Apprendimento statistico e machine learning

Bootstrap e metodi di insieme/aggregazione

Leonardo Egidi

Novembre 2024

Università di Trieste

Bootstrap

I metodi di ricampionamento

- I *metodi di ricampionamento* sono metodi *computazionalmente intensi* che utilizzano la simulazione per fornire conclusioni inferenziali sui dati a disposizione.
- In alcuni casi questi metodi sostituiscono alle formule matematiche le simulazioni, benché spesso per provarne la validità servano risultati matematici sofisticati.
- Ci sono molti metodi di ricampionamento, ma il più famoso e attuale, che ha aperto il campo computazionale della statistica, è il cosiddetto metodo *bootstrap*, inventato da Bradley Efron nel 1977.
  - Il nome "bootstrap" deriva dalla frase **to pull oneself up by one's bootstraps**, letteralmente *"a tirarsi su per le cinghie"*, attribuita probabilmente al libro *"The Surprising Adventures of Baron Munchausen"* di Rudolph Erich Raspe, in cui vi è la frase: *"Il barone era caduto sul fondo di un lago profondo. Solo quando sembrava che tutto fosse perduto, pensò di tirarsi su con le sue stesse cinghie"*.
  - Non è lo stesso termine "bootstrap" utilizzato in informatica.

Bootstrap

- Il bootstrap è un metodo statistico potente e flessibile che può venir utilizzato per quantificare l'incertezza associata a un dato stimatore o a un metodo di apprendimento statistico.
- Per esempio, tramite di esso ci si può ricavare una stima dello standard error di un coefficiente, o un intervallo di confidenza per lo stesso coefficiente.
- Supponiamo $y_i, \ldots, y_n$, con $Y_i \sim F$, per qualche F ignota. Siamo interessati a una statistica (stima di un parametro) $\widehat{\psi} = s(\mathbf{y})$, dove $s(\cdot)$ è funzione delle n osservazioni. *Come facciamo a valutare lo standard error di* $\widehat{\psi} = s(\mathbf{y})$*?*
- In principio, potremmo generare un numero alto di campioni iid dalla vera distribuzione F e calcolare la varianza $\mathrm{Var}(\widehat{\psi})$. Ma questo è impossibile, non conosciamo F!
- Il bootstrap usa allora una stima $\widehat{F}$ di F in luogo di F per generare campioni e calcolare lo s.e. sui campioni simulati.

Bootstrap non parametrico

- In particolare il *bootstrap non parametrico* usa come $\widehat{F}$ la funzione di ripartizione empirica: ogni campione simulato è quindi una sequenza proveniente da un *campionamento con sostituzione (random sampling with replacement)* rispetto al campione originario.
- A quel punto la stima dello s.e. per una data statistica di interesse si ottiene direttamente dai campioni bootstrap simulati.
- Si può anche andare oltre allo standard error, e reperire per esempio un' approssimazione per l'intera distribuzione dello stimatore.

Esempio: salari annuali

- Ecco un esempio tratto da Boos and Stefanski, 2010, *Significance*, con i redditi annuali in migliaia di dollari.

Table 1. Random sample of 25 yearly incomes in thousands of dollars (ordered from lowest to highest)

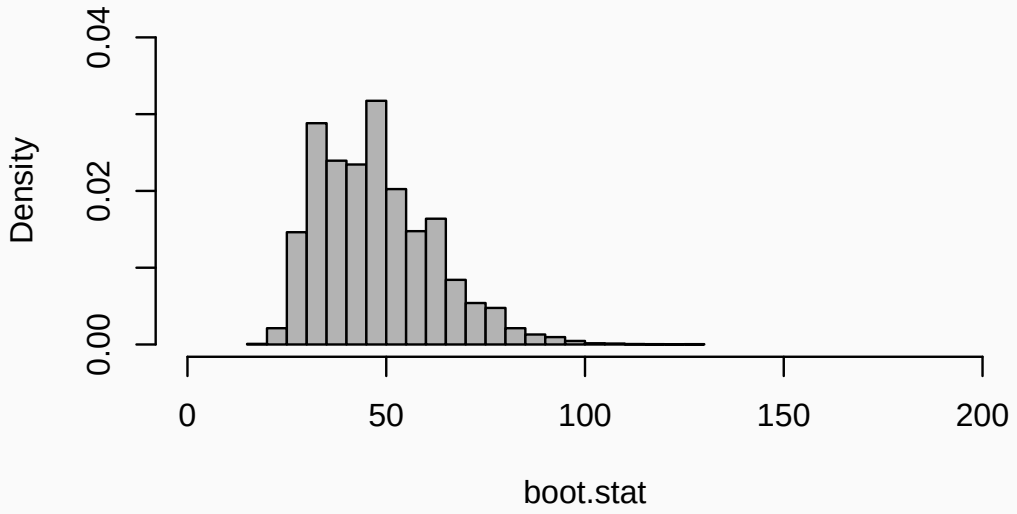
1	4	6	12	13	14	18	19	20	22	23	24	26
31	34	37	46	47	56	61	63	65	70	97	385	

- I campioni bootstrap possono essere utilizzati per ottenere la stima dello s.e. di una data statistica: denotando con $\widehat{\psi}^{*b}$, $b = 1, \ldots, B$ la statistica di interesse per ogni campione bootstrap, abbiamo:

$$\widehat{\mathrm{SE}}_{boot} = \left[\frac{1}{B-1} \sum_{b=1}^{B} \left(\widehat{\psi}^{*b} - \widehat{\psi}^{*\cdot} \right)^2 \right]^{1/2}, \quad \text{con} \quad \widehat{\psi}^{*\cdot} = \frac{1}{B} \sum_{b=1}^{B} \widehat{\psi}^{*b}.$$

Esempio in R: salari annuali

```
y <- c(1, 4, 6, 12, 13, 14, 18, 19, 20, 22, 23, 24, 26, 31, 34,
       37, 46, 47, 56, 61, 63, 65, 70, 97, 385)
n <- length(y)  # sample size
B <- 10^4       # bootstrap samples
boot.sample <- matrix(NA, nrow = B, ncol = n)
for(i in 1:B) boot.sample[i,] <- sample(y, n, replace = TRUE)
boot.stat <- rowMeans(boot.sample)
hist(boot.stat, main="", breaks=20, prob=TRUE, col=gray(0.7),
    xlim=c(0, 200), ylim=c(0, 0.04))
```



Qualche dettaglio teorico sul bootstrap

- Se il numero di campioni bootstrap B è sufficientemente grande, lo standard error basato sul bootstrap è *non distorto*.
- L'idea del bootstrap può essere bene apprezzata marcando il parallelo tra modello statistico per i dati 'originali'

$$F \xrightarrow{\text{i.i.d.}} \mathbf{y} \xrightarrow{s(\cdot)} \widehat{\psi}$$

e il meccanismo dei campioni simulati (o 'bootstrappati', in gergo)

$$\widehat{F} \xrightarrow{\text{i.i.d.}} \mathbf{y}^* \xrightarrow{s(\cdot)} \widehat{\psi}^* \,.$$

- Il collegamento tra le due rappresentazioni è dato dal fatto che $\widehat{F}$ **converge alla vera** F **quando** $n \to \infty$.

Commenti sul bootstrap non parametrico

- Completamente automatico! La matematica sottostante non è semplice, ma è stata derivata rigorosamente.
- Si tratta di un metodo basato sui campioni grandi, siccome la sua accuratezza cresce con n.
- Può essere esteso a qualsiasi statistica di interesse, non solo per stimare gli standard errors.
- Può essere esteso a modelli complessi.
- Ha anche alcune limitazioni, ad esempio non essere particolarmente appropriato per gli estremi della distribuzione dei dati, come il minimo e il massimo.

Bootstrap parametrico

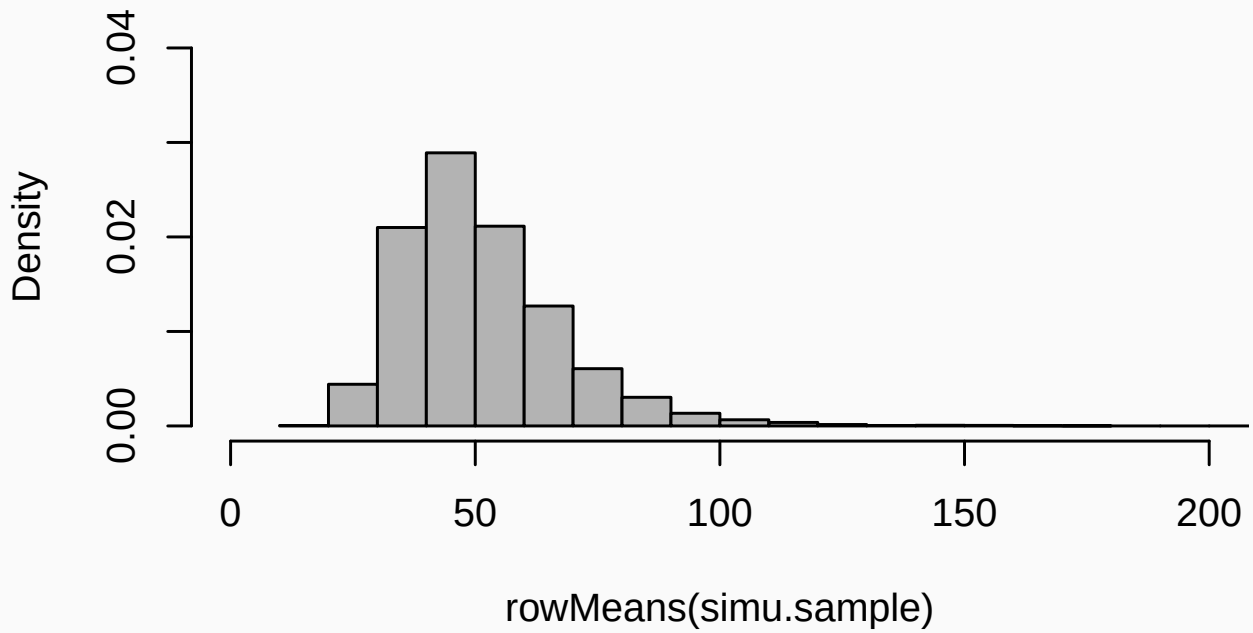
- Non c'è per forza bisogno che $\widehat{F}$ sia una stima non parametrica della F (la funzione di ripartizione empririca).
- Un'altra alternativa è considerare un modello parametrico per i dati $f_{\boldsymbol{\theta}}(\mathbf{y})$ e simulare i campioni bootstrap direttamente da $f_{\hat{\boldsymbol{\theta}}}$, dove $\hat{\boldsymbol{\theta}}$ è una stima puntuale per $\boldsymbol{\theta}$ (nella pratica un vettore di parametri).
- Il meccanismo diventa

$$f_{\hat{\boldsymbol{\theta}}} \longrightarrow \mathbf{y}^* \xrightarrow{s(\cdot)} \widehat{\psi}^*$$

- Nell'ambito dei test di ipotesi può essere usato per calcolare p-values approssimati tramite simulazione.
- La difficoltà può essere quella di specificare un modello sensato.

R lab: bootstrap parametrico per $\widehat{\psi} = \overline{Y}$

```r
n <- length(y); mu <- mean(log(y)); sigma <- sd(log(y))
simu.sample <- matrix(NA, nrow = B, ncol = n)
for(i in 1:B) simu.sample[i,] <- mean(exp(rnorm(n, mu, sigma)))
hist(rowMeans(simu.sample) , main="", breaks=25, prob=TRUE,
     col=gray(0.7), xlim=c(0, 200), ylim=c(0, 0.04))
```



Derivare intervalli di confidenza

- Tramite il bootstrap ci si può ricavare anche *intervalli di confidenza*, esistono vari metodi.
- Possiamo calcolarci per esempio l'*intervallo di Wald* al 95% per un parametro di interesse ψ, definito come:

$$\widehat{\psi} \pm 1.96 \times \mathrm{SE}(\widehat{\psi}),$$

  per il quale è sufficiente calcolarsi $\mathrm{SE}(\widehat{\psi})$ tramite i metodi visti prima. Una possibile inefficienza di questo intervallo è di essere simmetrico rispetto alla stima puntuale.
- Alternativamente vi è il cosiddetto *metodo del percentile*, che come altri funziona sia per il bootstrap parametrico che per quello non parametrico. Si usano semplicemente i percentili della distribuzione bootstrap $\widehat{\psi}^{*1}, \ldots, \widehat{\psi}^{*B}$. Molto facile in `R` una volta che ho i campioni.
- Altri metodi: metodo base e metodo studentizzato. In generale, piccole differenze.

Metodi di ensemble/aggregazione

Bagging: motivazioni

- Come abbiamo visto i CART possono soffrire di alcune inefficienze strutturali. Ad esempio, un piccolo cambio nei predittori/dati può causare grande instabilità nelle stime e previsioni. Si tratta di metodi ad *alta varianza e basso bias*.
- L'*aggregazione bootstrap*, detta anche *bagging*, è una procedura generale che mira ad abbassare la varianza di un metodo statistico. Per quanto non esclusiva dei soli CART, la introduciamo qui perché trova con i CART una delle massime espressioni di utilizzo.
- Ricordiamo che data una sequenza di n osservazioni indipendenti $Y_1, Y_2, \ldots, Y_n$, ognuno con varianza σ^2, la varianza della media $\bar{Y}$ è data da σ^2/n.
- In altre parole, *combinare più osservazioni riduce la variabilità*. Di fatto però, nelle applicazioni reali, non disponiamo di più di un training set! (Nel caso ad esempio di un clinical trial, registriamo i dati una volta sola).
- Perché usare il bootstrap qui? Perché ci aiuta a migliorare l'accuratezza di tanti metodi statistici tra i quali appunto i CART.

Bagging: aggregare più campioni

- Un modo naturale per ridurre la varianza - e quindi accrescere la bontà predittiva dell'algoritmo in questione - è quello di considerare *sequenze di campioni ripetuti* dal singolo training set.
- Con questo metodo generiamo B differenti campioni 'bootstrappati' dal training set. *Alleniamo il nostro metodo ad albero* sul b-esimo campione bootstrap per ottenere la previsione nel punto x, $\widehat{f}^{*b}(x)$. Calcoliamo poi la media di queste previsioni come

$$\widehat{f}_{\text{bag}}(x) = \frac{1}{B}\sum_{b=1}^{B}\widehat{f}^{*b}(x).$$

Questo è chiamato *bagging*.

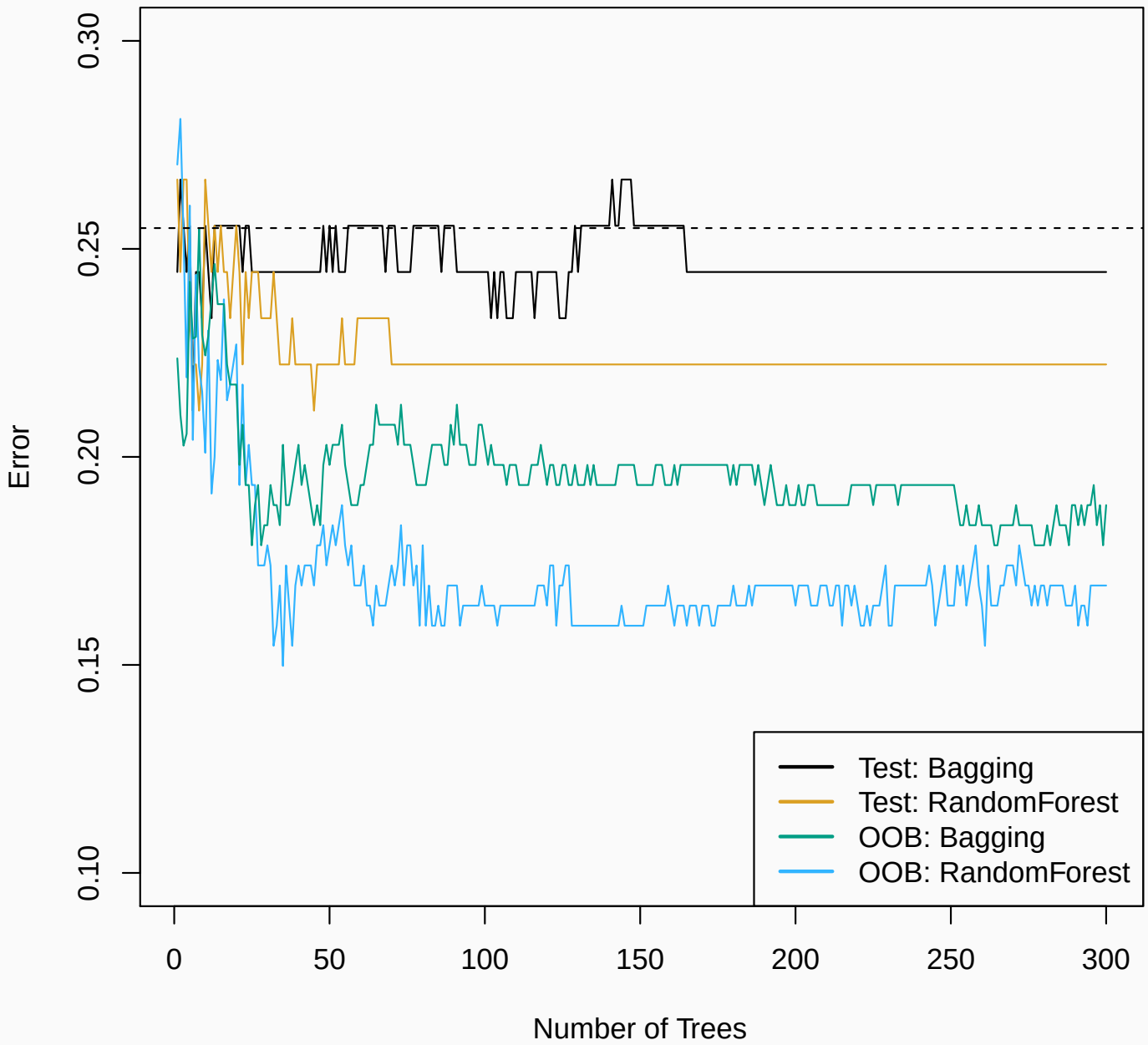
Due parole sulla costruzione dei B datasets bootstrappati

- *Attenzione*: nel concreto, il bagging è costruito con uno schema di *sotto-campionamento con sostituzione (replacement) dal training set*: non tutte le osservazioni statistiche vengono usate per costruire datasets replicati/bootstrappati, ma solo una porzione di essi.
- Per ogni albero, avremo quindi delle osservazioni *in-the-bag*, e altre *out-of-bag*, che varieranno dunque di albero in albero.
- Non solamente: tipicamente ogni albero 'bagged' coinvolgerà nella pratica diversi predittori, diversi splits e un diverso numero di nodi terminali rispetto a quelli che verrebbero usati per i dati originali!
- Il bagging però non *discrimina* sulla scelta dei predittori: ad ogni nuovo dataset b, potenzialmente infatti ciascuno dei p predittori può venir scelto per lo split - i *random forests*, come vedremo, pongono invece un vincolo su questa scelta.

Bagging: aggregare più campioni

- Disclaimer: stiamo parlando di bagging in ambito di alberi di regressione.
- Se volessimo applicarlo agli alberi di classificazione - con variabile risposta dicotomica per esempio - dovremmo ricorrere al cosiddetto metodo del *voto di maggioranza (majority vote)*: nella formula della slide precedente, $\widehat{f}_{\text{bag}}$ è la più frequente tra le due classi - se siamo in caso dicotomico - tra le B previsioni.
- Ogni singolo albero ha *alta varianza, e basso bias*. Aggregare più alberi significa avere *bassa varianza* e *bias più alto*.
- Si è dimostrato che le performance del bagging - per esempio con 100, 1000,. . . alberi combinati - possono migliorare *sensibilmente* quelle dei singoli alberi. Tuttavia, non è sempre il caso. . .

Bagging: esempio sui dati del cuore



Dettagli figura precedente

- `Heart` data: dataset `heart_disease` del pacchetto `R cheese`, contenente la risposta binaria `HD` per 303 pazienti che hanno manifestato dolori al petto. `"Yes"` indica la presenza di malattia cardiaca, mentre `"No"` indica il contrario. Ci sono 13 predittori, inclusi età, sesso e misurazione del colesterolo.
- L'errore di test è mostrato in funzione al numero di alberi bootstrappati B. Vediamo che l'errore sul test del bagging (curva nera) risulta leggermente inferiore a quello di un singolo albero (linea tratteggiata), mentre lo stesso errore per un random forest (curva arancione) con $m = \sqrt{p}$ predittori è ancora più basso e decresce al crescere degli alberi, per poi stabilizzarsi completamente.
- Le curve verdi e blu mostrano invece il cosiddetto errore stimato *out-of-bag* (lo introduciamo tra poco), che risulta sia per bagging che per RF considerevolmente più basso.

Out-of-bag error

- C'è un metodo molto semplice e immediato per stimare l'errore sul test-set di un metodo bagging, senza alcun bisogno di ricorrere a CV o metodi simili.
- Ricordiamo che solitamente gli alberi del bagging sono stimati ripetutamente su *sotto-insiemi bootstrappati delle osservazioni*. Si può dimostrare che in media ogni albero 'bagged' usa in realtà all'incirca due terzi delle osservazioni ('in-the-bag'). Il restante terzo di osservazioni non utilizzato per stimare un dato albero bagged viene denominato per l'appunto il set di osservazioni *out-of-bag* (OOB, letteralmente 'fuori dalla scatola').
- Possiamo quindi prevedere la risposta per la i-esima osservazione usando ciascuno degli alberi in cui questa osservazione è OOB. Questo fornirà grosso modo $B/3$ previsioni per la i-esima osservazione, di cui facciamo quindi la media per calcolare l'errore OOB.
- Essenzialmente, questo è l'errore di leave-one-out (LOO) CV per il bagging, supposto B sufficientemente grande. Ma è *automatico*!

Random forests

- I *random forests* (foreste casuali) producono un miglioramento rispetto al bagging introducendo una piccola modifica che *de-correla* i diversi alberi aggregati. Questo riduce ulteriormente la varianza del metodo quando combiniamo diversi alberi.
- Esattamente come per il bagging costruiamo B alberi su B dataset 'bootstrappati' a partire dal training set originale.
- Ma quando costruiamo il singolo albero, ogni volta in cui viene considerato uno split nell'albero viene considerata una *scelta casuale di m predittori* tra i p originali. Nello split in questione è consentito usare solo uno di questi m predittori scelti casualmente.
- Una selezione 'fresca' di m predittori viene quindi considerata ad ogni singolo split, e tipicamente la scelta ricade su $m \approx \sqrt{p}$ (4 su 13 nel caso dei data `Heart`).

Random forests: *ratio* dei RF

- Siccome ogni albero generato nel bagging è identicamente distribuito (id) agli altri, il bias degli alberi bagged è lo stesso di quello dei singoli alberi individuali. L'unico modo per migliorarne le performance è quindi quello di *ridurre la varianza*.
- Se ho B variabili iid, abbiamo visto che la varianza della loro media è data da σ^2/B. Ma se le variabili sono semplicemente id, allora si può provare che la varianza della media è data da:

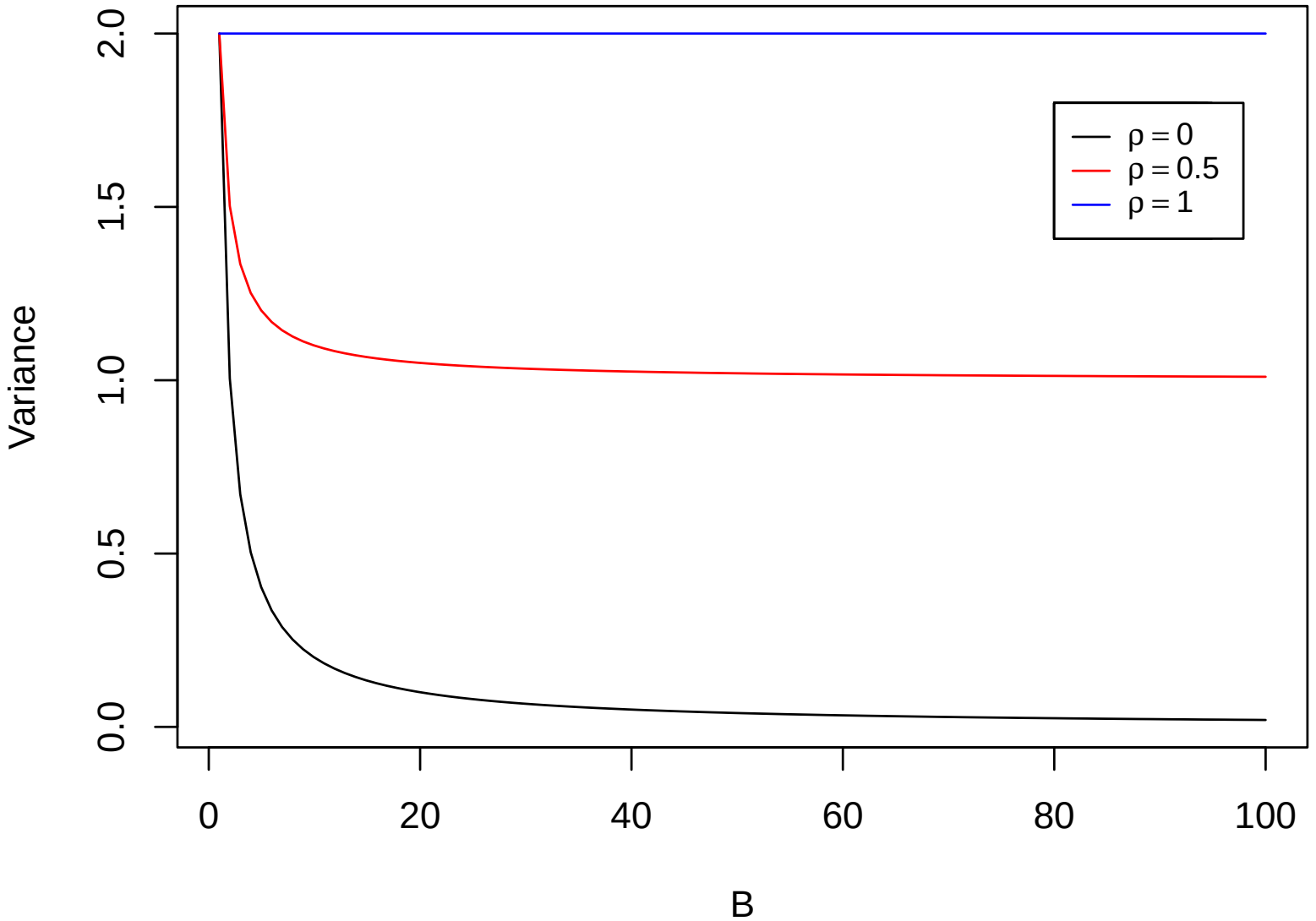
$$\rho\sigma^2 + \frac{1-\rho}{B}\sigma^2,$$

  dove ρ è l'indice di correlazione a coppie.
  - Se $B \to \infty$, il secondo termine sparisce, e rimane solo $\rho\sigma^2$.
  - Se $\rho = 1$, il secondo termine sparisce, e il primo diventa σ^2 (di fatto considero sempre lo stesso albero!)
  - La 'grandezza' della correlazione a coppie degli alberi limita i benefici del calcolare la media!
- I RF migliorano la riduzione della varianza del bagging riducendo la correlazione tra gli alberi nella formula qui sopra.

Random forests: *ratio* dei RF

- Perché $m \approx \sqrt{p}$ ad ogni split dovrebbe de-correlare gli alberi?
- Supponiamo vi sia un predittore molto rilevante nel dataset originale, assieme ad altri predittori moderatamente importanti. Nel bagging molti se non tutti gli alberi useranno questo predittore forte nel primo split in cima all'albero. Conseguentemente, *tutti gli alberi bagged saranno molto simili tra loro* e le relative previsioni molto correlate.
- Sfortunatamente, calcolare la media di molte quantità correlate non porta a una significativa riduzione della varianza come 'mediando' molte quantita non correlate (vedi formula slide precedente e grafico slide successiva). Quindi, il bagging non porta a una sostanziale riduzione della varianza rispetto a un singolo albero!
- I RF superano questo problema condiderando ad ogni split solo un sotto-insieme di predittori m, il *parametro di tuning*. In media $(p - m)/p$ degli splits non considereranno il predittore *forte*, e quindi gli altri predittori avranno più chances. Gli alberi saranno de-correlati e la media delle previsioni più affidabile e robusta.

Random forests: varianza della media in funzione di ρ e B



Boosting

- Come il bagging, anche il *boosting* è una procedura generale che può venir applicata a molti metodi di apprendimento statistico per migliorarne la bontà predittiva. Ma qui lo vediamo applicato con riferimento esclusivo agli alberi di regressione e classificazione.
- A differenza del bagging, che produce diverse 'copie' del dataset originario e applica gli alberi a ciascuno di questi datasets *indipendentemente*, nel boosting gli alberi crescono *sequenzialmente*: ogni albero è fatto crescere usando l'informazione sugli alberi cresciuti precedentemente.
- Il boosting non si basa dunque sul bootstrap, piuttosto ogni albero è stimato su una versione modificata del dataset originario.

Algoritmo boosting per alberi di regressione

1. Inizializzare $\widehat{f}(x) = 0$ e $r_i = y_i$ per tutti gli i del training set.
2. Per $b = 1, \ldots, B$ ripetere:

- Fittare un albero $\widehat{f}^b$ con d splits ($d + 1$ nodi terminali) sui dati di training (X, r).
- Aggiornare $\widehat{f}$ aggiungendo in una versione regolarizzata (shrunken) del nuovo albero:

$$\widehat{f}(x) \leftarrow \widehat{f}(x) + \lambda \widehat{f}^b(x).$$

- Aggiornare i residui,

$$r_i \leftarrow r_i - \lambda \widehat{f}^b(x_i).$$

3. Ottenere il modello boosted,

$$\widehat{f}(x) = \sum_{b=1}^{B} \lambda \widehat{f}^b(x).$$

Che idea dietro la procedura boosting?

- Invece di stimare un singolo e grande albero sul training set, che implica *un difficile apprendimento dai dati* e potenziale overfitting, il boosting invece *impara lentamente*.
- Dato un modello corrente, stimiamo un albero sui residui di questo modello, anziché sulla variabile risposta Y. Aggiungiamo poi questo nuovo albero di decisione nella funzione stimata per aggiornare i residui.
- Ognuno di questi alberi può essere piuttosto piccolo, con solamente qualche nodo terminale, con numero determinato dal parametro di tuning d.
- Stimando piccoli alberi sui residui miglioriamo lentamente $\widehat{f}$ in aree dove questa non performa bene. Il parametro di shrinkage λ rallenta il processo ulteriormente, consentendo ad alberi sempre più di forma diversa di 'aggredire' i residui.

Parametri di tuning per il boosting

- Il *numero di alberi* B. A differenza di quanto avviene con bagging e RF, il boosting può overfittare i dati se B è troppo grande - per quanto l'overfitting avvenga, nel caso, lentamente. Per stimare B si può usare CV.
- Il *parametro di shrinkage* λ, un piccolo numero positivo. Questo valore controlla la velocità con cui il boosting apprende. Tipici valori sono 0.01 o 0.001, e la corretta scelta può dipendere dal problema specifico. Valori di λ molto piccoli possono richiedere un B molto grande per ottenere buone performance.
- Il *numero di splits* d in ogni albero, che controlla la complessità della collezione boosting. Spesso $d = 1$ funziona bene, nel qual caso ogni albero è un *ceppo*, che consiste di un unico split in un modello additivo. Più generalmente d è la *profondità di interazione*, e controlla l'ordine di interazione del modello boosted, siccome d splits possono coinvolgere al più d predittori.

Gradient boosting per alberi di regressione

- L'algoritmo presentato alla slide 24 è sicuramente flessibile, tuttavia esistono procedure di boosting ancora più *robuste* e veloci, basate sul *gradiente di una data funzione di perdita*, $L(y_i, f(x_i))$, $i = 1, \ldots, n$.
- Quando abbiamo introdotto gli alberi, abbiamo visto che ad ogni split cerchiamo una coppia di parametri $\Theta = \{R_j, c_j\}_1^J$ in base a un *criterio di minimizzazione*. Possiamo 'tradurre' genericamente quel criterio come

$$\widehat{\Theta} = \underset{\Theta}{\operatorname{argmin}} \sum_{j=1}^{J} \sum_{x_i \in R_j} L(y_i, c_j),$$

  con $L(\cdot)$ opportuna funzione di perdita, *loss function*, tipicamente la RSS o una quantità direttamente collegata ad essa.
- Ricordiamoci che nel boosting la minimizzazione qui sopra avviene *sequenzialmente*, per ogni albero aggregato, su dei residui 'aggiustati' con una componente di shrinkage. Nella prossima slide introduciamo un algoritmo per gli alberi di *gradient boosting*, usato per esempio dal pacchetto `gbm`.

Algoritmo gradient boosting per alberi di regressione

1. Inizializzare $\widehat{f}^0(x) = \underset{c}{\text{argmin}} \sum_{i=1}^{n} L(y_i, c)$.
2. Per $b = 1, 2, \ldots, B$:

- Per $i = 1, 2, \ldots, n$ calcolare:

$$r_i^b = - \left[\frac{\partial L(y_i, \widehat{f}(x_i))}{\partial \widehat{f}(x_i)} \right]_{\widehat{f} = \widehat{f}^{b-1}}.$$

- Stimare un albero di regressione sui residui r_i^b che dà i nodi terminali R_j^b.
- Per $j = 1, 2, \ldots, J^b$ calcolare:

$$\hat{c}_j^b = \underset{c}{\text{argmin}} \sum_{x_i \in R_j^b} L(y_i, \widehat{f}^{b-1}(x_i) + c).$$

- Aggiornare

$$\widehat{f}^b(x) = \widehat{f}^{b-1}(x) + \sum_{j=1}^{J^b} \hat{c}_j^b I(x \in R_j^b).$$

3. Ottenere il modello boosted,

$$\widehat{f}(x) = \widehat{f}^B(x).$$

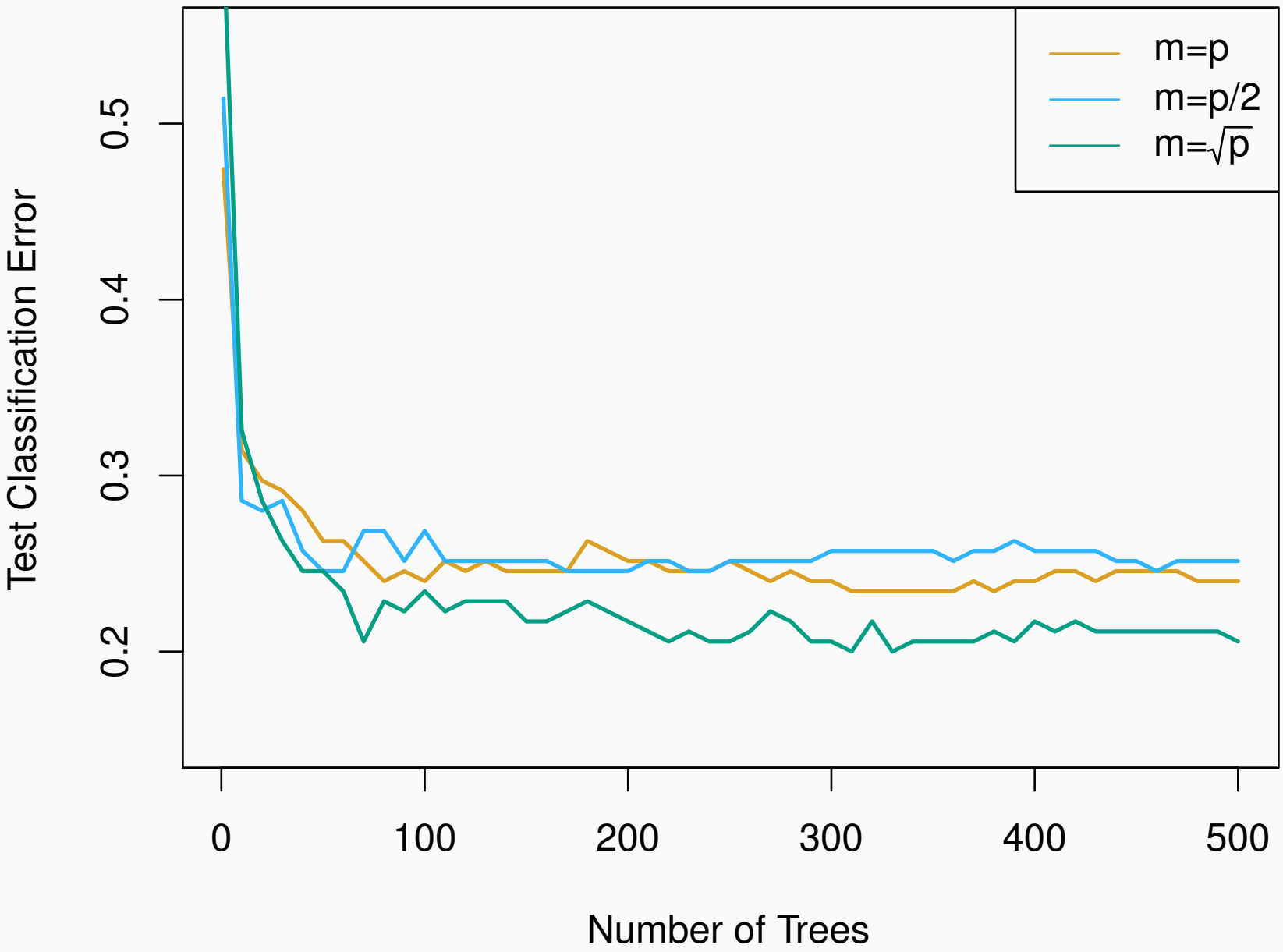
RF e boosting per classificazione

- Come detto, questi metodi ben si adattano a scenari di classificazione, dove la variabile risposta è categoriale e strutturata in classi - ad esempio, dicotomica. Nel caso del boosting lo scenario della classificazione è leggermente più complesso per quanto riguarda le procedure di stima.
- I pacchetti `R` `randomForest` e `gbm` consentono di stimare RF e (gradient) boosting sia in ambito di regressione che di classificazione.
- Il pacchetto `R` `caret` consente di stimare molti metodi di cosiddetto *machine learning*.

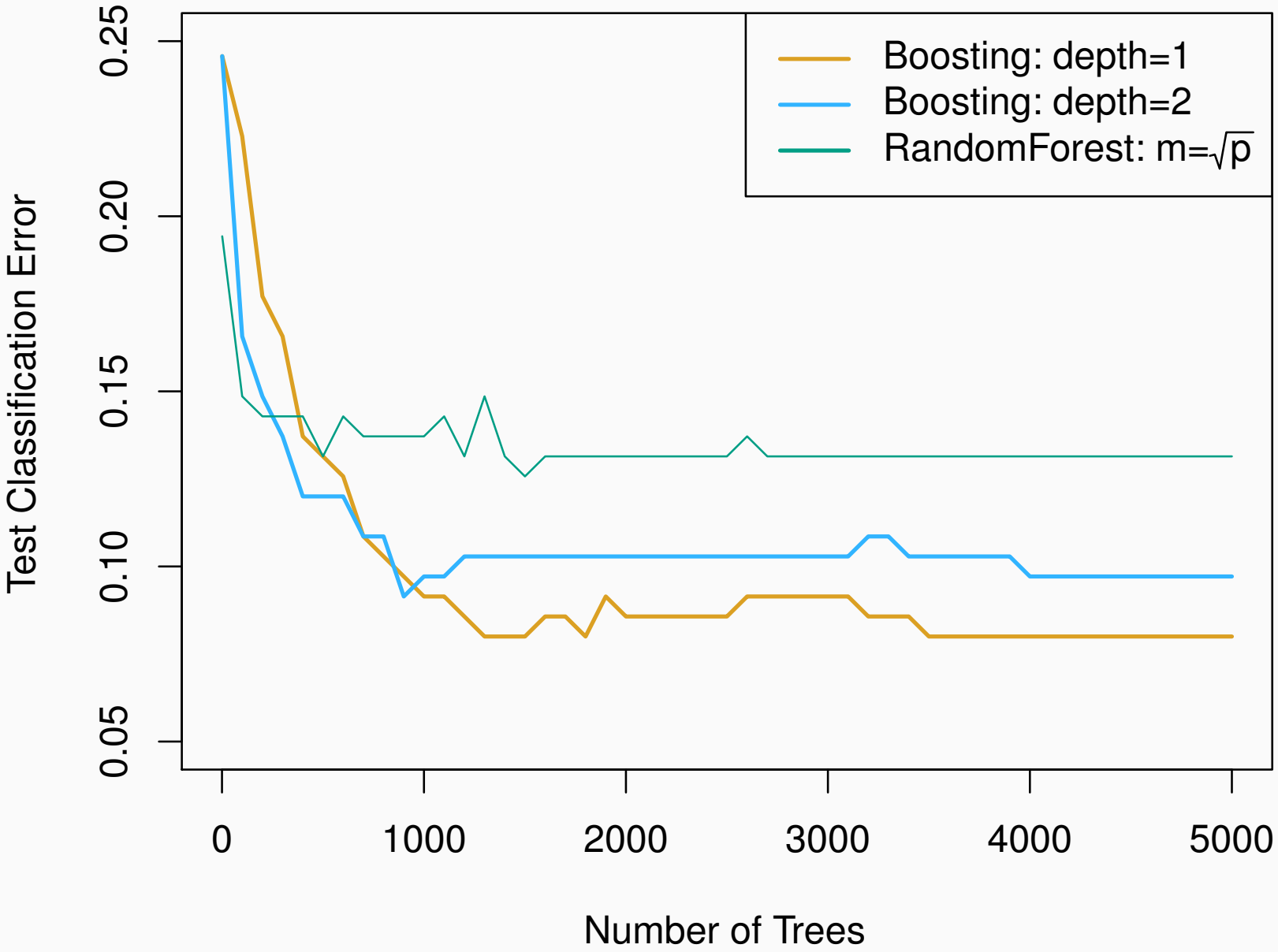
Esempio: dati su espressione genica

- Applichiamo bagging, RF e boosting ad un dataset di grandi dimensioni che consiste nelle misurazioni dell'espressione di 4,718 geni registrati su campioni di tessuti cellulari di 349 pazienti ($p >> n$).
- Ci sono circa 20,000 geni negli esseri umani, e geni individuali hanno diversi livelli di attività, o espressione, in particolari cellule, tessuti e condizioni biologiche.
- Ognuno dei campioni di pazienti ha un'etichetta qualitativa - la variabile risposta - con 15 diversi livelli: o normale, o uno tra 14 diversi livelli di tumore.
- Usiamo bagging, RF e boosting per prevedere il tipo di cancro basato su 500 geni che hanno la più grande varianza nel set di training.
- Dividiamo randomicamente l'insieme dei dati in training e test, e applichiamo RF e boosting con diversi parametri di tuning calcolando l'errore di classificazione sul test set.

Esempio: dati su espressione genica



Esempio: dati su espressione genica



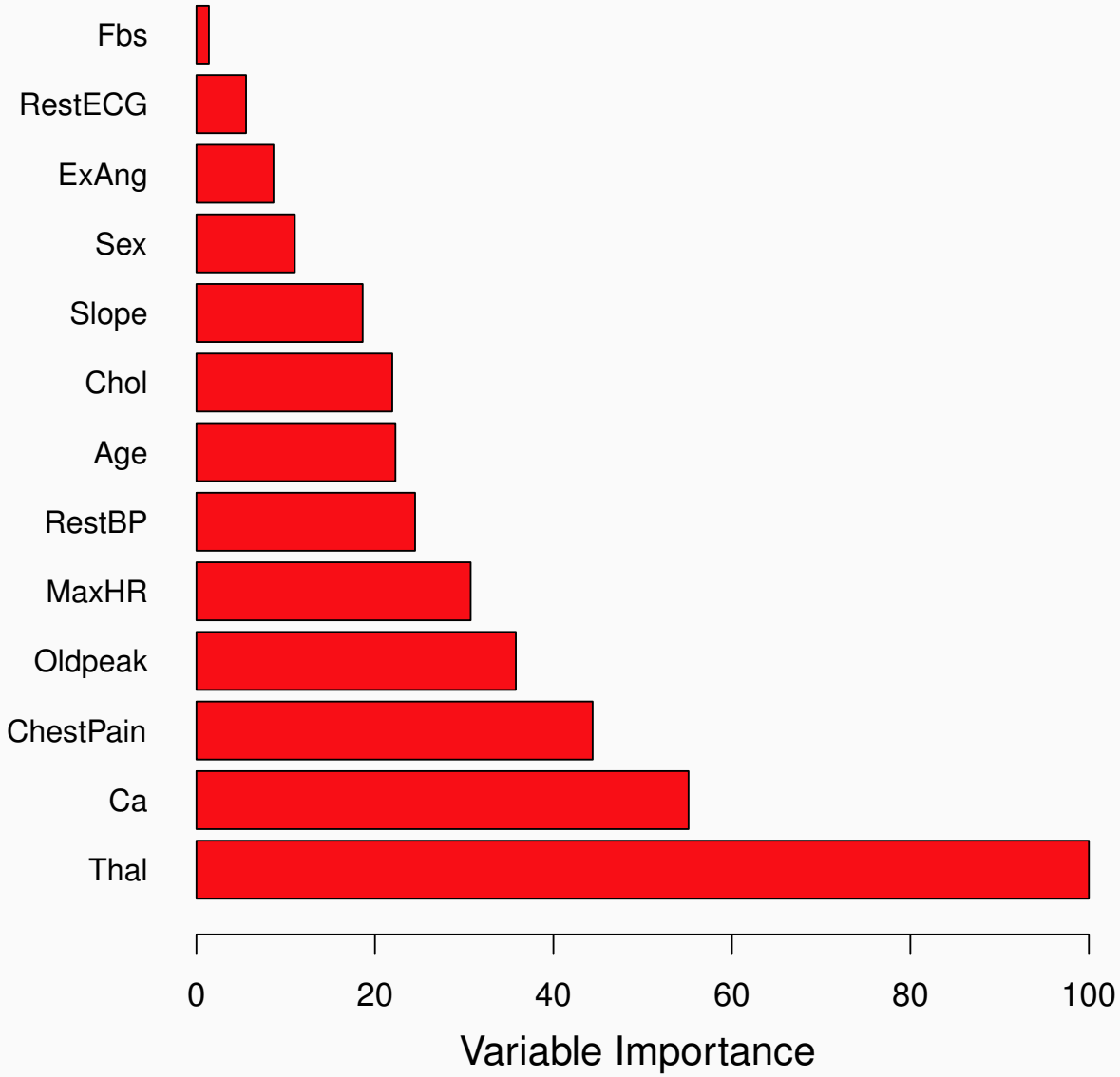
Commenti sulle figure precedenti

- Nella prima figura vediamo come al crescere dei campioni B i RF con $m = \sqrt{p}$ migliorano l'efficacia predittiva del bagging ($m = p$). Un singolo albero di classificazione qui ha un errore sul test set del 45.7%.
- Nella seconda figura vediamo come i Depth-1 boosting trees (con $d = 1$) vanno meglio dei Depth-2 ($d = 2$), e tutti e due i metodi di boosting dominano i RF al crescere di B.

Misure di importanza dei predittori

- In questi metodi di insieme un altro grande vanataggio, rispetto al singolo albero, è di poter riassumere l'*importanza dei predittori*.
- Nei bagged/RF di regressione possiamo registrare quanto la RSS diminuisce in seguito a splits di un dato predittore, facendo poi la media sui B alberi. Un valore alto segnala la presenza di un predittore importante.
- Nei bagged/RF di classificazione possiamo usare l'indice di Gini con la stessa modalità (se ne parlerà propriamente durante le lezioni sulla classificazione).

Dati sui cuori: importanza delle variabili per RF



Conclusioni

- Gli alberi di classificazione sono metodi semplici ed interpretabili sia per regressione che per classificazione.
- Tuttavia spesso non sono competitivi con altri metodi in quanto a efficacia predittiva.
- Bagging, random forests e boosting rappresentano metodi validi per migliorare le performance predittive di un singolo albero. Lavorano combinando e applicando diversi alberi sui dati e poi calcolando le previsioni come medie di queste collezioni/ensembles.
- Gli ultimi due metodi - RF e boosting - sono oramai tra i più utilizzati e popolari per problemi di apprendimento supervisionato. Tuttavia i loro risultati sono spesso difficili da interpretare, tali da renderli spesso delle *black boxes*.