

Gradient Boosting

- Disegnato inizialmente per problemi di classificazione, poi esteso anche a problemi di regressione.
- Idea: Assegna maggior peso alle osservazioni che sono state classificate male, per far lavorare maggiormente il classificatore su questi punti → AdaBoost
- Si mediano tanti alberi cresciuti su versioni pesate dei dati di stima

Gradient Boosting

- Potente algoritmo di machine learning
- Usato per problemi di regressione e classificazione
- Gradient Boosting = Gradient Descent + Boosting

Gradient Boosting: intuizione

Consideriamo un problema di **regressione** ... con un semplice caso:
 $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$.

Vogliamo stimare un modello $y = f(x)$ che minimizzi una **funzione di perdita**, ad esempio il Mean Squared Error.

Supponiamo di disporre di un buon modello f , ma notiamo alcuni errori:
 $f(x_1) = 0.8$ mentre $y_1 = 0.9$, $f(x_2) = 1.4$ mentre $y_2 = 1.3$.

Come migliorare il modello?

Gradient Boosting: intuizione

Teniamo conto che:

- non possiamo in alcun modo modificare f
- ma possiamo aggiungere ad f un altro modello, ad esempio un **albero di regressione**, h ,
- per cui la nuova previsione quindi sarà
$$y_i = f(x_i) + h(x_i)$$

Gradient Boosting: intuizione

La previsione viene dunque aggiornata in questo modo

$$\begin{aligned}f(x_1) + h(x_1) &= y_1 \\f(x_2) + h(x_2) &= y_2 \\&\vdots \\f(x_n) + h(x_n) &= y_n.\end{aligned}$$

Possiamo però anche scrivere

$$\begin{aligned}y_1 - f(x_1) &= h(x_1) \\y_2 - f(x_2) &= h(x_2) \\&\vdots \\y_n - f(x_n) &= h(x_n)\end{aligned}$$

dove $r(x_i) = y_i - f(x_i)$ sono i **residui**

Gradient Boosting: intuizione

- **Gradient Boosting** → stimare un albero di regressione, h , sui dati $(x_1, r_1), (x_2, r_2), \dots, (x_n, r_n)$ per migliorare la previsione
- il ruolo di h è di compensare i 'difetti' del modello f

Gradient Boosting: intuizione

A questo punto abbiamo un nuovo modello per la y , che dovrebbe essere migliore rispetto al precedente:

$$f_2(x) = f_1(x) + h_1(x)$$

e possiamo ripetere il ragionamento ottenendo i residui rispetto a questo nuovo modello f_2 e stimare un nuovo albero h_2 per migliorare ulteriormente la previsione.

Ora quindi la previsione sarà

$$f_3(x) = f_2(x) + h_2(x)$$

Possiamo ripetere il ragionamento M volte e ad ogni iterazione $1 < m < M$ avremo

$$f_{m+1}(x) = f_m(x) + h_m(x)$$

Ma come questo si collega con il Gradient Descent?

Gradient Boosting

Come questo si collega con il Gradient Descent?

Consideriamo la funzione di perdita quadratica

$$L(y, f(x)) = \frac{1}{2}(y - f(x))^2$$

Vogliamo minimizzare $J = \sum_i L(y_i, f(x_i))$

$$\frac{\partial J}{\partial f(x_i)} = \frac{\partial \sum_i L(y_i, f(x_i))}{\partial f(x_i)} = \frac{\partial L(y_i, f(x_i))}{\partial f(x_i)} = f(x_i) - y_i$$

Possiamo quindi interpretare i residui come gradienti negativi

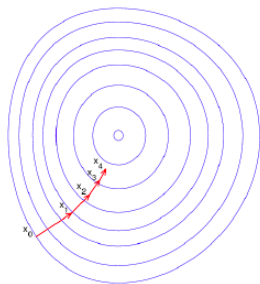
$$-g(x_i) = - \left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)} \right] = y_i - f(x_i)$$

Gradient Boosting

Gradient Descent

Minimizza una funzione muovendosi nella **direzione opposta al gradiente**.
A partire da una soluzione iniziale ϑ_0 , si procede iterativamente con aggiornamento

$$\vartheta_{m+1} = \vartheta_m - \rho \frac{\partial J}{\partial \vartheta_m}$$



Gradient Boosting

Come questo si collega con il Gradient Descent?

Per un problema di regressione con funzione di perdita quadratica,

- residuo $\leftrightarrow$ gradiente negativo
- adattare h al residuo $\leftrightarrow$ adattare h al gradiente negativo
- aggiornare f tramite il residuo $\leftrightarrow$ aggiornare f tramite il gradiente negativo

Stiamo quindi usando il gradiente negativo

Gradient Boosting: Algoritmo

Un Gradient Boosting può essere quindi definito a partire dai seguenti elementi di input:

- Insieme di stima $(x_i, y_i) \dots (x_n, y_n)$
- funzione di perdita $L(y, f(x))$
- numero di iterazioni M

Gradient Boosting: Algoritmo

Algoritmo Gradient Tree Boosting

- inizializzare il modello con un valore costante

$$f_0(x) = \arg \min_{\gamma} \sum_{i=1}^n L(y_i, \gamma)$$

con funzione di perdita quadratica si ha $f_0 = \bar{y}$

- ad ogni iterazione $1 < m < M$ calcolare i gradienti negativi per $i = 1, 2, \dots, n$

$$-g(x_i) = - \left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)} \right] = y_i - f(x_i)$$

Gradient Boosting: Algoritmo

... continua

- stimare un albero di regressione $h_m(x)$ su $-g(x_i)$ assegnando regioni terminali $R_{jm}, j = 1, 2, \dots, J_m$
- per $j = 1, 2, \dots, J_m$ calcolare
$$\gamma_{jm} = \arg \min_{\gamma} \sum_{x_i \in R_{jm}} L(y_i, f_{m-1}(x_i) + \gamma)$$
- aggiornare il modello $f_m(x) = f_{m-1}(x) + \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})$
- Output: $\hat{f}(x) = f_M(x)$

Nota: Il vantaggio di usare i gradienti negativi è poter usare funzioni di perdita diverse da quella quadratica e derivare i corrispondenti algoritmi

Gradient Boosting

Perchè usare funzioni di perdita diverse?

Funzione di perdita quadratica è:

- semplice da gestire matematicamente ...
- ma poco robusta rispetto agli outlier

y_i	0.5	1.2	2	5*
$f(x_i)$	0.6	1.4	1.5	1.7
$(y - f)^2/2$	0.005	0.02	0.125	5.445

→ La presenza di un outlier può avere quindi conseguenze negative sulla performance generale del modello

Gradient Boosting

Funzione di perdita alternative

- funzione di perdita **assoluta**

$$L(y, f) = |y - f|$$

- funzione di perdita di **Huber** → più robusta rispetto agli outlier

$$L(y, f) = \begin{cases} 1/2(y - f)^2 & |y - f| \leq \delta \\ \delta(|y - f| - \delta/2) & |y - f| > \delta \end{cases}$$

y_i	0.5	1.2	2	5*
$f(x_i)$	0.6	1.4	1.5	1.7
quadratica	0.005	0.02	0.125	5.445
assoluta	0.1	0.2	0.5	3.3
Huber($\delta = 0.5$)	0.005	0.02	0.125	1.525

Gradient Boosting

Anche nel caso del GB, è possibile introdurre delle tecniche di regolazione, per ridurre il rischio di sovradattamento.

Shrinkage

La regola di aggiornamento viene modificata in questo modo

$$f_m(x) = f_{m-1}(x) + \nu \cdot \sum_{j=1}^J \gamma_{jm} I(x \in R_{jm})$$

Il parametro $0 < \nu < 1$ controlla il 'tasso di apprendimento' della procedura di boosting.

Valori più piccoli di $\nu \rightarrow$ maggiore shrinkage $\rightarrow M$ più grande

Esiste un trade-off tra ν ed M .

Gradient Boosting

Perchè usare il Gradient Boosting?

- utilizzo di dati di natura 'mista'
- robusto alla presenza di outlier nei dati di input
- interpretabilità
- capacità previsiva

Gradient Boosting

Confronto tra modelli (da HTF)

MART → Gradient Boosting

Some characteristics of different learning methods.

Key: ● = good, ● = fair, and ● = poor.

Characteristic	Neural Nets	SVM	CART	GAM	KNN, kernels	MART
Natural handling of data of "mixed" type	●	●	●	●	●	●
Handling of missing values	●	●	●	●	●	●
Robustness to outliers in input space	●	●	●	●	●	●
Insensitive to monotone transformations of inputs	●	●	●	●	●	●
Computational scalability (large N)	●	●	●	●	●	●
Ability to deal with irrelevant inputs	●	●	●	●	●	●
Ability to extract linear combinations of features	●	●	●	●	●	●
Interpretability	●	●	●	●	●	●
Predictive power	●	●	●	●	●	●

Gradient Boosting: esempio

- Data set relativo al prezzo delle case in California
- variabile risposta $y =$ **prezzo mediano** misurato in centinaia di migliaia di dollari
- **variabili demografiche**: reddito medio (MedInc), densità delle case (House), numero medio di persone in una casa (AveOcc), popolazione (Population)
- **caratteristiche della casa**: latitudine e longitudine (latitude, longitude), numero medio di stanze (AveRooms) e numero medio di stanze da letto (AveBedrms), anni della casa (HouseAge)
- 8 predittori

Gradient Boosting: esempio

Visualizzazione dei risultati

- grafico di importanza relativa: descrive la riduzione di errore quadratico attribuibile ad ogni variabile.

Gradient Boosting: esempio

Gradient Boosting con profondità alberi= 6, shrinkage= 0.1, funzione di perdita= Huber

