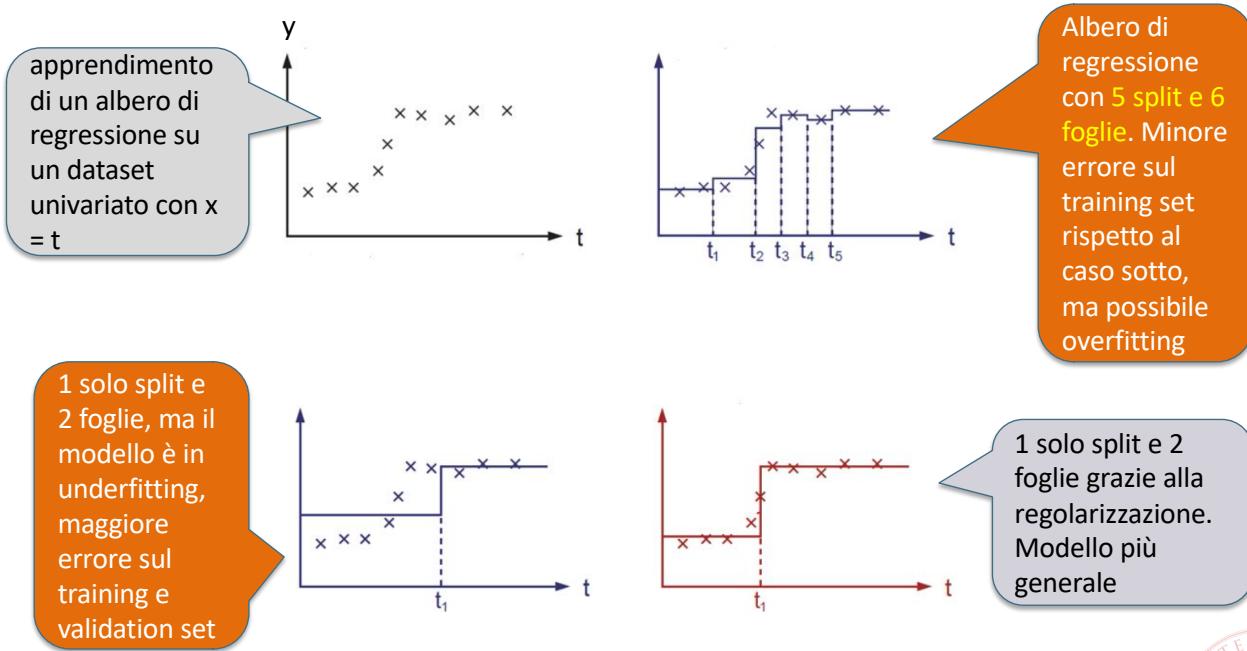
- regressione con  $K$  alberi
- $\hat{y}_i = \sum_{k=1}^K f_k(x_i), \quad f_k \in \mathcal{F}$ 
  - Spazio (foresta) degli alberi di regressione
- dove ogni  $f_k$  è un albero da apprendere e la previsione  $\hat{y}_i$  è data dalla sommatoria delle previsioni dei  $K$  alberi
- la loss è la somma delle loss e delle regolarizzazioni dei  $K$  alberi

$$Obj = \sum_{i=1}^n l(y_i, \hat{y}_i) + \sum_{k=1}^K \Omega(f_k)$$

- La minimizzazione delle loss  $l$  riduce l'errore sul training set
- La regolarizzazione  $\Omega$  riduce la complessità degli alberi e tende a rendere i modelli più generali e stabili sui nuovi dati
  - e.g. validation e test set, e sui dati a regime dopo il deployment del modello



# Regolarizzazione: Overfitting e Underfitting



Gianluca Moro - DISI, Università di Bologna

21

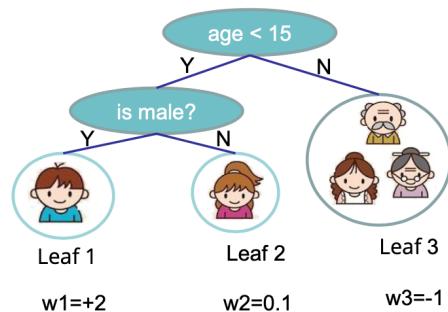


## Definizione della Complessità di un Albero

- Gli split aumentano la complessità perché generano più foglie e aumentano la profondità dell'albero
- Con la regolarizzazione vogliamo controllarne la complessità
- Sia  $f_t(x) = w_{q(x)}$ ,  $w \in \mathbf{R}^T$ ,  $q : \mathbf{R}^d \rightarrow \{1, 2, \dots, T\}$ 
  - dove  $f_t$  è il  $t$ -esimo albero di regressione con  $w_{q(x)}$  coefficienti di regressione,  $T$  numero di foglie,  $q$  funzione di struttura dell'albero,  $d$  dimensione dei dati di input  $x$
- Esempio con il seguente albero di regressione
  - $q(x_i)$  assegna ogni istanza  $x_i$  ad una foglia: la 1a istanza qui alla 1a foglia e la 2a alla terza foglia

$$\begin{aligned} q(\text{boy}) &= 1 \\ q(\text{girl}) &= 3 \end{aligned}$$

Gianluca Moro - DISI, Università di Bologna



22



# Complessità degli Alberi in XGBoost:

## Extreme Gradient Boosting

- Definizione della complessità

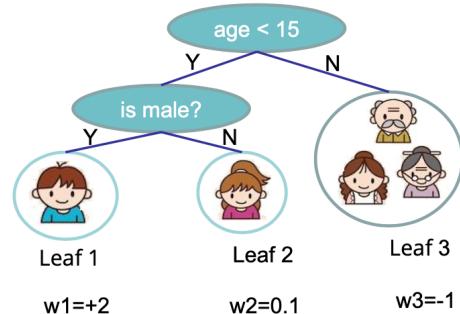
$$\Omega(f_t) = \gamma T + \frac{1}{2} \lambda \sum_{j=1}^T w_j^2$$

- dove  $f_t$  è il t-esimo albero di regressione
- gamma è l'iperparametro relativo al numero di foglie
- lambda l'iperparametro per la regolarizzazione delle regressioni

- Per l'albero di esempio con

- $T = 3$  foglie
- $w_1 = 2, w_2 = 0.1, w_3 = -1$
- otteniamo

$$\Omega = \gamma 3 + \frac{1}{2} \lambda (4 + 0.01 + 1)$$



Gianluca Moro - DISI, Università di Bologna

23



# XGBoost: Apprendere una Foresta di Alberi

- Loss obiettivo

$$\sum_{i=1}^n l(y_i, \hat{y}_i) + \sum_k \Omega(f_k), f_k \in \mathcal{F}$$

- minimizzazione della somma delle loss  $l \rightarrow$  riduce l'errore sul training set
- la regolarizzazione  $\Omega$  riduce la complessità dei  $K$  alberi  $f_k$
- loss obiettivo non differenziabile, i.e. derivabile
- training additivo di modelli come in GBM, da qui *boosting*
  - inizia con una funzione a previsione costante, e.g. restituisce zero, ed aggiunge ad ogni iterazione una nuova funzione

Modello alla iterazione  $t$ 

$$\begin{aligned}\hat{y}_i^{(0)} &= 0 \\ \hat{y}_i^{(1)} &= f_1(x_i) = \hat{y}_i^{(0)} + f_1(x_i) \\ \hat{y}_i^{(2)} &= f_1(x_i) + f_2(x_i) = \hat{y}_i^{(1)} + f_2(x_i) \\ &\dots \\ \hat{y}_i^{(t)} &= \sum_{k=1}^t f_k(x_i) = \hat{y}_i^{(t-1)} + f_t(x_i)\end{aligned}$$

nuova funzione t-esima

sommatoria dei modelli precedenti

Gianluca Moro - DISI, Università di Bologna

24



## XGBoost: Training Additivo

- decide quale funzione aggiungere ottimizzando la loss obiettivo
- la previsione all'iterazione  $t$  è  $\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + f_t(x_i)$  con  $f_t$  la nuova funzione da apprendere

$$Obj^{(t)} = \sum_{i=1}^n l(y_i, \hat{y}_i^{(t)}) + \sum_{i=1}^t \Omega(f_i)$$

- riscritta per esplicitare nella loss  $l$  il modello all'iterazione  $t-1$  e la  $f_t$  da minimizzare

$$Obj^{(t)} = \sum_{i=1}^n l\left(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)\right) + \Omega(f_t) + constant$$

- Con i minimi quadrati minimizza

$$\begin{aligned} Obj^{(t)} &= \sum_{i=1}^n \left( y_i - (\hat{y}_i^{(t-1)} + f_t(x_i)) \right)^2 + \Omega(f_t) + const \\ &= \sum_{i=1}^n \left[ 2(\hat{y}_i^{(t-1)} - y_i)f_t(x_i) + f_t(x_i)^2 \right] + \Omega(f_t) + const \end{aligned}$$

Chiamato residuo dalla iterazione precedente

sommatoria di regolarizzazione fino a  $t-1$  sostituita da una costante

Gianluca Moro - DISI, Università di Bologna

25



## Approssimazione della Loss con Espansione di Taylor del Secondo Ordine

Opzionale

- Ricordiamo la funzione obiettivo

$$Obj^{(t)} = \sum_{i=1}^n l\left(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)\right) + \Omega(f_t) + constant$$

- Approssimazione della funzione obiettivo con Taylor

- Espansione di Taylor  $f(x + \Delta x) \simeq f(x) + f'(x)\Delta x + \frac{1}{2}f''(x)\Delta x^2$

- Definiamo con  $g_i$  e  $h_i$   $g_i = \partial_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$ ,  $h_i = \partial_{\hat{y}^{(t-1)}}^2 l(y_i, \hat{y}^{(t-1)})$

- La funzione obiettivo diventa

$$Obj^{(t)} \simeq \sum_{i=1}^n \left[ l(y_i, \hat{y}_i^{(t-1)}) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \right] + \Omega(f_t) + constant$$

- Con i minimi quadrati abbiamo

$$g_i = \partial_{\hat{y}^{(t-1)}} (\hat{y}^{(t-1)} - y_i)^2 = 2(\hat{y}^{(t-1)} - y_i) \quad h_i = \partial_{\hat{y}^{(t-1)}}^2 (y_i - \hat{y}^{(t-1)})^2 = 2$$

Risultato della derivata seconda

Gianluca Moro - DISI, Università di Bologna

27



# Nuova Funzione Obiettivo con Approssimazione di Taylor

Opzionale

- funzione obiettivo senza le costanti

$$\sum_{i=1}^n [g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + \Omega(f_t)$$

$$g_i = \partial_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)}), \quad h_i = \partial_{\hat{y}^{(t-1)}}^2 l(y_i, \hat{y}^{(t-1)})$$

- Definiamo l'insieme delle istanze nella foglia  $j$  come  $I_j$

$$\begin{aligned} Obj^{(t)} &\simeq \sum_{i=1}^n [g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + \Omega(f_t) \\ &= \sum_{i=1}^n [g_i w_{q(x_i)} + \frac{1}{2} h_i w_{q(x_i)}^2] + \gamma T + \lambda \frac{1}{2} \sum_{j=1}^T w_j^2 \\ &= \sum_{j=1}^T \left[ (\sum_{i \in I_j} g_i) w_j + \frac{1}{2} (\sum_{i \in I_j} h_i + \lambda) w_j^2 \right] + \gamma T \end{aligned}$$

Raggruppamento per foglie (T foglie)

- L'obiettivo è una somma di  $T$  funzioni quadratiche indipendenti



## Score della Complessità dell'Albero

Opzionale

- Due risultati noti per funzioni quadratiche in una variabile

$$\operatorname{argmin}_x Gx + \frac{1}{2} Hx^2 = -\frac{G}{H}, \quad H > 0 \quad \min_x Gx + \frac{1}{2} Hx^2 = -\frac{1}{2} \frac{G^2}{H}$$

È la  $x$  che minimizza l'espressione

- Definiamo  $G_j = \sum_{i \in I_j} g_i \quad H_j = \sum_{i \in I_j} h_i$

È il valore della espressione sostituendo la  $x$  che la minimizza

$$\begin{aligned} Obj^{(t)} &= \sum_{j=1}^T \left[ (\sum_{i \in I_j} g_i) w_j + \frac{1}{2} (\sum_{i \in I_j} h_i + \lambda) w_j^2 \right] + \gamma T \\ &= \sum_{j=1}^T [G_j w_j + \frac{1}{2} (H_j + \lambda) w_j^2] + \gamma T \end{aligned}$$

- Supponendo che la struttura dell'albero  $q(x)$  sia fissa, i pesi  $w^*$  ottimali in ogni foglia ed i valori risultanti della funzione sono

$$w_j^* = -\frac{G_j}{H_j + \lambda}$$

$$Obj = -\frac{1}{2} \sum_{j=1}^T \frac{G_j^2}{H_j + \lambda} + \gamma T$$

Misura della complessità dell'albero

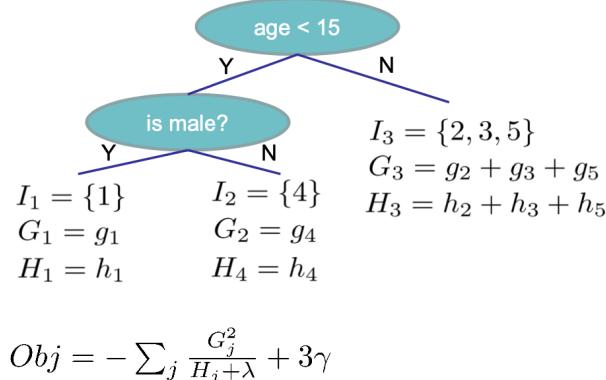


# Score della Complessità dell'Albero: Esempio

Opzionale

- 5 istanze, con indice da 1 a 5, e con  $I_j$  e valori  $G_j$  e  $H_j$  per ciascuna delle 3 foglie

1		$g_1, h_1$
2		$g_2, h_2$
3		$g_3, h_3$
4		$g_4, h_4$
5		$g_5, h_5$



Minore è lo score,  
migliore è la struttura  
dell'albero



## Algoritmo di Ricerca Greedy dell'Albero

Opzionale

- la ricerca inizia da un albero di profondità zero
- per ogni nodo foglia prova ad eseguire uno split creando due nodi, Left e Right, e calcola il guadagno ottenuto come

$$Gain = \frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{(G_L + G_R)^2}{H_L + H_R + \lambda} - \gamma$$

Guadagno  
prodotto dal  
nodo sinistro

Guadagno  
prodotto dal  
nodo destro

Guadagno  
senza eseguire  
lo split

Costo  
nell'aumento  
della complessità  
nell'aggiungere  
nuove foglie

- come trovare, per ogni foglia, il miglior split ?

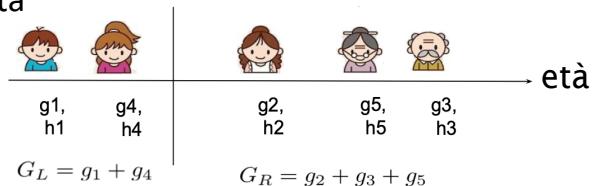


# Miglior Split, Regolarizzazione e Potatura

- Calcola il guadagno dato da uno split del tipo  $x_j < a$

nell'esempio l'asse  $x_j$  può essere l'età

- Con le istanze ordinate per età, applica da sinistra a destra una serie di split



- per ciascuno split calcola  $g$  ed  $h$  e li somma nelle rispettive due parti  $G_L$  e  $G_R$  e sceglie lo split che produce il maggior guadagno

$$Gain = \frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{(G_L + G_R)^2}{H_L + H_R + \lambda} - \gamma$$

- L'algoritmo termina con una soluzione di trade-off tra semplicità ed errore minimo

- Stop allo split quando il guadagno diventa negativo, anche se può portare a soluzioni migliori successive (approccio greedy)
- Post potatura: lascia crescere l'albero alla max profondità e ricorsivamente elimina gli split con guadagno negativo

Quando la riduzione della loss di training è inferiore alla regolarizzazione il guadagno è negativo

Gianluca Moro - DISI, Università di Bologna

32



## XGBOOST in Python

```
import xgboost as xgb

# read in data
dtrain = xgb.DMatrix('demo/data/agaricus.txt.train')
dtest = xgb.DMatrix('demo/data/agaricus.txt.test')

# specify parameters via map
param = {'max_depth':2, 'eta':1, 'silent':1, 'objective':'binary:logistic' }
num_round = 2
bst = xgb.train(param, dtrain, num_round)

# make prediction
preds = bst.predict(dtest)
```

Learning rate

Per la classificazione

Per la regressione MSE  
'objective':'reg:squarederror'

- determina anche l'importanza di feature, predice la direzione di discesa nell'albero di istanze con valori mancanti sulla feature di split, mostra la struttura dell'albero per interpretare la conoscenza appresa, automatizza la gestione di dati sparsi

<https://github.com/dmlc/xgboost>    <https://xgboost.readthedocs.io/en/latest/index.html>

Gianluca Moro - DISI, Università di Bologna

33

