Università degli Studi di Milano Bicocca

Department of Informatics, systems and communication

High dimensional data analysis

Matteo Suardi

A.Y.: 2025-2026

Abstract

Questo documento presenta i principali metodi di regressione utilizzabili nel contesto dell'analisi di dati ad alta dimensionalità. Il corso prende avvio dalla regressione lineare classica, introducendo il trade-off bias–varianza e il fenomeno dell'overfitting, attraverso il confronto tra errore di training ed errore di test. Vengono quindi discussi criteri e strumenti per la selezione del modello e la valutazione delle prestazioni predittive, tra cui i criteri informativi e le tecniche di validazione incrociata. Un'attenzione particolare è dedicata ai metodi di regressione penalizzata, come ridge regression e lasso, fondamentali quando il numero di variabili è elevato rispetto alla numerosità campionaria. Il corso prosegue con lo studio di tecniche di regressione non parametrica, includendo k-nearest neighbors (k-NN), kernel smoothing, regression splines, smoothing splines e local regression, con l'obiettivo di modellare relazioni non lineari in modo flessibile. Infine, vengono affrontati temi di inferenza in alta dimensionalità, con metodi moderni per la selezione delle variabili e il controllo dell'errore statistico. L'obiettivo complessivo è fornire una comprensione unificata dei compromessi tra accuratezza predittiva, complessità del modello e interpretabilità, supportata da esempi teorici e implementazioni pratiche in R.

Contents

Chapter 1

Linear regression, bias/variance trade-off

1.1 Bias e Varianza: un problema prototipo

Con l'avvento dei computer, la statistica ha subito un'evoluzione radicale. La **statistica computazionale** è oggi un campo in forte crescita, caratterizzato da nuovi metodi e algoritmi, e da un numero sempre maggiore di applicazioni pratiche. Una delle sfide principali riguarda la **crescente dimensione e complessità dei dataset**: non solo in termini di archiviazione e gestione, ma anche di analisi. Per gestire e interpretare grandi quantità di dati, è stato necessario sviluppare nuove tecniche e tecnologie statistiche.

Un problema prototipo

Per introdurre il concetto di *bias-variance trade-off*, consideriamo un problema semplice, che fungerà da modello per situazioni più complesse e realistiche.

Immaginiamo di avere due insiemi di dati, ciascuno composto da $n = 30$ osservazioni:

$$(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$$

che rappresentano il **training set**, e

$$(x_1, y_1^*), (x_2, y_2^*), \ldots, (x_n, y_n^*)$$

che rappresentano il **test set**.

1.1.1 L'impostazione *Fixed-X*

Per semplicità, adottiamo il cosiddetto **Fixed-X setting**, che assume:

1. I valori predittori $x_1, \ldots, x_n$ del training set sono fissati (non casuali);
2. I valori $x_1, \ldots, x_n$ del test set coincidono esattamente con quelli del training set.

Questa assunzione semplifica l'analisi, concentrandosi esclusivamente sulla variabilità dovuta alla componente aleatoria del modello.

1.1.2 Il modello generativo

I dati di training sono generati dal modello

$$Y = f(X) + \varepsilon$$

dove:

- $f(x)$ è una funzione **regolare e liscia** ma sconosciuta;
- ε è un termine di errore con distribuzione

  $$\varepsilon \sim \mathcal{N}(0, \sigma^2)$$

  con $\sigma = 0.01$.

In forma discreta:

$$Y_i = f(x_i) + \varepsilon_i, \quad i = 1, \ldots, n.$$

Analogamente, per il test set:

$$Y_i^* = f(x_i) + \varepsilon_i^*, \quad i = 1, \ldots, n,$$

con ε_i e ε_i^* indipendenti e identicamente distribuiti secondo $\mathcal{N}(0, \sigma^2)$.

Obiettivo: stimare f

Il nostro scopo è stimare f a partire dai dati di training, ottenendo una funzione stimata $\hat{f}$ che ci consenta di predire le osservazioni future:

$$\hat{y}_i^* = \hat{f}(x_i), \quad i = 1, \ldots, n.$$

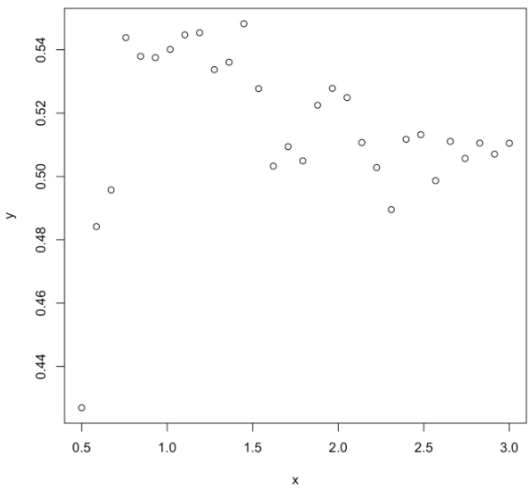
In altre parole, vogliamo che $\hat{f}$ riproduca il comportamento della vera funzione f nel modo più accurato possibile, minimizzando l'errore di previsione sul test set.

Esempio pratico in R

Il seguente esempio mostra come importare i dati e costruire il dataset di training in R:

```
# Import dei dati dal web
library(readr)
df <- read_table("http://azzalini.stat.unipd.it/Book-DM/yesterday.dat",
                 col_types = cols( x = col_double(),
                                   y.yesterday = col_double(),
                                   y.tomorrow = col_double())
                 )[-31,]

# Creazione del training set
train <- data.frame(x = df$x, y = df$y.yesterday)

# Visualizzazione dei dati
plot(y ~ x, train)
```

Il grafico risultante mostra la relazione tra i valori di x e le corrispondenti osservazioni y del training set. Questa rappresentazione visiva è essenziale per valutare la forma di $f(x)$ e per scegliere un modello di stima appropriato (ad esempio lineare, polinomiale, o non parametrico).



Osservazioni

Questo esempio, pur semplice, illustra bene l'idea chiave dell'analisi statistica: data una funzione sconosciuta f che lega la variabile predittiva X alla risposta Y, vogliamo costruire una stima $\hat{f}$ che generalizzi bene anche su nuovi dati. Nel prosieguo, analizzeremo come la complessità del modello influisce sull'**errore di previsione**, introducendo il concetto di **bias-variance trade-off**.

1.2 Mean squared error (MSE)

Una volta stimata la funzione $\hat{f}$ a partire dai dati di training, è naturale chiedersi quanto essa descriva bene i dati osservati. Una misura fondamentale di accuratezza del modello è l'**errore quadratico medio** (*Mean Squared Error*, MSE).
Per i dati di ieri (training data), definiamo:

$$MSE_{Tr} = \frac{1}{n} \sum_{i=1}^{n} \Big(y_i - \hat{f}(x_i) \Big)^2$$

dove:

- y_i è il valore osservato,
- $\hat{f}(x_i)$ è il valore predetto dal modello,
- n è il numero di osservazioni.

Interpretazione

Il valore di MSE_{Tr} rappresenta quanto le predizioni del modello si discostano, in media, dai valori reali nel dataset di addestramento.

- Un **MSE basso** indica che le predizioni $\hat{f}(x_i)$ sono molto vicine ai valori osservati y_i.

- Un **MSE alto** suggerisce che le differenze tra predetto e osservato sono grandi, quindi il modello non si adatta bene ai dati.

Tuttavia, poiché il MSE_{Tr} è calcolato sugli stessi dati utilizzati per costruire il modello, esso misura solo la capacità di *adattarsi ai dati noti* e non di generalizzare su dati nuovi. Per questo motivo, si parla di **training MSE** e si riconosce che non è una buona misura delle prestazioni globali del modello.

1.2.1 Errore di test

Per valutare la capacità di generalizzazione, si utilizza invece il **test MSE**, calcolato sui dati di test (cioè nuovi dati non utilizzati in fase di stima):

$$MSE_{Te} = \frac{1}{n}\sum_{i=1}^{n}\left(y_i^* - \hat{f}(x_i)\right)^2$$

dove y_i^* rappresenta le osservazioni del test set.
In pratica, il MSE_{Te} ci indica quanto bene il modello riesce a prevedere valori futuri (o non osservati), quindi è un indicatore molto più affidabile della qualità del modello rispetto al MSE_{Tr}.

Esempio intuitivo

Supponiamo di avere un modello che, sul training set, ottiene:

$$MSE_{Tr} = 0.001$$

mentre sul test set otteniamo:

$$MSE_{Te} = 0.15$$

In questo caso, il modello ha imparato molto bene i dati di addestramento ma non generalizza bene a dati nuovi: è un chiaro sintomo di **overfitting**. Un buon modello dovrebbe mantenere entrambi i valori di MSE bassi e, soprattutto, **simili tra loro**.

Osservazione finale

Il confronto tra MSE_{Tr} e MSE_{Te} rappresenta il punto di partenza per comprendere il **trade-off tra bias e varianza**, cioè l'equilibrio tra un modello troppo rigido (sottostima, alto bias) e uno troppo flessibile (sovrastima, alta varianza). Nella prossima sezione analizzeremo come questo equilibrio influenzi direttamente l'errore di previsione.

1.3 Regressione polinomiale

Per stimare la funzione $f(x)$ possiamo utilizzare diversi modelli. Una scelta naturale e flessibile è la **regressione polinomiale**, che permette di approssimare relazioni non lineari tra x e y.
La forma generale di un modello polinomiale di grado d è:

$$f(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_d x^d$$

dove i parametri $\beta_0, \ldots, \beta_d$ vengono stimati a partire dai dati osservati.
Il numero totale di parametri liberi del modello è:

$$p = d + 1$$

(uno per ciascun coefficiente β, incluso l'intercetta).

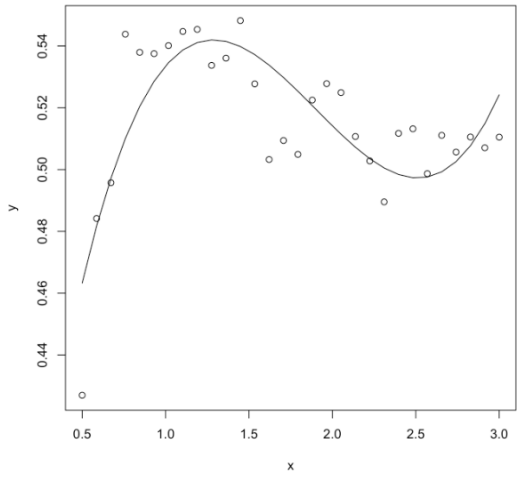
Scelta del grado d

In generale, non disponiamo di informazioni a priori sul grado ottimale del polinomio. Un approccio esplorativo consiste quindi nel provare tutti i gradi da $d = 0$ fino a $d = n - 1$, ossia fino al caso in cui il numero di parametri p coincide con il numero di osservazioni n.
Questo ultimo caso corrisponde a un modello **di interpolazione esatta**, in cui il modello passa per tutti i punti osservati e il MSE_{Tr} risulta esattamente nullo.

Esempio pratico in R: modello di grado 3

Consideriamo ora un modello polinomiale di terzo grado ($d = 3$):

```
# Creazione del test set
test <- data.frame(x = train$x, y = df$y.tomorrow)

# Stima del modello polinomiale di grado 3
fit <- lm(y ~ poly(x, degree = 3), train)
yhat <- predict(fit, newdata = test)

# Visualizzazione del fit
plot(y ~ x, train)
lines(yhat ~ x, train)
```



Il grafico mostra la curva del modello polinomiale di grado 3 sovrapposta ai dati di training. La domanda è: *il modello rappresenta bene la relazione tra x e y?*

1.3.1 Calcolo dell'errore di training

Possiamo valutare la bontà dell'adattamento tramite l'errore quadratico medio sul training set:

$$MSE_{Tr} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}(x_i))^2$$

In R:

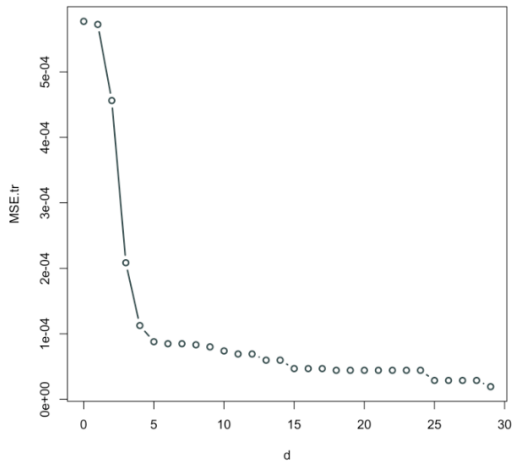
```
# Calcolo dell'errore di training
MSE.tr <- mean((train$y - yhat)^2)
MSE.tr
# Output: [1] 0.0002085353
```

Il valore di MSE_{Tr} è molto basso, suggerendo che il modello di grado 3 si adatta bene ai dati di training.

Analisi per tutti i gradi

Ripetiamo ora il calcolo del MSE_{Tr} per tutti i gradi da 0 a $n-1$.

```
n  <- nrow(train)
ds <- 0:(n - 1)      # gradi polinomiali
ps <- ds + 1         # numero di parametri

# Funzione per stimare un modello polinomiale di grado d
fun <- function(d) if (d == 0) lm(y ~ 1, train) else
                   lm(y ~ poly(x, degree = d, raw = TRUE), train)

# Lista di modelli
fits <- sapply(ds, fun)

# Calcolo dell'MSE di training per ciascun grado
MSEs.tr <- unlist(lapply(fits, deviance)) / n

# Grafico
plot(ds, MSEs.tr, type = "b", xlab = "d", ylab = "MSE.tr",
     col = "darkslategray", lwd = 2)
```



Il grafico mostra come il MSE_{Tr} diminuisca all'aumentare del grado del polinomio: più il modello è complesso, migliore è la sua capacità di adattarsi ai dati.

Osservazione cruciale

Un caso particolare si verifica quando $p = n$, cioè quando il polinomio ha grado $d = n - 1$. In questo scenario, il modello passa **esattamente per tutti i punti**:

$$MSE_{Tr} = 0$$

Tuttavia, questo risultato non deve trarci in inganno: un MSE_{Tr} nullo non implica un buon modello predittivo. Anzi, spesso significa che il modello ha **overfittato** i dati, adattandosi anche al rumore anziché solo alla struttura vera di $f(x)$.

1.3.2 Il vero obiettivo: minimizzare l'errore di previsione

Lo scopo ultimo dell'apprendimento statistico non è minimizzare l'errore di training, ma ridurre l'errore di previsione su dati nuovi. Per un nuovo punto (x_0, y_0) non usato per la stima, l'errore quadratico di previsione è:

$$(y_0 - \hat{f}(x_0))^2$$

Il **problema fondamentale dell'apprendimento statistico** è quindi che:

$$\text{un basso } MSE_{Tr} \not\Rightarrow \text{ un basso } MSE_{Te}.$$

In altre parole, un modello che si adatta perfettamente ai dati noti può comportarsi molto male su dati nuovi. Questo fenomeno rappresenta il cuore del **bias-variance trade-off**.

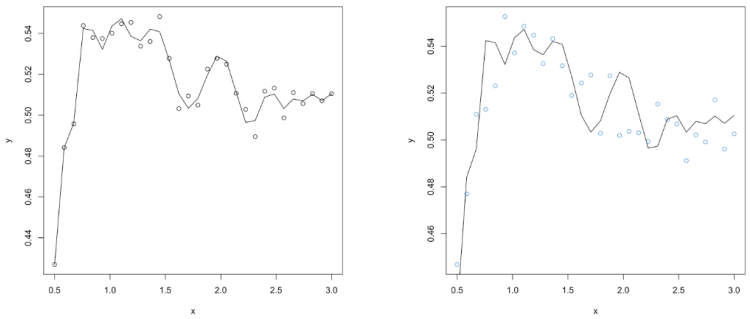
1.4 Overfitting

Come già accennato, il nostro obiettivo è ottenere una stima $\hat{f}$ basata sui dati di training, in grado di prevedere accuratamente i valori futuri generati dallo stesso processo. Per valutare la qualità delle previsioni, possiamo "simulare il futuro" confrontando le predizioni ottenute ieri con i dati di domani, cioè utilizzando il modello stimato sul training set per predire i valori del test set.
Consideriamo ora un modello polinomiale di grado elevato, ad esempio $d = 20$.

Esempio pratico in R

```
# Fit di un polinomio di 20° grado
fit <- lm(y ~ poly(x, degree = 20), train)
yhat <- predict(fit, newdata = test)

# Matrice del disegno (design matrix)
X <- model.matrix(fit)

# Grafici
plot(y ~ x, train)                    # dati di training
lines(yhat ~ x, train)                # fit del modello
plot(y ~ x, test, col = 4)            # dati di test
lines(yhat ~ x, train)
```

La funzione `poly()` del pacchetto base stats genera una matrice di polinomi **ortogonali**, che evitano collinearità numeriche. L'opzione `raw = TRUE`, invece, produce i **polinomi grezzi**, i cui termini sono fortemente correlati tra loro — una fonte potenziale di instabilità numerica nei modelli di grado elevato.



Analisi del risultato

Il grafico risultante mostra:

- i dati di **ieri** (training, a sinistra);
- i dati di **domani** (test, a destra);
- e la curva del polinomio di grado 20.

È evidente che il modello di 20° grado si adatta *quasi perfettamente* ai dati di training (seguendo anche le piccole oscillazioni casuali) ma risulta altamente instabile quando applicato ai dati di test. Questa perdita di capacità predittiva è un chiaro esempio di **overfitting**.

> *L'overfitting si verifica quando il modello apprende non solo la struttura sottostante del fenomeno, ma anche il rumore casuale dei dati di training.*

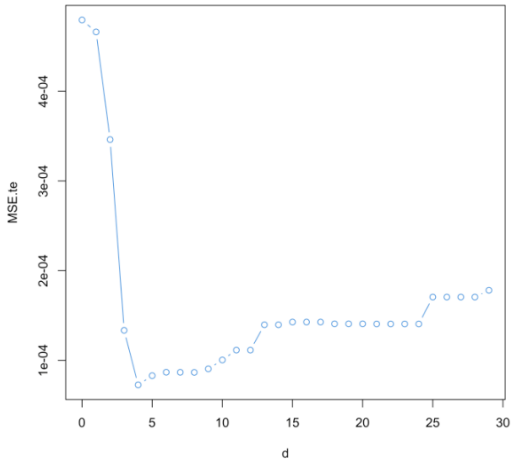
Il risultato è un modello che ha un errore di training molto basso ma un errore di test elevato.

1.4.1 Comportamento del MSE di test

Calcoliamo ora il MSE_{Te} per tutti i possibili gradi polinomiali $d = 0, \ldots, n-1$.

```
# Predizioni per tutti i gradi
yhats <- lapply(fits, predict)

# Calcolo del MSE di test
MSEs.te <- unlist(lapply(yhats,
               function(yhat) mean((test$y - yhat)^2))
               )

# Grafico dell'andamento del MSE di test
plot(ds, MSEs.te, type = "b", col = 4, xlab = "d", ylab = "MSE.te")

# Identificazione del grado ottimale
ds[which.min(MSEs.te)]
## [1] 4
```

Il grafico ottenuto mostra che il MSE_{Te} **diminuisce inizialmente** all'aumentare del grado del polinomio, poiché il modello diventa più flessibile e riesce a catturare meglio la relazione tra x e y. Tuttavia, oltre un certo punto, il MSE_{Te} **inizia a crescere** di nuovo.



Interpretazione del risultato

Il comportamento tipico dell'errore di test può essere riassunto così:

- Per modelli troppo semplici (basso grado d): il modello non riesce a rappresentare la vera struttura dei dati $\Rightarrow$ alto bias, alto MSE_{Te}.
- Per modelli troppo complessi (alto grado d): il modello segue il rumore anziché la tendenza generale $\Rightarrow$ alta varianza, di nuovo alto MSE_{Te}.

In questo esempio, il grado che minimizza l'errore di test è $d = 4$. Questo rappresenta un buon compromesso tra **bias** e **varianza**: il modello è sufficientemente flessibile da adattarsi ai dati, ma non così complesso da perdere la capacità di generalizzare.

Osservazione finale

Il fenomeno dell'overfitting è cruciale in ogni contesto di *machine learning* e statistica predittiva. Riconoscerlo e prevenirlo è essenziale per costruire modelli robusti. Le sezioni successive introdurranno formalmente il **bias-variance trade-off**, che spiega matematicamente il comportamento appena osservato del MSE_{Te}.

1.5 Bias–Variance decomposition

Nel **Fixed-X setting** disponiamo di due insiemi di osservazioni sugli stessi punti $x_1, \dots, x_n$:

$$(x_1, y_1), \dots, (x_n, y_n) \quad \text{(training)} \qquad (x_1, y_1^*), \dots, (x_n, y_n^*) \quad \text{(test)}.$$

Assumiamo che i predittori $x_1, \dots, x_n$ siano **fissati** (non aleatori) e che i **valori di test coincidano con quelli di training**.

Notazione in forma matriciale. Vettori delle risposte:

$$\mathbf{y} \in R^{n\times 1} = \begin{bmatrix} y_1 & \cdots & y_n \end{bmatrix}^\top, \qquad \mathbf{y}^* \in R^{n\times 1} = \begin{bmatrix} y_1^* & \cdots & y_n^* \end{bmatrix}^\top.$$

Matrice del disegno:

$$\mathbf{X} \in R^{n\times p} = \begin{bmatrix} x_{11} & \cdots & x_{1p} \\ x_{21} & \cdots & x_{2p} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{np} \end{bmatrix} = \begin{bmatrix} \mathbf{x}_1^\top \\ \mathbf{x}_2^\top \\ \vdots \\ \mathbf{x}_n^\top \end{bmatrix}.$$

Modello generativo. Per $i = 1, \ldots, n$,

$$Y_i = f(x_i) + \varepsilon_i, \qquad Y_i^* = \hat{f}(x_i) + \varepsilon_i^*,$$

dove f è la **funzione di regressione** (segnale) e gli errori sono i.i.d. con

$$E[\varepsilon] = 0, \qquad \mathrm{Var}(\varepsilon) = \sigma^2.$$

1.5.1 Prediction error

Nel Fixed-X definiamo l'errore di previsione atteso come

$$\mathrm{Err}_F = E(MSE_{Te}) = E\left[\frac{1}{n}\sum_{i=1}^{n}\big(Y_i^* - \hat{f}(x_i)\big)^2\right] = \frac{1}{n}\sum_{i=1}^{n} E\Big[\big(Y_i^* - \hat{f}(x_i)\big)^2\Big],$$

dove l'aspettativa è presa rispetto alle variabili aleatorie $\{Y_i\}$ e $\{Y_i^*\}$ (a parità di $\{x_i\}$).

1.5.2 Fonti di errore

Errore irriducibile. Anche la vera f non può prevedere perfettamente:

$$E\big[(Y - f(X))^2\big] = \sigma^2.$$

Bias. Se la classe di modelli non contiene (o approssima male) la vera f, emerge un errore sistematico tra $E[\hat{f}(x)]$ e $f(x)$.
Varianza. Se $\hat{f}$ cambia molto al variare del campione di training (ri-addestrando su dati **simili**), l'errore aumenta per eccessiva variabilità del predittore.

1.5.3 Reducible e irreducible error

Fissato x_i, vogliamo prevedere y_i^* con $\hat{y}_i^* = \hat{f}(x_i)$. Allora

$$\begin{aligned} E\Big[(Y_i^* - \hat{Y}_i^*)^2\Big] &= E\Big[(f(x_i) + \varepsilon_i^* - \hat{f}(x_i))^2\Big] \\ &= \underbrace{E\Big[(f(x_i) - \hat{f}(x_i))^2\Big]}_{\text{errore } \textbf{riducibile}} + \underbrace{\mathrm{Var}(\varepsilon_i^*)}_{\text{errore } \textbf{irriducibile}} \\ &= E\Big[(f(x_i) - \hat{f}(x_i))^2\Big] + \sigma^2. \end{aligned}$$

Scomposizione dell'errore riducibile. Aggiungi e sottrai $E[\hat{f}(x_i)]$:

$$\begin{aligned} E\Big[(f(x_i) - \hat{f}(x_i))^2\Big] &= \big(E[\hat{f}(x_i)] - f(x_i)\big)^2 + \mathrm{Var}\big(\hat{f}(x_i)\big) \\ &= \underbrace{\mathrm{Bias}^2\big(\hat{f}(x_i)\big)}_{\text{errore sistematico}} + \underbrace{\mathrm{Var}\big(\hat{f}(x_i)\big)}_{\text{errore da variabilità}}. \end{aligned}$$

Decomposizione finale (Fixed-X)

Mediando su $i = 1, \dots, n$,

$$\begin{aligned} \mathrm{Err}_F &= \sigma^2 + \frac{1}{n}\sum_{i=1}^{n}\big(E[\hat{f}(x_i)] - f(x_i)\big)^2 + \frac{1}{n}\sum_{i=1}^{n}\mathrm{Var}\big(\hat{f}(x_i)\big) \\ &= \underbrace{\sigma^2}_{\text{irriducibile}} + \underbrace{\mathrm{Bias}^2}_{\text{riducibile}} + \underbrace{\mathrm{Var}}_{\text{riducibile}} . \end{aligned}$$

Conclusione: bias e varianza sono in tensione; non si possono minimizzare simultaneamente. Occorre un **trade-off**.

Risultati teorici per il modello lineare

Consideriamo la regressione lineare (o polinomiale scritta come lineare nei parametri) con $p = d + 1$ parametri e $\mathbf{X}$ a rango pieno. Lo stimatore dei coefficienti è

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{y}, \qquad \hat{f}(x_i) = \mathbf{x}_i^\top\hat{\boldsymbol{\beta}}.$$

Bias del predittore. Poiché $E[\mathbf{y}] = \mathbf{f}$ con $\mathbf{f} = (f(x_1), \dots, f(x_n))^\top$,

$$\begin{aligned} E[\hat{f}(x_i)] - f(x_i) &= \mathbf{x}_i^\top(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\, E[\mathbf{y}] - f(x_i) \\ &= \mathbf{x}_i^\top(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{f} - f(x_i). \end{aligned}$$

Se la vera f appartiene allo spazio colonnare di $\mathbf{X}$ (modello correttamente specificato), il bias è nullo.

Varianza del predittore.

$$\mathrm{Var}(\hat{\boldsymbol{\beta}}) = \sigma^2(\mathbf{X}^\top\mathbf{X})^{-1} \quad\Longrightarrow\quad \mathrm{Var}\big(\hat{f}(x_i)\big) = \mathbf{x}_i^\top\,\mathrm{Var}(\hat{\boldsymbol{\beta}})\,\mathbf{x}_i = \sigma^2\,\mathbf{x}_i^\top(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{x}_i.$$

Sommando su i e usando la traccia della **hat matrix** $\mathbf{H} = \mathbf{X}(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top$,

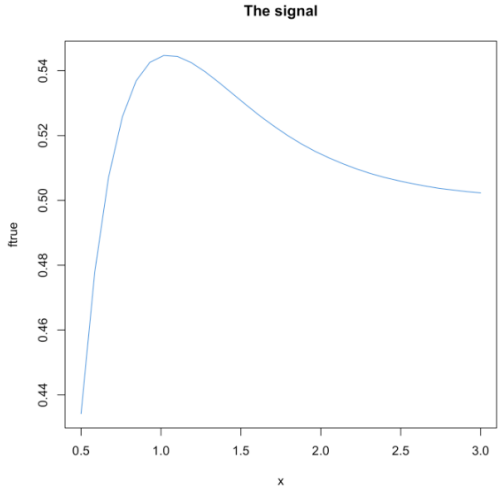
$$\sum_{i=1}^{n}\mathrm{Var}\big(\hat{f}(x_i)\big) = \sigma^2\,\mathrm{tr}(\mathbf{H}) = \sigma^2\,p.$$

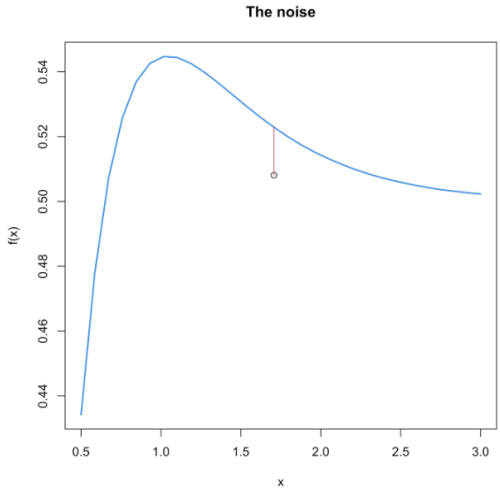
Osservazione: nel modello lineare, la **somma** delle varianze di predizione dipende da p e non dalla forma di $f(x)$.

Esempi in R: segnale vero, bias e varianza

Segnale vero.

```
# true regression function
ftrue <- c(0.4342, 0.4780, 0.5072, 0.5258, 0.5369, 0.5426, 0.5447,
           0.5444, 0.5425, 0.5397, 0.5364, 0.5329, 0.5294, 0.5260,
           0.5229, 0.5200, 0.5174, 0.5151, 0.5131, 0.5113, 0.5097,
           0.5083, 0.5071, 0.5061, 0.5052, 0.5044, 0.5037, 0.5032,
           0.5027, 0.5023)
x <- seq(.5, 3, length = 30)
plot(x, ftrue, type = "l", col = 4, main="The signal")
```

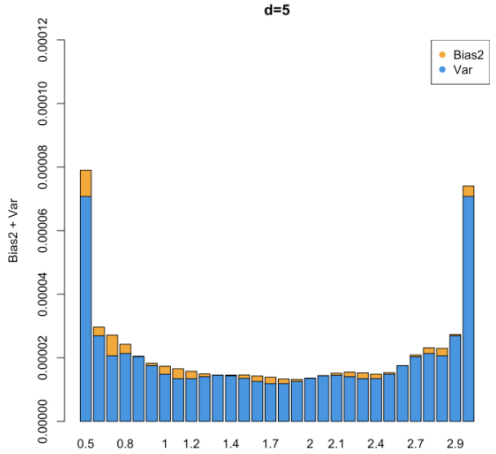




Bias^2 e Var per $d = 5$.

```
d <- 5
sigmatrue <- 0.01

# design matrix
X <- model.matrix(lm(ftrue ~ poly(x, degree = d)))
invXtX <- solve(crossprod(X))

# Bias^2 punto-per-punto
Bias2 <- ( apply(X, 1, function(x)
  x %*% invXtX %*% t(X) %*% ftrue) - ftrue )^2

# Var punto-per-punto
```

```
Var <- apply(X, 1, function(x)
  sigmatrue^2 * t(x) %*% invXtX %*% x )

barplot(Bias2 + Var, ylab = "Bias2 + Var", names.arg = round(x, 1),
        col = "orange", ylim=c(0,0.00012), main="d=5")
barplot(Var, add = TRUE, col = 4, names.arg=" ")
legend("topright", c("Bias2", "Var"), col=c("orange", 4), pch = c(19, 19))
```



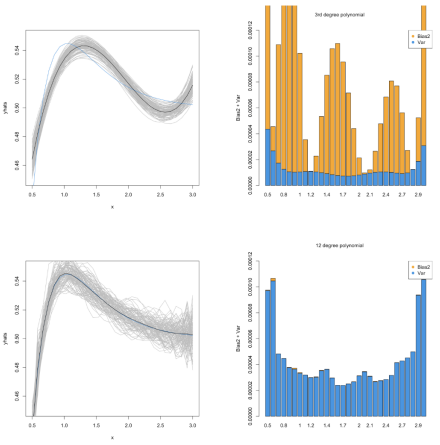
Confronto errore teorico vs empirico (simulazione).

```
# Expected prediction error (teorico)
# MSE.te teorico
sigmatrue^2 + mean(Bias2) + mean(Var)
# [1] 0.0001217199

# via simulazione
ErrF <- function(d){
  y     <- ftrue + rnorm(n, 0, sigmatrue)
  fit   <- lm(y ~ poly(x, degree = d))
  yhat  <- fitted(fit)
  y_new <- ftrue + rnorm(n, 0, sigmatrue)
  mean( (yhat - y_new)^2 )
}
B <- 1000
set.seed(123)
d <- 5
mean(replicate(B, ErrF(d=5)))
# [1] 0.0001184768
```

Simulation: stima empirica di Bias^2 e Var

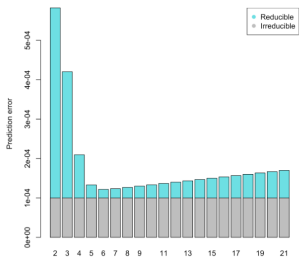
```
# setting
```

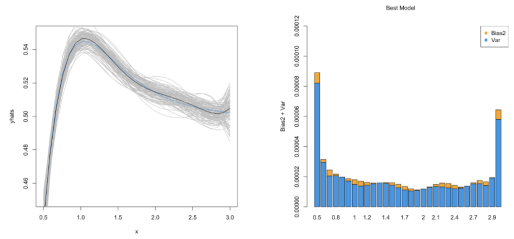
```
n <- length(x)
d <- 3
p <- d + 1
B <- 100

# simulazione
sim <- function(d){
  y    <- ftrue + rnorm(n, 0, sigmatrue)
  fit  <- lm(y ~ poly(x, degree = d))
  fitted(fit)
}

set.seed(123)
yhats <- replicate(B, sim(d))
matplot(x, yhats, type="l", col="gray", lty=1, ylim=c(.45,.55))
lines(x, ftrue, col=4)

# valore atteso delle predizioni
Ave <- apply(yhats, 1, mean)
lines(x, Ave)

# componenti della decomposizione
Bias2 <- (ftrue - Ave)^2
Var   <- apply(yhats, 1, var)

barplot(Bias2 + Var, ylab = "Bias2 + Var", names.arg = round(x, 1),
        col = "orange", ylim=c(0,0.00012))
barplot(Var, add = TRUE, col = 4, names.arg = " ")
legend("topright", c("Bias2","Var"), col = c("orange", 4), pch = c(19, 19))
mtext("3rd degree polynomial", side = 3, line = -2, outer = TRUE)
```



Decomposizione di Err_F per tutti i gradi (senza simulazione)

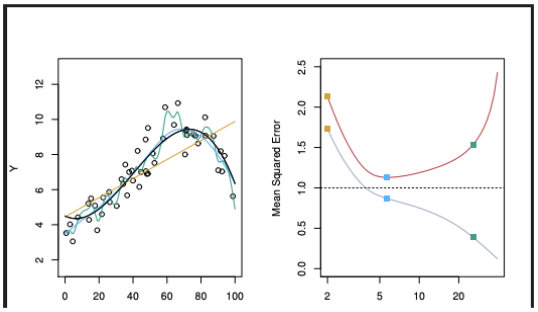
```
ds <- 1:20
ps <- ds + 1

Bias2s <- sapply(ps, function(p)
  mean( ( ftrue - fitted(lm(ftrue ~ poly(x, degree = (p - 1)))) )^2 )
)
Vars <- ps * (sigmatrue^2) / n
Reds <- Bias2s + Vars

barplot(Reds, ylab = "Reducible error", names.arg = ps, col = "orange")
barplot(Vars, add = TRUE, col = 4, names.arg = " ")
legend("topright", c("Bias2", "Var"), col = c("orange", 4), pch = c(19, 19))

# best model size
ds[which.min(Reds)]
# [1] 5

# errore totale = riducibile + irriducibile
Irr   <- rep(sigmatrue^2, length(ps))
ErrFs <- Reds + Irr
barplot(ErrFs, ylab = "Prediction error", names.arg = ps, col = 5)
barplot(Irr, add = TRUE, col = "gray", names.arg = " ")
legend("topright", c("Reducible", "Irreducible"),
       col = c(5, "gray"), pch = c(19, 19))
```



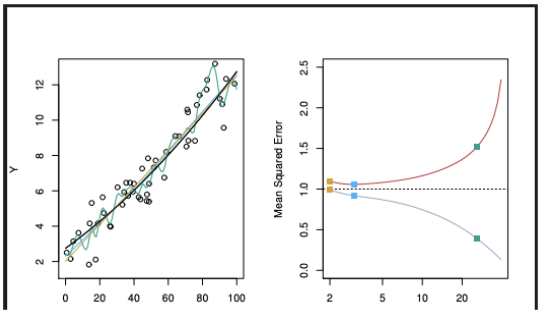


1.5.4 Assessing model accuracy: tre casi tipici (ISLR)

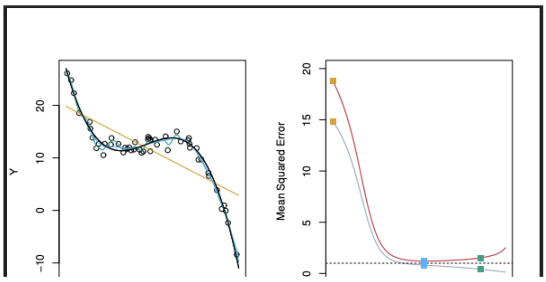
Caso 1: f non lineare (curva nera). Tre stime con flessibilità crescente: regressione lineare (arancio), spline molto liscia (blu), spline meno liscia (verde). A destra: MSE di training (grigio) e MSE di test (rosso). Il test MSE ha spesso una forma a "U": diminuisce con la flessibilità, poi risale per overfitting.



Caso 2: f circa lineare. Metodi semplici (lineare) hanno **bias** piccolo e varianza contenuta: spesso vincono in test. Un metodo con MSE di training piccolo ma MSE di test grande **overfitta**.



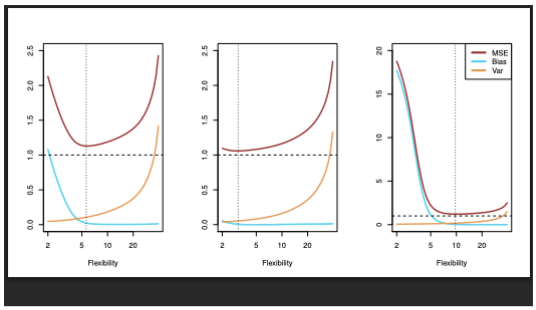
Caso 3: f altamente non lineare. La regressione lineare è inadeguata: **alto bias** $\Rightarrow$ test MSE elevato. Servono modelli più flessibili (ma controllando la varianza).



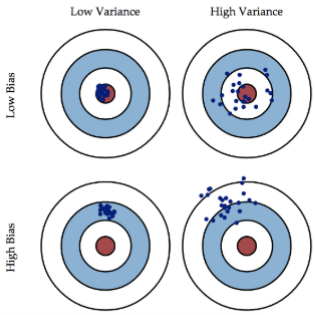
Relazione bias–varianza–MSE di test. All'aumentare della flessibilità di $\hat{f}$:

$$\text{Bias}^2 \downarrow \quad \text{ma} \quad \text{Var} \uparrow .$$

La scelta della flessibilità ottimale (minimo dell'errore medio di test) è quindi un **bias–variance trade-off**.



Source: ISLR p. 36



Nota computazionale. In `R`, `poly(x, degree)` genera **polinomi ortogonali** (migliore condizionamento numerico); con `raw=TRUE` si usano i polinomi "grezzi", spesso fortemente collineari per gradi elevati.

1.6 Esercizio: Bias–Variance trade-off (Fixed-X, $n = 50$, $p = 30$)

Obiettivo e idea generale

Vogliamo **misurare sperimentalmente** (via simulazione) come si scompone l'errore di previsione di un modello lineare in:

$$\text{Err}_F = \underbrace{\sigma^2}_{\text{irriducibile}} + \underbrace{\text{Bias}^2}_{\text{riducibile}} + \underbrace{\text{Var}}_{\text{riducibile}} ,$$

in un'impostazione **Fixed-X** con $X \in R^{n \times p}$ fissata, $n = 50$, $p = 30$, e rumore $\varepsilon \sim \mathcal{N}(0, I_n)$ (quindi $\sigma^2 = 1$).

Setup del problema

Generiamo una matrice dei predittori con colonne debolmente correlate:

$$X_{ij} \overset{\text{iid}}{\sim} \mathcal{N}(0,1), \qquad X \in R^{50\times 30}.$$

Definiamo un vettore di **veri** coefficienti (oracle) $\beta^\star \in R^{30}$ con *10 coefficienti grandi* (tra 0.5 e 1) e *20 piccoli* (tra 0 e 0.3). Così otteniamo un segnale

$$\mu \;=\; X\beta^\star \in R^{50}.$$

Perché 10 grandi e 20 piccoli? Per simulare uno scenario realistico: poche variabili con effetto forte (**segnale**) e molte con effetto debole (**quasi rumore**). Questo aiuta a capire come il modello recupera il segnale vero e dove accumula varianza.

Cosa stiamo stimando (Fixed-X)

Modello lineare corretto:

$$y \;=\; \mu+\varepsilon \;=\; X\beta^\star+\varepsilon, \qquad \varepsilon\sim\mathcal{N}(0,I_n).$$

Stima OLS:

$$\hat\beta \;=\; (X^\top X)^{-1}X^\top y, \qquad \hat\mu \;=\; X\hat\beta.$$

Nel Fixed-X, se il modello è correttamente specificato e $p<n$ con $X^\top X$ invertibile:

$$E[\hat\mu]=\mu \;\Rightarrow\; \text{Bias}^2\approx 0, \quad \frac{1}{n}\sum_{i=1}^{n}\text{Var}(\hat\mu_i) \;=\; \frac{\sigma^2 p}{n}.$$

Errore di previsione per un nuovo $y'=\mu+\varepsilon'$ indipendente:

$$\text{Err}_F \;=\; E\left[\frac{1}{n}\sum_{i=1}^{n}(y_i'-\hat\mu_i)^2\right] \;=\; \sigma^2 \;+\; \underbrace{\text{Bias}^2}_{\approx 0} \;+\; \frac{\sigma^2 p}{n}.$$

Con $n=50$, $p=30$, $\sigma^2=1$:

$$\boxed{\text{Bias}^2\approx 0, \quad \text{Var}=\tfrac{p}{n}=0.6, \quad \text{Err}_F=1+0.6=\mathbf{1.6}.}$$

Procedura standard di simulazione (concettuale)

1. **Fissa** X una volta per tutte (Fixed-X) e costruisci $\beta^\star$, poi $\mu=X\beta^\star$.
2. Per $b=1,\dots,B$ (con $B=100$):
   (a) Genera $y^{(b)}=\mu+\varepsilon^{(b)}$ con $\varepsilon^{(b)}\sim\mathcal{N}(0,I_n)$.
   (b) Stima OLS: $\hat\beta^{(b)}=(X^\top X)^{-1}X^\top y^{(b)}$ e $\hat\mu^{(b)}=X\hat\beta^{(b)}$.
   (c) Genera un nuovo vettore indipendente $y'^{(b)}=\mu+\varepsilon'^{(b)}$.
   (d) Registra l'errore di previsione:

$$\text{PE}^{(b)}=\frac{1}{n}\sum_{i=1}^{n}\big(y_i'^{(b)}-\hat\mu_i^{(b)}\big)^2.$$

3. **Stime finali**:

$$\widehat{\mathrm{Err}}_F = \frac{1}{B}\sum_{b=1}^{B} \mathrm{PE}^{(b)}, \quad \widehat{\mathrm{Bias}^2} = \frac{1}{n}\sum_{i=1}^{n} (\bar{\hat{\mu}}_i - \mu_i)^2, \quad \widehat{\mathrm{Var}} = \frac{1}{n}\sum_{i=1}^{n} \mathrm{Var}_b(\hat{\mu}_i^{(b)}),$$

con $\bar{\hat{\mu}} = \frac{1}{B}\sum_b \hat{\mu}^{(b)}$.

Ci aspettiamo empiricamente:

$$\widehat{\mathrm{Err}}_F \approx 1 + \widehat{\mathrm{Bias}^2} + \widehat{\mathrm{Var}} \approx 1.6.$$

Codice R

```
# Fissiamo X e il "vero" segnale
set.seed(0)
n <- 50; p <- 30
X <- matrix(rnorm(n*p), nrow = n)

bstar <- c(runif(10, 0.5, 1.0), runif(20, 0.0, 0.3))  # 10 forti, 20 deboli
mu <- as.numeric(X %*% bstar)                          # segnale vero
sigma2 <- 1

B <- 100
PEs    <- numeric(B)            # prediction errors
HatMus <- matrix(NA_real_, n, B)  # fitted values per run

for (b in 1:B) {
  y  <- mu + rnorm(n, 0, sqrt(sigma2))    # y = mu + rumore
  fit <- lm(y ~ X - 1)                    # -1: niente intercetta (mu = X beta*)
  yhat <- as.numeric(fitted(fit))         # hat(mu)
  HatMus[, b] <- yhat

  yprime <- mu + rnorm(n, 0, sqrt(sigma2))      # nuovo y' indipendente
  PEs[b] <- mean((yprime - yhat)^2)             # errore di previsione
}

# Stime Bias^2 e Var
HatMu_bar <- rowMeans(HatMus)                 # E_hat[hat(mu)]
Bias2_hat <- mean( (HatMu_bar - mu)^2 )       # bias^2 medio
Var_hat   <- mean( apply(HatMus, 1, var) )    # varianza media sui 50 punti
ErrF_hat  <- mean(PEs)

# Confronto con la teoria
theory_Var  <- sigma2 * p / n   # = 1 * 30/50 = 0.6
theory_ErrF <- sigma2 + theory_Var  # = 1.6

c(ErrF_hat = ErrF_hat,
  OnePlusBias2PlusVar = 1 + Bias2_hat + Var_hat,
  Theory_Var = theory_Var,
  Theory_ErrF = theory_ErrF)
```

Cosa imparare (in pratica)

- **Training vs Test**: aumentare la flessibilità riduce sempre il training MSE; il test MSE segue una curva a "U" (prima scende, poi sale per overfitting).
- **Errore irriducibile** σ^2: non può essere eliminato (dipende dal rumore).
- **Errore riducibile** = Bias^2 + Var: va bilanciato scegliendo la **giusta complessità**.
- **Fixed-X, OLS corretto**: $\text{Bias}^2 \simeq 0$ e $\text{Var} = \sigma^2 p/n \Rightarrow \text{Err}_F = \sigma^2(1 + p/n)$.
- **Nota tecnica**: qui $p < n$ quindi $X^\top X$ è invertibile. Se $p \geq n$ servono metodi con **regolarizzazione** (ridge, lasso, ecc.).

Chapter 2

Avoiding overfitting: information criteria and cross-validation

2.1 Ottimismo

L'idea di base è che il **training error** tende ad essere troppo ottimistico: il modello sembra funzionare bene perché è stato valutato sugli stessi dati che ha utilizzato per imparare. L'**ottimismo** misura quanto questo errore medio di training sottostima l'errore reale che ci aspetteremmo su nuovi dati. Formalmente:

$$\text{Opt} = E(MSE_{Te}) - E(MSE_{Tr}).$$

Se riusciamo a stimare questo valore, possiamo stimare anche l'errore di previsione sui dati futuri:

$$\widehat{\text{Err}}_F = MSE_{Tr} + \widehat{\text{Opt}}.$$

Caso Fixed-X

In questo contesto i valori di x_i sono considerati noti e fissi, mentre i valori di Y_i sono realizzazioni casuali. Definiamo due insiemi di risposte:

$$Y_i \quad \text{(campione di training)}, \qquad Y_i^* \quad \text{(nuovo campione indipendente)}.$$

L'ottimismo può allora essere espresso come:

$$\text{Opt}_F = E\left[\frac{1}{n}\sum_{i=1}^{n}(Y_i^* - \hat{f}(x_i))^2 - \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{f}(x_i))^2\right].$$

L'obiettivo è capire in che modo questa differenza dipende dalla varianza e dalla covarianza tra i termini Y_i e $\hat{f}(x_i)$.

Decomposizione della varianza e covarianza

Consideriamo per ogni i:

$$E\Big[(Y_i - \hat{f}(x_i))^2\Big] = \mathrm{Var}(Y_i) + \mathrm{Var}(\hat{f}(x_i)) - 2\,\mathrm{Cov}(Y_i, \hat{f}(x_i)) + \big(E[Y_i] - E[\hat{f}(x_i)]\big)^2.$$

In modo analogo, per il nuovo campione Y_i^*:

$$E\Big[(Y_i^* - \hat{f}(x_i))^2\Big] = \mathrm{Var}(Y_i^*) + \mathrm{Var}(\hat{f}(x_i)) - 2\,\mathrm{Cov}(Y_i^*, \hat{f}(x_i)) + \big(E[Y_i^*] - E[\hat{f}(x_i)]\big)^2.$$

Poiché Y_i e Y_i^* sono **i.i.d.** (indipendenti ma distribuiti allo stesso modo), vale:

$$\mathrm{Var}(Y_i) = \mathrm{Var}(Y_i^*), \quad E(Y_i) = E(Y_i^*), \quad \mathrm{Cov}(Y_i^*, \hat{f}(x_i)) = 0.$$

Sostituendo e semplificando, otteniamo:

$$E\Big[(Y_i^* - \hat{f}(x_i))^2\Big] = \mathrm{Var}(Y_i) + \mathrm{Var}(\hat{f}(x_i)) + \big(E[Y_i] - E[\hat{f}(x_i)]\big)^2,$$

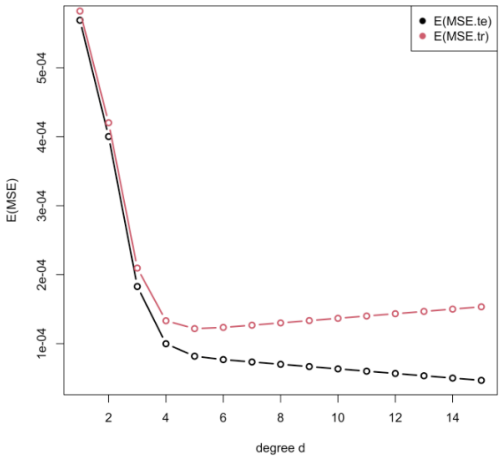
$$E\Big[(Y_i - \hat{f}(x_i))^2\Big] = \mathrm{Var}(Y_i) + \mathrm{Var}(\hat{f}(x_i)) - 2\,\mathrm{Cov}(Y_i, \hat{f}(x_i)) + \big(E[Y_i] - E[\hat{f}(x_i)]\big)^2.$$

Differenza tra errore di test e di training. Sottraendo membro a membro e poi mediando su $i = 1, \ldots, n$:

$$E(MSE_{Te}) = E(MSE_{Tr}) + \frac{2}{n}\sum_{i=1}^{n} \mathrm{Cov}(Y_i, \hat{f}(x_i)).$$

Quindi:

$$\boxed{\mathrm{Opt}_F = \frac{2}{n}\sum_{i=1}^{n} \mathrm{Cov}(Y_i, \hat{f}(x_i))} \quad \Longrightarrow \quad \boxed{MSE_{Te} \approx MSE_{Tr} + \frac{2}{n}\sum_{i=1}^{n} \mathrm{Cov}(Y_i, \hat{f}(x_i))}.$$



Interpretazione intuitiva

Più la predizione $\hat{f}(x_i)$ è "legata" a Y_i (cioè più il modello si adatta ai dati di training), maggiore è la covarianza e quindi maggiore è l'ottimismo. In altre parole, quando un modello impara troppo dai dati visti, il suo errore di training sarà molto più basso del vero errore di test.

Stima pratica dell'ottimismo

L'equazione precedente fornisce un modo generale per stimare l'errore di previsione:

$$\widehat{\mathrm{Err}}_F = MSE_{Tr} + \widehat{\mathrm{Opt}}_F, \qquad \widehat{\mathrm{Opt}}_F \approx \frac{2}{n}\sum_{i=1}^{n} \widehat{\mathrm{Cov}}(Y_i, \hat{f}(x_i)).$$

Nel modello lineare con errori omoschedastici ($\mathrm{Var}(\varepsilon) = \sigma^2 I$) e **hat matrix H**,

$$\sum_{i=1}^{n} \mathrm{Cov}(Y_i, \hat{f}(x_i)) = \sigma^2 \,\mathrm{tr}(\mathbf{H}).$$

Il termine $\mathrm{tr}(\mathbf{H})$ rappresenta i **gradi di libertà effettivi** del modello. Nel caso della regressione lineare classica, $\mathrm{tr}(\mathbf{H}) = p$, quindi:

$$\mathrm{Opt}_F = \frac{2\sigma^2 p}{n} \quad \Longrightarrow \quad \widehat{\mathrm{Err}}_F = MSE_{Tr} + \frac{2\hat{\sigma}^2 p}{n}.$$

Questa è l'idea alla base dei criteri di selezione come il C_p **di Mallows** o l'**AIC** nella regressione gaussiana: aggiungere una **penalità di complessità** (proporzionale ai gradi di libertà) al training error per ottenere una stima più realistica dell'errore di previsione.

Procedura operativa:

1. Calcolare il MSE_{Tr} del modello stimato.
2. Stimare la varianza del rumore $\hat{\sigma}^2$ e la complessità del modello (es. numero di parametri p o $\mathrm{tr}(H)$).
3. Calcolare l'ottimismo stimato:
$$\widehat{\mathrm{Opt}}_F = \frac{2\hat{\sigma}^2 \,\mathrm{df}}{n},$$
dove df rappresenta i gradi di libertà.
4. Stimare l'errore di previsione:
$$\widehat{\mathrm{Err}}_F = MSE_{Tr} + \widehat{\mathrm{Opt}}_F.$$

2.1.1 Ottimismo per il modello lineare

Nel modello lineare possiamo calcolare in modo esplicito quanto l'errore di training sottostima l'errore di test. L'obiettivo è capire di quanto il modello risulta "troppo ottimista" quando lo valutiamo sugli stessi dati che ha usato per imparare.

$$\mathrm{Opt}_F = \frac{2}{n} \sum_{i=1}^{n} \mathrm{Cov}(Y_i, \hat{f}(x_i))$$

dove Y_i sono le risposte osservate e $\hat{f}(x_i)$ le predizioni. Questa formula dice che l'ottimismo è proporzionale alla **covarianza** tra il valore vero e la sua predizione: se il modello è molto "legato" ai dati osservati (alta correlazione), allora è più ottimista.

Caso lineare: nel modello lineare classico le predizioni si possono scrivere come:

$$\hat{\mathbf{f}} = \mathbf{HY}$$

dove $\mathbf{H} = \mathbf{X}(\mathbf{X}^\top \mathbf{X})^{-1}\mathbf{X}^\top$ è la **hat matrix**, detta così perché produce le stime $\hat{\mathbf{Y}} = \mathbf{HY}$. La matrice $\mathbf{H}$ ha due proprietà fondamentali:

$$\mathbf{H}^\top = \mathbf{H}, \quad \mathbf{H}^2 = \mathbf{H}.$$

Derivazione passo per passo:

$$\mathrm{Opt}_F = \frac{2}{n} \operatorname{tr}\big(\mathrm{Cov}(\mathbf{Y}, \hat{\mathbf{f}})\big)$$

Poiché $\hat{\mathbf{f}} = \mathbf{HY}$, si ha:

$$\mathrm{Cov}(\mathbf{Y}, \hat{\mathbf{f}}) = \mathrm{Cov}(\mathbf{Y}, \mathbf{HY})$$

La matrice $\mathbf{H}$ è costante rispetto a $\mathbf{Y}$, quindi:

$$\mathrm{Cov}(\mathbf{Y}, \mathbf{HY}) = \mathrm{Cov}(\mathbf{Y}, \mathbf{Y})\mathbf{H}^\top$$

Nel modello lineare:

$$\mathrm{Cov}(\mathbf{Y}, \mathbf{Y}) = \sigma^2 \mathbf{I}_n$$

perciò:

$$\mathrm{Cov}(\mathbf{Y}, \hat{\mathbf{f}}) = \sigma^2 \mathbf{H}.$$

Il tracciato è quindi:

$$\operatorname{tr}\big(\mathrm{Cov}(\mathbf{Y}, \hat{\mathbf{f}})\big) = \sigma^2 \operatorname{tr}(\mathbf{H}).$$

Poiché nel modello lineare $\operatorname{tr}(\mathbf{H}) = p$, otteniamo:

$$\boxed{\mathrm{Opt}_F = \frac{2\sigma^2 p}{n}}$$

e quindi:

$$\boxed{MSE_{Te} \approx MSE_{Tr} + \frac{2\sigma^2 p}{n}}.$$

Interpretazione

Questa formula ci dice che l'errore calcolato sui dati di training è sempre troppo ottimistico. Per stimare meglio l'errore di test, bisogna aggiungere un termine correttivo:

$$\frac{2\sigma^2 p}{n},$$

che dipende dalla varianza del rumore σ^2, dal numero di parametri p e dal numero di osservazioni n. Più grande è p, più aumenta la penalità e quindi la correzione di ottimismo.

Stima della varianza σ^2

In pratica, σ^2 non è noto. Lo si stima usando la somma dei residui al quadrato:

$$\hat{\sigma}^2 = \frac{\text{RSS}}{n-p} = \frac{\sum_{i=1}^{n}(y_i - \hat{f}(x_i))^2}{n-p}.$$

Poiché $\text{RSS} = n \cdot MSE_{Tr}$, la stima di σ^2 può anche essere scritta come:

$$\hat{\sigma}^2 = \frac{n \cdot MSE_{Tr}}{n-p}.$$

Errore di previsione stimato:

$$\widehat{\text{Err}}_F = MSE_{Tr} + \frac{2\hat{\sigma}^2 p}{n}.$$

Mallows' C_p

Questa quantità prende il nome di **Mallows' C_p** e si definisce come:

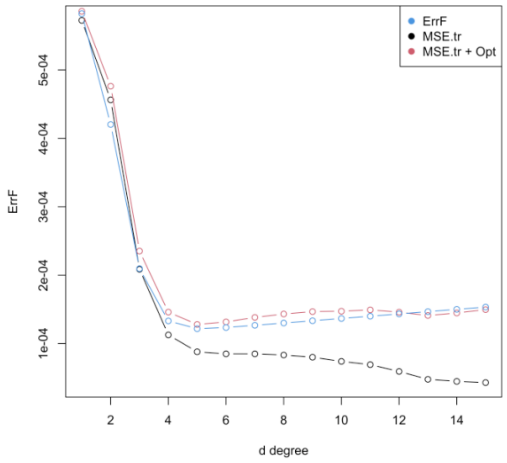
$$C_p = MSE_{Tr} + \frac{2\hat{\sigma}^2 p}{n}.$$

Il modello con il valore di C_p più basso è quello che ottiene il miglior compromesso tra qualità del fit e complessità. L'idea è che C_p aggiunge al training error una penalità che cresce con il numero di parametri p. In questo modo si evitano modelli troppo complessi che rischiano di overfittare i dati.

Esempio pratico (dati yesterday–tomorrow)

```
rm(list=ls())
library(readr)
df <- read_table("http://azzalini.stat.unipd.it/Book-DM/yesterday.dat")[-31,]
train <- data.frame(x = df$x, y = df$y.yesterday)

# Calcolo MSE.tr per polinomi di grado d = 1..15
n <- nrow(train)
ds <- 1:15
```

```
ps <- ds + 1
x <- seq(.5, 3, length = 30)
fun <- function(d) if (d == 0) lm(y ~ 1, train) else lm(y ~ poly(x, degree = d), trai
fits <- lapply(ds, fun)
MSEs.tr <- unlist(lapply(fits, deviance)) / n

# Parametri veri
sigmatrue <- 0.01
ftrue <- c(0.4342, 0.4780, 0.5072, 0.5258, 0.5369, 0.5426, 0.5447, 0.5444, 0.5425,
           0.5397, 0.5364, 0.5329, 0.5294, 0.5260, 0.5229, 0.5200, 0.5174, 0.5151,
           0.5131, 0.5113, 0.5097, 0.5083, 0.5071, 0.5061, 0.5052, 0.5044, 0.5037,
           0.5032, 0.5027, 0.5023)

# Calcolo errore teorico
Bias2s <- sapply(ps, function(p)
  mean((ftrue - fitted(lm(ftrue ~ poly(x, degree = (p - 1))))^2)
)
Vars <- ps * (sigmatrue^2) / n
ErrFs <- Bias2s + Vars + sigmatrue^2

# Stima ErrF con sigma nota
hatErrFs <- MSEs.tr + (2 * sigmatrue^2 * ps) / n
plot(ds, MSEs.tr, type = "b", xlab = "d degree", ylab = "ErrF")
lines(ds, hatErrFs, type = "b", col = 2)
lines(ds, ErrFs, type = "b", col = 4)
legend("topright", c("ErrF", "MSE.tr", "MSE.tr + Opt"),
       col = c(4, 1, 2), pch = 19)

ps[which.min(hatErrFs)]
## [1] 6
```

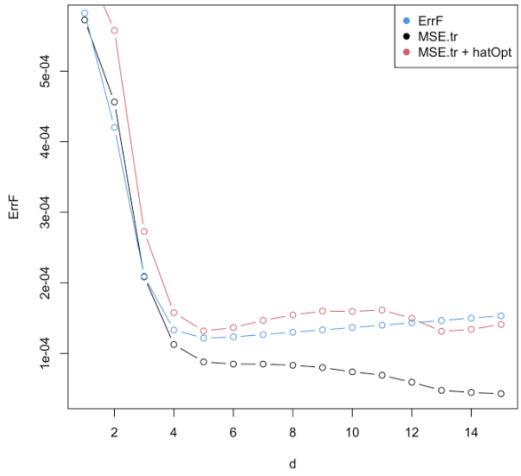


Nel caso con varianza nota ($\sigma^2 = 0.01^2$), l'errore stimato tramite ottimismo ($MSE_{Tr} + \text{Opt}$)

coincide quasi con l'errore teorico. Il grado ottimale del polinomio risulta essere $d = 6$.

Caso con varianza stimata:

```
hatsigma2 <- (n * MSEs.tr) / (n - ps)
Cps <- MSEs.tr + (2 * hatsigma2 * ps) / n
plot(ds, MSEs.tr, type = "b", xlab = "d", ylab = "ErrF")
lines(ds, Cps, type = "b", col = 2)
lines(ds, ErrFs, type = "b", col = 4)
legend("topright", c("ErrF", "MSE.tr", "MSE.tr + hatOpt"),
       col = c(4, 1, 2), pch = 19)

ds[which.min(Cps)]
## [1] 13
```

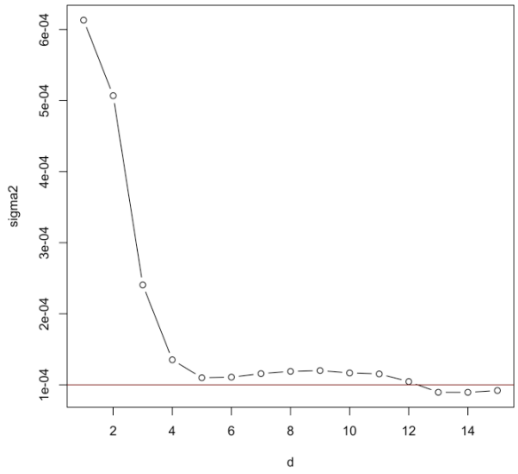



```
plot(ds, hatsigma2, type = "b", xlab = "d", ylab = "sigma2")
abline(h = sigmatrue^2, col = "red4")
```

Quando si usa la varianza stimata $\hat{\sigma}^2 = \frac{RSS}{n-p}$, la penalità di complessità cambia con il grado del modello, e la scelta ottimale può spostarsi. In questo esempio, il valore minimo di C_p si trova per $d = 13$, non per $d = 6$. La differenza nasce dal fatto che $\hat{\sigma}^2$ varia da modello a modello e può essere instabile quando il modello diventa molto flessibile.

Conclusione

- L'**ottimismo** misura quanto il training error è più basso del vero test error.
- Nei modelli lineari: $\text{Opt}_F = \frac{2\sigma^2 p}{n}$.
- La versione stimata porta al criterio di **Mallows'** C_p, usato per confrontare modelli di diversa complessità.



- Più grande è p, più forte è la penalità, per evitare l'overfitting.
- Se σ^2 non è noto, si può stimare dai residui o usare la **cross-validation** come alternativa generale.

2.2 Information criteria

2.2.1 AIC

Il criterio di **Akaike Information Criterion (AIC)** fornisce un modo generale per valutare e confrontare modelli stimati tramite il principio di **massima verosimiglianza (MLE)**. L'obiettivo dell'AIC è stimare quanto bene un modello potrà generalizzare su nuovi dati, tenendo conto non solo della bontà di adattamento ma anche della complessità del modello.

Definizione generale

L'AIC è definito come:

$$AIC = -2 \cdot \text{log-likelihood}(\hat{\beta}, \hat{\sigma}^2) + 2p,$$

dove:

- il primo termine, $-2\cdot$log-likelihood, misura quanto bene il modello si adatta ai dati osservati (un valore minore indica un fit migliore);
- il secondo termine, $2p$, è una penalità che aumenta con il numero di parametri p, per evitare modelli troppo complessi e quindi soggetti a overfitting.

Caso del modello lineare

Nel caso della regressione lineare con errori gaussiani indipendenti e varianza costante, la stima per massima verosimiglianza coincide con quella per minimi quadrati ordinari (OLS). In questo contesto:

$$-2 \cdot \text{log-likelihood}(\hat{\beta}, \hat{\sigma}^2) = n \log(MSE_{Tr}),$$

dove MSE_{Tr} è l'errore medio quadratico sul training set.

Sostituendo, otteniamo:

$$AIC = n \log(MSE_{Tr}) + 2p.$$

Interpretazione

Il primo termine premia un buon adattamento (valori piccoli di MSE_{Tr}), mentre il secondo penalizza la complessità (modelli con molti parametri). Come nel caso del C_p di Mallows, l'AIC cerca un equilibrio tra bias e varianza, scegliendo il modello che minimizza la stima dell'errore di previsione.

Relazione con C_p

Nel caso dei modelli lineari stimati con OLS e varianza omoschedastica, i criteri C_p e AIC sono proporzionali:

$$C_p \propto AIC.$$

Questo significa che il modello che minimizza C_p minimizza anche l'AIC. Entrambi esprimono la stessa idea: il training error da solo non basta — serve una penalità proporzionale al numero di parametri stimati per evitare overfitting e ottenere buone previsioni su nuovi dati.

2.2.2 BIC

Il **Bayesian Information Criterion (BIC)** nasce da una prospettiva bayesiana, ma nella pratica assume una forma molto simile ai criteri C_p e AIC. Anche il BIC bilancia la qualità dell'adattamento con la complessità del modello, ma utilizza una penalità più severa per i modelli con molti parametri.

Definizione

Per un modello lineare con p predittori e n osservazioni, il BIC è definito come:

$$BIC = -2 \cdot \text{log-likelihood}(\hat{\beta}, \hat{\sigma}^2) + \log(n)\, p.$$

Nel caso della regressione lineare con errori gaussiani, sappiamo che:

$$-2 \cdot \text{log-likelihood}(\hat{\beta}, \hat{\sigma}^2) = n \log(MSE_{Tr}),$$

quindi:

$$BIC = n \log(MSE_{Tr}) + \log(n)\, p.$$

Interpretazione

Il BIC, come l'AIC, misura la bontà di adattamento del modello ai dati penalizzando la complessità. Tuttavia, la penalità è diversa:

$$\text{AIC: penalità } = 2p, \qquad \text{BIC: penalità } = \log(n)p.$$

Poiché per ogni $n > 7$ vale $\log(n) > 2$, la penalità del BIC cresce più rapidamente con il numero di parametri. Questo significa che il BIC tende a preferire modelli più **parsimoniosi**, ossia con meno variabili, rispetto all'AIC.

Uso pratico

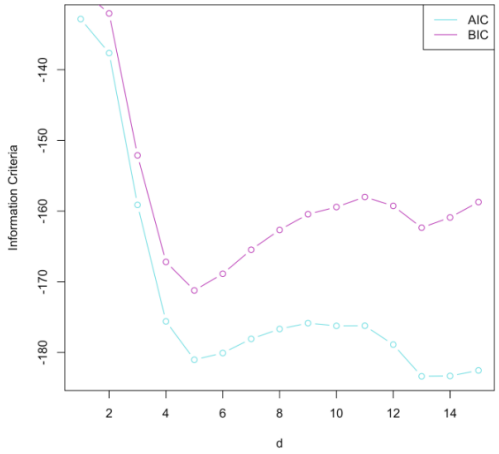
Il modello ottimale è quello che minimizza il valore del criterio:

$$\boxed{\text{Modello ottimale:} \quad \min(BIC).}$$

Un BIC più basso indica un compromesso migliore tra adattamento ai dati e semplicità del modello. In genere, se l'obiettivo è la previsione, l'AIC tende a scegliere modelli leggermente più complessi; se invece si cerca interpretabilità e parsimonia, il BIC è preferibile.

Esempio pratico (R)

```
AICs <- unlist(lapply(fits, AIC))
BICs <- unlist(lapply(fits, BIC))

plot(ds, AICs, type = "b", col = 5, ylab = "Information Criteria", xlab = "d")
lines(ds, BICs, type = "b", col = 6)
legend("topright", c("AIC", "BIC"), col = c(5, 6), lty = 1)

ps[which.min(AICs)]
## [1] 14

ps[which.min(BICs)]
## [1] 6
```



In questo esempio:

- L'AIC suggerisce un modello con $p = 14$ (più complesso);
- Il BIC suggerisce un modello con $p = 6$ (più semplice).

Questo comportamento è tipico: l'AIC tende a sottostimare leggermente la complessità necessaria, mentre il BIC — con una penalità più forte — privilegia modelli più compatti e interpretativi.

2.3 Random-X setting

Finora abbiamo considerato l'impostazione **Fixed-X**, in cui i valori dei predittori sono trattati come fissi e non casuali. Nella maggior parte delle applicazioni moderne di modellazione predittiva (ad esempio in machine learning), è più realistico adottare la prospettiva **Random-X**, dove anche i predittori X sono variabili casuali.

Assunzioni del modello Random-X:

- La risposta Y e i predittori casuali $\mathbf{X} = (X_1, \ldots, X_p)^\top$ hanno una distribuzione congiunta sconosciuta.
- Condizionatamente a $X = x$, il modello assume la forma:

$$Y = f(x) + \varepsilon,$$

  dove:

  - $f(x) = E(Y \mid X = x)$ è la **funzione di regressione vera**;
  - ε è un termine di errore indipendente da X, con:

$$E(\varepsilon) = 0, \qquad \mathrm{Var}(\varepsilon) = \sigma^2.$$

- L'ipotesi di omoschedasticità implica che la varianza condizionata è costante:

$$\mathrm{Var}(Y \mid X = x) = \sigma^2.$$

Campione di training e di test

- **Training set:** n osservazioni indipendenti e identicamente distribuite (i.i.d.):

$$(x_1, y_1), \ldots, (x_n, y_n) \text{ i.i.d. da } (X, Y).$$

- **Test set:** m nuove osservazioni i.i.d. dalla stessa distribuzione:

$$(x_1^*, y_1^*), \ldots, (x_m^*, y_m^*) \text{ i.i.d. da } (X, Y).$$

Errore di previsione Random-X

L'errore di previsione in questo contesto si definisce come:

$$\mathrm{Err}_R = E(MSE_{Te}) = E\left[\frac{1}{m}\sum_{i=1}^{m}(Y_i^* - \hat{f}(X_i^*))^2\right].$$

Poiché le osservazioni del test set sono i.i.d., per simmetria si può scrivere:

$$\mathrm{Err}_R = E\Big[(Y_1^* - \hat{f}(X_1^*))^2\Big].$$

Questa aspettativa è calcolata rispetto sia al campione di training $(X_1, Y_1), \ldots, (X_n, Y_n)$ sia al nuovo punto di test (X_1^*, Y_1^*).

Collegamento con la teoria delle decisioni statistiche

In questa prospettiva, (X, Y) segue una distribuzione congiunta sconosciuta. Cerchiamo una funzione $f(X)$ che permetta di prevedere Y minimizzando la perdita quadratica media:

$$(Y - f(X))^2.$$

Il problema può essere formulato come:

$$f = \arg\min_g E_{X,Y}\big[(Y - g(X))^2\big].$$

Soluzione del problema di minimizzazione

Condizionando su $X = x$ e applicando la legge dell'aspettativa iterata:

$$E_{X,Y}\big[(Y - g(X))^2\big] = E_X\left[E_{Y|X=x}\big[(Y - g(x))^2 \mid X = x\big]\right].$$

Poiché l'aspettativa esterna non influisce sulla minimizzazione punto per punto, basta minimizzare il termine interno rispetto a $g(x)$:

$$f(x) = \arg\min_c E_{Y|X=x}\big[(Y - c)^2 \mid X = x\big].$$

Dimostrazione

Espandendo il termine all'interno dell'aspettativa:

$$E_{Y|X=x}\big[(Y - c)^2\big] = E_{Y|X=x}\big[(Y - E(Y|X = x) + E(Y|X = x) - c)^2\big].$$

Sviluppando:

$$E_{Y|X=x}\big[(Y - c)^2\big] = \mathrm{Var}(Y|X = x) + \big[E(Y|X = x) - c\big]^2.$$

Poiché la varianza non dipende da c, il minimo si ottiene per:

$$\boxed{f(x) = E(Y|X = x)}.$$

Interpretazione

La migliore previsione di Y per un dato $X = x$, nel senso del **quadrato medio**, è la media condizionata $E(Y|X = x)$. Questo risultato è fondamentale in tutta la teoria della regressione: la funzione di regressione $f(x)$ rappresenta il valore atteso di Y dato $X = x$, ovvero la previsione "ottimale" in media.

2.3.1 Esempio Gaussiano

Consideriamo ora un esempio concreto in cui la relazione tra X e Y segue una distribuzione normale congiunta. Supponiamo che:

$$\begin{pmatrix} Y \\ X \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} \mu_y \\ \mu_x \end{pmatrix}, \begin{pmatrix} \sigma_y^2 & \rho\,\sigma_x\sigma_y \\ \rho\,\sigma_x\sigma_y & \sigma_x^2 \end{pmatrix} \right).$$

In questo caso, la distribuzione condizionata di Y dato $X = x$ è anch'essa normale:

$$(Y \mid X = x) \sim \mathcal{N}\left(\mu_y + \rho\,\frac{\sigma_y}{\sigma_x}(x - \mu_x),\ \sigma_y^2(1-\rho^2) \right).$$

Interpretazione

La media condizionata di Y dato $X = x$ è una funzione lineare di x:

$$f(x) = E(Y \mid X = x) = \underbrace{\left(\mu_y - \rho\,\tfrac{\sigma_y}{\sigma_x}\mu_x\right)}_{\alpha} + \underbrace{\left(\rho\,\tfrac{\sigma_y}{\sigma_x}\right)}_{\beta} x = \alpha + \beta x.$$

Pertanto, il modello lineare che collega Y e X è una conseguenza diretta della struttura gaussiana congiunta. La pendenza β riflette la correlazione ρ tra X e Y: quanto più alta è ρ, tanto più forte sarà la relazione lineare tra le due variabili.

Generazione dei dati di training

Per simulare un campione di training con n osservazioni:

1. Generiamo x_i come realizzazione da $X_i \sim \mathcal{N}(\mu_x, \sigma_x^2)$.
2. Generiamo y_i condizionatamente a x_i secondo:

   $$Y_i \mid X_i = x_i = f(x_i) + \varepsilon_i,$$

   dove:

   $$f(x_i) = \alpha + \beta x_i, \qquad \varepsilon_i \sim \mathcal{N}(0, \sigma^2),$$

   e

   $$\sigma^2 = \sigma_y^2(1-\rho^2).$$

Generazione dei dati di test

Analogamente, per un test set di m osservazioni indipendenti:

1. Si generano x_i^* da $X_i^* \sim \mathcal{N}(\mu_x, \sigma_x^2)$;
2. Si generano y_i^* condizionatamente a x_i^* secondo:

   $$Y_i^* \mid X_i^* = x_i^* = f(x_i^*) + \varepsilon_i^*,$$

   con

   $$f(x_i^*) = \alpha + \beta x_i^*, \qquad \varepsilon_i^* \sim \mathcal{N}(0, \sigma^2).$$

Riepilogo delle quantità fondamentali:

$$\alpha = \mu_y - \rho \frac{\sigma_y}{\sigma_x}\mu_x,$$

$$\beta = \rho \frac{\sigma_y}{\sigma_x},$$

$$\sigma^2 = \sigma_y^2(1-\rho^2).$$

Interpretazione finale

In un modello lineare generato da una coppia di variabili gaussiane, il coefficiente di regressione β quantifica quanto la variazione di Y è spiegata da quella di X; il termine di errore ε_i rappresenta la parte di Y non spiegata da X, con varianza proporzionale a $1-\rho^2$. Quando $\rho = 1$, la relazione è perfettamente lineare e non vi è rumore residuo ($\sigma^2 = 0$).

2.3.2 Random-X optimism per modelli lineari

Nel caso **Random-X**, anche i predittori X sono variabili casuali. Consideriamo il modello lineare:

$$f(x) = x^\top \beta,$$

dove i predittori seguono una distribuzione normale multivariata:

$$X = (X_1, \ldots, X_p)^\top \sim \mathcal{N}(0, \Sigma),$$

con Σ matrice di covarianza invertibile e $p < n-1$.

Ipotesi di base

Queste condizioni sono soddisfatte, ad esempio, se (X, Y) ha una distribuzione normale congiunta. In tal caso, il modello lineare rappresenta la forma corretta della relazione tra X e Y.

Ottimismo nel caso Random-X

Nel contesto Fixed-X, abbiamo già visto che:

$$\mathrm{Opt}_F = \frac{2\sigma^2 p}{n}.$$

Tuttavia, quando X è casuale (Random-X), l'ottimismo medio aumenta a causa della variabilità addizionale introdotta dalla stima di X stesso. Il risultato teorico (Efron, 2004) mostra che:

$$\boxed{\mathrm{Opt}_R = \mathrm{Opt}_F + \frac{\sigma^2 p}{n}\left(\frac{p+1}{n-p-1}\right).}$$

Interpretazione intuitiva

- Il primo termine, $\text{Opt}_F = \dfrac{2\sigma^2 p}{n}$, misura l'ottimismo quando i predittori sono fissi (ossia il "classico" caso OLS).
- Il termine aggiuntivo $\dfrac{\sigma^2 p}{n}\left(\dfrac{p+1}{n-p-1}\right)$ cattura l'**incremento di varianza dovuto alla casualità di** X: più il modello è complesso (p grande) o più piccolo è il campione (n vicino a p), maggiore sarà questo contributo.

Effetto pratico

Nel regime Random-X, l'errore di previsione stimato tramite training data tende a essere ancora più ottimistico rispetto al Fixed-X, poiché il modello non solo si adatta al rumore di Y, ma anche alle fluttuazioni casuali dei predittori X. Questo giustifica l'uso di tecniche più robuste per la stima dell'errore predittivo, come la **cross-validation**, presentata nelle sezioni successive.

2.3.3 Random-X Mallows' C_p

Nel caso Random-X, l'errore di previsione atteso è dato da:

$$\text{Err}_R = E(MSE_{Te}) + \text{Opt}_R.$$

Come visto nella sezione precedente, nel modello lineare vale:

$$\text{Opt}_R = \text{Opt}_F + \frac{\sigma^2 p}{n}\left(\frac{p+1}{n-p-1}\right), \qquad \text{con} \quad \text{Opt}_F = \frac{2\sigma^2 p}{n}.$$

Sostituendo otteniamo quindi:

$$\text{Err}_R = E(MSE_{Te}) + \frac{\sigma^2 p}{n}\left(2 + \frac{p+1}{n-p-1}\right).$$

Stima empirica dell'errore di previsione

Poiché l'errore di test non è osservabile, lo stimiamo tramite:

$$\boxed{\widehat{\text{Err}}_R = MSE_{Tr} + \frac{\sigma^2 p}{n}\left(2 + \frac{p+1}{n-p-1}\right).}$$

Il primo termine, MSE_{Tr}, rappresenta l'errore di training medio; il secondo è la **correzione per ottimismo**, che aumenta rispetto al caso Fixed-X perché tiene conto della variabilità dei predittori casuali.

Versione Random-X di Mallows' C_p

Se non conosciamo la varianza del rumore σ^2, possiamo sostituirla con la stima usuale della regressione lineare:

$$\hat{\sigma}^2 = \frac{RSS}{n-p},$$

dove $RSS = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = n\,MSE_{Tr}$ è la somma dei quadrati dei residui.

Sostituendo nella formula precedente otteniamo la versione Random-X del C_p di Mallows:

$$\boxed{RC_p = C_p + \frac{\hat{\sigma}^2 p}{n}\left(\frac{p+1}{n-p-1}\right),}$$

dove ricordiamo che nel caso Fixed-X:

$$C_p = MSE_{Tr} + \frac{2\hat{\sigma}^2 p}{n}.$$

Esplicitando tutto in funzione dell'RSS, si ottiene:

$$RC_p = \frac{RSS}{n}\left[1 + \frac{p}{n-p}\left(2 + \frac{p+1}{n-p-1}\right)\right] = \frac{RSS(n-1)}{(n-p)(n-p-1)}.$$

Interpretazione

- Il termine aggiuntivo rispetto a C_p corregge l'ottimismo per il caso Random-X, dove la variabilità dei predittori amplifica la differenza tra errore di training e di test.
- Quando $n \gg p$, il termine aggiuntivo è piccolo, quindi $RC_p \approx C_p$.
- Quando invece p è grande rispetto a n, la penalizzazione cresce rapidamente, spingendo la selezione verso modelli più semplici (meno variabili).

Chapter 3

Cross-validation

Quando vogliamo scegliere il modello "migliore", l'obiettivo è trovare quello con il **più basso errore di test**. Esistono due strategie principali per farlo:

1. **Stima indiretta**: correggere l'errore di training aggiungendo una penalità per l'overfitting (come avviene per C_p, AIC o BIC).
2. **Stima diretta**: stimare l'errore di test in modo empirico, tramite *validation set* o *cross-validation*.

I criteri informativi come AIC o BIC misurano un compromesso tra qualità dell'adattamento e complessità del modello. La cross-validation, invece, non fa ipotesi specifiche sul tipo di modello o sulla distribuzione dei dati: è un approccio **non parametrico e generale**.

Problema di fondo: se usiamo gli stessi dati per allenare e valutare il modello, otteniamo un risultato troppo ottimistico. Il modello ha già "visto" quei dati e li riproduce troppo bene: questo è l'**overfitting**.

Idea della cross-validation (CV): per stimare realisticamente l'errore di test, simuliamo la presenza di nuovi dati dividendo il dataset in due parti:

$$\text{training set} \ + \ \text{validation set}.$$

Il training set serve per addestrare il modello, mentre il validation set viene usato come se fossero dati "mai visti", per valutare la sua capacità di generalizzare.

Formalmente, la cross-validation mira a stimare l'errore medio di previsione:

$$\widehat{\text{Err}} = E(\widehat{MSE}_{Te}).$$

In pratica, si calcola l'errore di previsione su vari sottogruppi dei dati e poi si fa la media.

Perché è utile:

- funziona anche con **pochi dati**, suddividendoli in modo efficiente;
- non richiede di conoscere σ^2 o i gradi di libertà del modello;
- può essere applicata a qualsiasi algoritmo di previsione (lineare o non lineare);

- fornisce una **stima diretta** dell'errore di test, più realistica rispetto ai criteri AIC, BIC o C_p.

In sintesi:

- se divido i dati una sola volta in train e validation, ottengo una **stima di validazione**;
- se ripeto la divisione più volte (con diverse partizioni) e faccio la media degli errori, ottengo una **stima di cross-validation**.

In formule, se indichiamo con $MSE_{Te}^{(b)}$ l'errore di test nella b-esima suddivisione dei dati, la cross-validation media è:

$$E(\widehat{MSE}_{Te}) \;=\; \frac{1}{B}\sum_{b=1}^{B} MSE_{Te}^{(b)}.$$

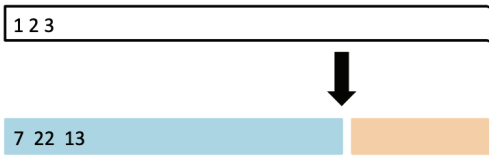
Interpretazione pratica: ogni volta che "togliamo" una parte di dati, stiamo chiedendo: *"quanto bene il mio modello predice osservazioni che non ha mai visto?"* Questo ci fornisce una misura empirica della capacità di generalizzazione, molto più affidabile rispetto a quella stimata sui dati di training.

3.1 Validation set approach

Un modo semplice per stimare l'errore di test consiste nel dividere casualmente le n osservazioni in due parti: un **training set** e un **validation set** (detto anche *hold-out set*).

L'idea è la seguente:

- si usa il training set per **stimare** il modello;
- si usa il validation set per **valutare** quanto bene il modello predice osservazioni che non ha mai visto.



Formalmente, indichiamo con:

$$T \subset \{1, \ldots, n\}$$

l'insieme degli indici delle osservazioni usate per il training, e con:

$$V = \{1, \ldots, n\} \setminus T$$

l'insieme degli indici delle osservazioni usate per la validazione.

Il modello stimato sui dati di training si indica con:

$$\hat{f}^{-V},$$

dove il simbolo "$-V$" ricorda che il modello è stato costruito **escludendo** le osservazioni nel validation set V.

L'errore di validazione fornisce una stima dell'errore di test atteso:

$$\widehat{\mathrm{Err}} = \frac{1}{|V|} \sum_{i \in V} \big(y_i - \hat{f}^{-V}(x_i)\big)^2,$$

dove:

- $|V|$ è il numero di osservazioni nel validation set;
- y_i è il valore reale osservato;
- $\hat{f}^{-V}(x_i)$ è il valore predetto dal modello per l'osservazione i.

Vantaggi:

- concettualmente semplice e facile da implementare;
- permette di ottenere una stima diretta della capacità di generalizzazione del modello.

Limiti:

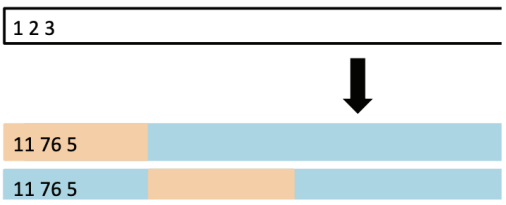
- si riduce la quantità di dati usati per stimare il modello (meno informazioni per il training);
- se il campione totale n non è grande, la stima dell'errore può essere **molto variabile**, cioè dipendere fortemente da come i dati sono stati divisi in training e validation set.

In sintesi, questo approccio funziona bene quando il dataset è molto grande, ma con campioni piccoli può produrre stime instabili dell'errore di previsione.

3.2 K-fold cross-validation

L'idea di base è ridurre l'arbitrarietà della singola divisione **train/validation** suddividendo i dati in K parti (dette **fold**) di dimensione (quasi) uguale:

$$V_1, V_2, \ldots, V_K \quad \text{con} \quad V_k \subset \{1, \ldots, n\}, \quad \bigcup_{k=1}^{K} V_k = \{1, \ldots, n\}, \quad V_j \cap V_k = \ (j \neq k).$$



Per ogni $k = 1, \ldots, K$ si allena il modello **escludendo** il fold V_k (quindi usando tutte le osservazioni con indici $i \notin V_k$), e si valuta la prestazione **solo** sulle osservazioni nel fold lasciato fuori V_k:

$$\frac{1}{|V_k|} \sum_{i \in V_k} \big(y_i - \hat{f}^{-V_k}(x_i)\big)^2,$$

dove $\hat{f}^{-V_k}$ è il modello stimato senza usare i dati in V_k. Infine, si fa la media **tra i fold** per ottenere una stima dell'errore di test atteso:

$$\widehat{\mathrm{Err}} \;=\; \frac{1}{K}\sum_{k=1}^{K}\left[\frac{1}{|V_k|}\sum_{i\in V_k}\left(y_i-\hat{f}^{-V_k}(x_i)\right)^2\right].$$

Procedimento:

1. Mischia gli indici $1,\ldots,n$ e suddividili in K blocchi $V_1,\ldots,V_K$.
2. Per ogni k: stima il modello sui dati con indici $\{1,\ldots,n\}\setminus V_k$ e calcola l'MSE sul fold lasciato fuori V_k.
3. Fai la media dei K MSE ottenuti: è la stima $\widehat{\mathrm{Err}}$ per quel modello.
4. Ripeti la procedura per ogni modello/capacità (es. grado del polinomio d) e scegli quello con $\widehat{\mathrm{Err}}$ più basso.

Perché funziona meglio di un singolo split? Ogni osservazione fa da **test** una volta e da **train** $K-1$ volte. Questo riduce la dipendenza della stima da una singola divisione casuale e usa i dati in modo più efficiente.

Scelte comuni: $K=5$ o $K=10$ sono standard. Con $K=n$ si ottiene la **LOOCV** (leave-one-out): bias molto basso ma varianza più alta e costo computazionale maggiore. Con K piccolo (es. 5 o 10) il bias aumenta leggermente, ma la varianza si riduce e il costo è molto più contenuto.

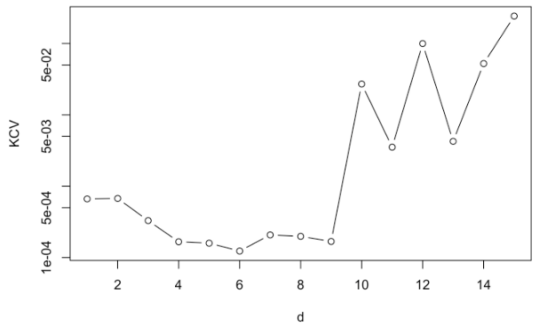
Nota pratica: per problemi di classificazione si usa spesso la **stratified K-fold** (le classi sono bilanciate in ogni fold). Per serie temporali non si mischiano i dati: si usano schemi **bloccati** (forward chaining).

Esempio R (K-fold per un polinomio di grado $d=3$)

```
# By hand
rm(list=ls())
library(readr)
df <- read_table("http://azzalini.stat.unipd.it/Book-DM/yesterday.dat")[-31,]
train <- data.frame(x = df$x, y = df$y.yesterday)

n <- nrow(train)
d <- 3
K <- 5
set.seed(123)

# Crea i K fold (etichette 1..K mescolate)
folds <- sample(rep(1:K, length = n))

# Vettore per memorizzare l'MSE di ciascun fold
KCV <- numeric(K)

# Loop sui fold
for (k in 1:K){
  # Allena escludendo il fold k
```

```
  fit   <- lm(y ~ poly(x, degree = d), train, subset = which(folds != k))

  # Predici sulle osservazioni nel fold k
  x.out <- train$x[which(folds == k)]
  yhat  <- predict(fit, newdata = list(x = x.out))
  y.out <- train$y[which(folds == k)]

  # Errore sul fold k
  KCV[k] <- mean( (y.out - yhat)^2 )
}

# Stima K-fold CV: media sugli MSE dei K fold
mean(KCV)
# [1] 0.0003160667
```

Esempio R con funzione `boot::cv.glm` su gradi $d = 1, \ldots, 15$



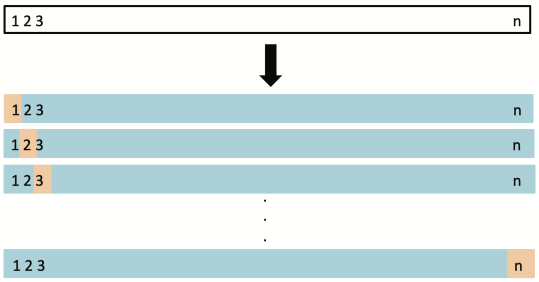

```
# By function
library(boot)
ds <- 1:15
ps <- ds + 1
K  <- 5
set.seed(123)

KCV <- sapply(ds, function(d)
  cv.glm(
    data = train,
    glmfit = glm(y ~ poly(x, degree = d), train, family = gaussian),
    K = K
  )$delta[1]
)

plot(ds, KCV, type = "b", log = "y", xlab = "d", ylab = sprintf("%d-fold CV MSE", K))

ds[which.min(KCV)]
# [1] 6
```

Come leggere i risultati:

- il grafico (d vs $\widehat{\text{Err}}$) mostra l'andamento dell'errore stimato al variare della complessità;
- scegli il grado d (o, in generale, il modello) che **minimizza** la curva K-fold CV;
- rispetto a C_p, AIC, BIC, la K-fold CV **non** richiede stime di σ^2 né conteggi "semplici" dei gradi di libertà, ed è applicabile a molti algoritmi diversi.

3.3 Leave-One-Out Cross-Validation (LOOCV)

La **Leave-One-Out Cross-Validation** (LOOCV) è un caso particolare della K-fold cross-validation, in cui $K = n$, cioè ogni osservazione viene esclusa una sola volta e utilizzata come test.



In questo modo, ogni punto del dataset funge una volta da **dato di validazione** e $n-1$ volte da **dato di training**.

Formalmente, per ogni osservazione $i = 1, \ldots, n$:

1. Escludi l'osservazione (x_i, y_i) dal training set.
2. Stima il modello sui rimanenti $n-1$ dati, ottenendo $\hat{f}^{-i}$.
3. Calcola l'errore quadratico sul punto lasciato fuori:

$$(y_i - \hat{f}^{-i}(x_i))^2.$$

La stima complessiva dell'errore di test è la media di tutti questi errori:

$$\widehat{\text{Err}} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}^{-i}(x_i))^2.$$

Interpretazione intuitiva: ogni volta che lasciamo fuori un punto, testiamo la capacità del modello di generalizzare a un nuovo dato. Ripetendo l'operazione per tutti i punti, otteniamo una misura molto stabile dell'errore di previsione.

Vantaggi:

- Usa praticamente tutto il dataset per stimare il modello ($n-1$ osservazioni ogni volta).
- Non dipende da una particolare divisione casuale dei dati.

- Fornisce una stima dell'errore quasi non distorta.

Svantaggi:

- È computazionalmente costosa: servono n riaddestramenti del modello.
- La varianza della stima può essere elevata, perché i training set sono molto simili tra loro (differiscono di una sola osservazione).

Esempio in R (implementazione manuale):

```
# LOOCV "by hand"
oneout <- numeric(n)
for (i in 1:n){
  fit_i  <- lm(y ~ poly(x, degree = d), data = train[-i, ])
  yhat_i <- predict(fit_i, newdata = data.frame(x = train$x[i]))
  oneout[i] <- (train$y[i] - yhat_i)^2
}
mean(oneout)
# [1] 0.0003439458
```

Shortcut per modelli lineari: Per i modelli lineari si può evitare di ricalcolare il modello n volte usando la **matrice cappello** (**hat matrix**):

$$H = X(X^\top X)^{-1}X^\top,$$

dove X è la matrice di progetto ($n \times p$). Il valore diagonale h_{ii} misura l'influenza del punto i sul proprio valore predetto.
Con questa scorciatoia, l'errore LOOCV può essere scritto come:

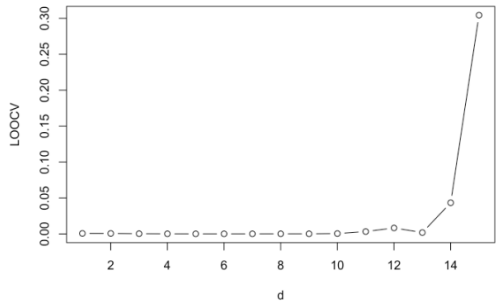
$$\widehat{\mathrm{Err}} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i - \hat{f}(x_i)}{1 - h_{ii}}\right)^2.$$

Esempio R con matrice:

```
fit <- lm(y ~ poly(x, d), train)
X <- model.matrix(fit)
H <- X %*% solve(t(X) %*% X) %*% t(X)
mean(((train$y - predict(fit)) / (1 - diag(H)))^2)
# [1] 0.0003439458
```

Esempio R con funzione:

```
LOOCV <- sapply(ds, function(d)
  cv.glm(train, glm(y ~ poly(x, degree = d),
  train, family = gaussian))$delta[1]
)
plot(ds, LOOCV, type = "b", xlab = "d")
ds[which.min(LOOCV)]
# [1] 6
```



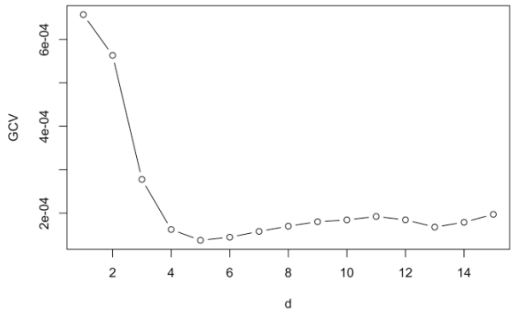
Generalized Cross-Validation (GCV): Per ridurre il costo computazionale, si approssima ogni h_{ii} con la media dei valori diagonali di H:

$$\frac{1}{n}\sum_{i=1}^{n} h_{ii} = \frac{p}{n},$$

ottenendo così:

$$\widehat{\mathrm{Err}}_{\mathrm{GCV}} = \frac{MSE_{Tr}}{(1-p/n)^2}.$$

Esempio R per GCV:




```
fun <- function(d)
  if (d == 0) lm(y ~ 1, train)
  else lm(y ~ poly(x, degree = d), train)

fits <- lapply(ds, fun)
MSEs.tr <- unlist(lapply(fits, deviance)) / n
GCV <- MSEs.tr / (1 - ps/n)^2
plot(ds, GCV, type = "b", xlab = "d")
ds[which.min(GCV)]
# [1] 5
```

In sintesi:

- LOOCV è il caso limite della K-fold CV con $K = n$;
- fornisce una stima quasi non distorta dell'errore di test;
- il GCV ne è una versione approssimata e più veloce, utile soprattutto nei modelli lineari.

3.4 Bias–Variance Trade-off in Cross-Validation

La scelta di K nella K-fold cross-validation è una questione di equilibrio tra **bias** e **varianza**. Nella pratica, i valori più comuni sono $K = 5$ o $K = 10$, che offrono un buon compromesso. Tuttavia, non esiste una "regola universale", poiché le prestazioni della CV dipendono fortemente dal contesto e dal tipo di modello.

Bias

Il bias misura quanto la stima dell'errore di test $\widehat{\text{Err}}$ tende a sottostimare o sovrastimare l'errore vero.

- Quando K è piccolo (ad esempio $K = 5$ o 10), ogni modello viene allenato su una porzione ridotta dei dati — rispettivamente 4/5 o 9/10 delle osservazioni. ⇒ Questo porta a una stima dell'errore di test **più alta del vero valore**, quindi un **bias positivo (upward bias)**.
- Con la LOOCV ($K = n$), invece, ogni modello è allenato su $n-1$ osservazioni, quindi quasi tutto il dataset. ⇒ Il bias è **molto basso**, perché il modello addestrato è quasi identico a quello che si otterrebbe usando tutti i dati.

$$\text{Bias}_{K=5,10} > \text{Bias}_{LOOCV}$$

Variance

La varianza misura quanto varia la stima dell'errore di test $\widehat{\text{Err}}$ se ripetiamo la procedura su diversi campioni o diverse divisioni dei dati.

- Nella LOOCV, ogni stima di errore parziale $(y_i - \hat{f}^{-i}(x_i))^2$ è calcolata da modelli che differiscono per una sola osservazione nel training set. Le n stime parziali sono quindi **altamente correlate**. ⇒ L'errore medio risultante tende ad avere **alta varianza**.
- Con $K = 5$ o $K = 10$, i modelli nei diversi fold condividono meno osservazioni e sono quindi meno correlati. ⇒ La varianza della stima complessiva è **più bassa**.

$$\text{Var}_{LOOCV} > \text{Var}_{K=5,10}$$

Questo fenomeno può essere interpretato ricordando che, per due quantità casuali correlate A e B:

$$\text{Var}(A + B) = \text{Var}(A) + \text{Var}(B) + 2\,\text{Cov}(A, B).$$

Quando la covarianza è alta (cioè A e B sono molto simili), la varianza della loro somma aumenta. Perciò, mediare errori molto correlati — come nella LOOCV — produce una stima più variabile.

Conclusione intuitiva

Metodo	Bias	Varianza
LOOCV ($K = n$)	molto basso	alto
5-fold CV	medio-alto	basso
10-fold CV	medio-basso	medio

In generale, la scelta di K determina un **compromesso**:

$$\text{Bias} \downarrow \Rightarrow \text{Varianza} \uparrow, \quad \text{Bias} \uparrow \Rightarrow \text{Varianza} \downarrow .$$

Per questo motivo, $K = 5$ o $K = 10$ rappresentano nella pratica un buon equilibrio tra accuratezza e stabilità della stima.

Chapter 4

K-Nearest Neighbours

4.1 Metodi non parametrici

I **metodi non parametrici** sono metodi di stima della funzione di regressione f che **non impongono una forma specifica a priori** per f. Questo significa che **non** partiamo dicendo, per esempio, "$f(x)$ è lineare" oppure "$f(x)$ è un polinomio di grado 3" oppure "$f(x)$ è esponenziale". Al contrario, lasciamo che siano i dati stessi a determinare la forma di f.

Idea di base

Nei metodi parametrici (come la regressione lineare o polinomiale), assumiamo che:

$$f(x) \approx \beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_d x^d,$$

cioè imponiamo a f una struttura fissata (ad esempio polinomiale di grado d). In questo caso, stimare f significa trovare un **piccolo numero di parametri** $(\beta_0, \ldots, \beta_d)$.

Nei metodi **non parametrici**, invece:

- non imponiamo che f sia una retta, né un polinomio, né una forma chiusa nota;
- permettiamo a f di avere una forma molto flessibile e potenzialmente irregolare;
- la stima di f viene costruita in modo più "locale", cioè guardando come si comportano i dati vicino a ciascun punto x.

In altre parole, **non diciamo prima com'è la curva**: lasciamo che sia il dataset a "disegnarla".

Vantaggio principale

Il vantaggio più importante dei metodi non parametrici è la **flessibilità**.

Poiché non costringiamo f ad avere una forma rigida (ad esempio lineare), questi metodi possono adattarsi a relazioni complesse tra la variabile esplicativa X e la risposta Y. Per esempio, se la vera relazione tra X e Y è molto non lineare, con curve, pieghe, cambi di pendenza, effetti locali, allora un metodo non parametrico ha molte più possibilità di seguire questa forma reale.

Possiamo riassumere così:

$$\text{Metodo non parametrico} \quad \Longrightarrow \quad \text{f può avere molte forme diverse.}$$

Quindi:

- Se la forma vera di f è complicata, un metodo non parametrico può approssimarla meglio di un modello lineare o di un polinomio di basso grado.
- Questo riduce il **bias**: non stiamo forzando una forma sbagliata.

Svantaggio principale

Il principale svantaggio è che, essendo così flessibili, i metodi non parametrici hanno bisogno di **molti dati**.

Per capire perché, ragioniamo così:

- Nei modelli parametrici stimiamo pochi numeri (ad esempio i coefficienti β in una regressione lineare). Con pochi parametri, anche un campione moderato è spesso sufficiente.
- Nei metodi non parametrici, invece, non riassumiamo f con pochi parametri globali. Questo vuol dire che **stimiamo f in modo "pezzo per pezzo" o "punto per punto"**, usando i dati vicini.

Quindi:

- per stimare bene $f(x_0)$ in un certo punto x_0, ci servono abbastanza osservazioni "vicine" a x_0;
- per stimare bene $f(x_1)$ in un altro punto x_1, ci servono abbastanza osservazioni "vicine" a x_1;
- e così via per tutti i punti del dominio.

Se i dati sono pochi o molto sparsi, questa informazione locale non è sufficiente, e la stima diventa molto rumorosa (alta varianza). In pratica:

$$\text{Metodo non parametrico} \quad \Longrightarrow \quad \text{serve un campione grande per essere stabile.}$$

Riassunto concettuale

- **Definizione:** un metodo non parametrico stima la relazione tra X e Y senza imporre una particolare forma analitica per f.
- **Vantaggio:** *alta flessibilità.* Può catturare relazioni complesse e non lineari, riducendo il bias dovuto a una forma sbagliata del modello.
- **Svantaggio:** *alta richiesta di dati.* Siccome non riassume f in pochi parametri, ma la costruisce in modo dettagliato lungo tutto il dominio, servono molte osservazioni per evitare alta varianza e instabilità.

Collegamento con il trade-off bias-varianza

Possiamo collegare i metodi non parametrici al solito schema bias-varianza:

- Un metodo molto flessibile (tipicamente non parametrico) ha:

$$\text{Bias basso} \quad \text{ma} \quad \text{Varianza alta.}$$

- Un metodo rigido (tipicamente parametrico semplice, come una retta) ha:

$$\text{Bias alto} \quad \text{ma} \quad \text{Varianza bassa.}$$

Questo spiega perché i metodi non parametrici funzionano bene quando abbiamo tanti dati: con tanti dati, la varianza si riduce (perché ogni zona del dominio è ben campionata), ma il bias rimane basso perché il metodo è flessibile.

In sintesi:

$$\text{Metodi non parametrici} \quad \Longrightarrow \quad \text{potenza espressiva alta, ma fame di dati.}$$

4.2 k-nearest neighbors (kNN)

Il metodo **k-nearest neighbors** (kNN) è uno dei metodi non parametrici più semplici e più usati per la regressione e la classificazione.

L'idea è: per prevedere il valore della risposta in un nuovo punto, guardo i "vicini" più simili nel training set e faccio una media dei loro valori osservati.

Come funziona kNN (regressione)

Supponiamo di voler fare una previsione nel punto x_1^* (cioè un nuovo input per cui non conosciamo ancora la risposta y).

1. Calcoliamo la distanza tra x_1^* e **tutti** i punti di training x_i.
2. Selezioniamo i k punti di training più vicini a x_1^*. Questo insieme di indici lo chiamiamo $\mathcal{N}_k(x_1^*)$ (il "vicinato" di x_1^*).
3. Prendiamo la media delle loro risposte osservate y_i.

Formalmente, la previsione kNN in x_1^* è:

$$\hat{f}(x_1^*) = \frac{1}{k} \sum_{i \in \mathcal{N}_k(x_1^*)} y_i.$$

Quindi: **per prevedere il valore in x_1^*, faccio la media dei k vicini più simili.**

Come misuriamo la distanza (distanza euclidea)

Per capire chi sono i "più vicini", dobbiamo definire cosa vuol dire "vicino". Di solito usiamo la **distanza euclidea** (norma ℓ_2):

$$\|x_i - x_1^*\|_2 = \sqrt{(x_i - x_1^*)^\top (x_i - x_1^*)}.$$

Più questa distanza è piccola, più x_i è simile a x_1^*. I k punti con distanza più piccola formano $\mathcal{N}_k(x_1^*)$.

Nota pratica importante

Se le variabili esplicative (i predittori) hanno scale diverse (esempio: una è "età" in anni, un'altra è "stipendio" in euro), allora la distanza euclidea è dominata dalla variabile con i numeri più grandi. Per evitare questo problema:

- di solito **centriamo** i predittori (togliamo la media),
- e **ridimensioniamo** (dividiamo per la deviazione standard).

Così tutte le variabili sono confrontabili e la distanza ha senso.

Il ruolo di k: complessità del modello

Il valore di k è un **parametro di scelta** (tuning parameter). Non viene "imparato" dal modello: lo scegli tu (tipicamente con la cross-validation).

- k **piccolo** (ad esempio $k = 1, 2, 3$):
  - la previsione usa pochissimi vicini;
  - il modello è molto flessibile e segue molto da vicino i dati osservati;
  - rischio: **overfitting** (alta varianza).
- k **grande** (ad esempio $k = 20, 30, \ldots$):
  - la previsione è una media di molti punti;
  - il modello è più liscio e meno flessibile;
  - rischio: **underfitting** (alto bias, cioè la stima è troppo "piatta").

Quindi k controlla il classico **bias-varianza trade-off**: piccolo $k \Rightarrow$ basso bias ma alta varianza, grande $k \Rightarrow$ alto bias ma bassa varianza.

Analisi teorica (Fixed-X)

Ora ragioniamo nel caso Fixed-X, cioè assumiamo che i punti $x_1, \ldots, x_n$ siano fissati e non casuali, e che le sole quantità casuali siano le risposte Y_i.

Ricordiamo che il modello è:

$$Y_i = f(x_i) + \varepsilon_i, \quad E[\varepsilon_i] = 0, \quad \mathrm{Var}(\varepsilon_i) = \sigma^2.$$

Noi stimiamo f con $\hat{f}$ usando kNN.

Bias della stima kNN

La stima kNN in un punto x_i è la media dei k vicini di x_i:

$$\hat{f}(x_i) = \frac{1}{k} \sum_{j \in \mathcal{N}_k(x_i)} Y_j.$$

Se prendiamo l'aspettativa rispetto al rumore (cioè immaginiamo di poter ripetere l'esperimento tante volte con stesso X ma nuovi errori),

$$E[\hat{f}(x_i)] = \frac{1}{k} \sum_{j \in \mathcal{N}_k(x_i)} E[Y_j] = \frac{1}{k} \sum_{j \in \mathcal{N}_k(x_i)} f(x_j).$$

Quindi il **bias** in x_i è:

$$E[\hat{f}(x_i)] - f(x_i) = \frac{1}{k} \sum_{j \in \mathcal{N}_k(x_i)} f(x_j) - f(x_i).$$

Interpretazione:

- Se tutti i vicini di x_i hanno valori di $f(x_j)$ molto simili a $f(x_i)$, allora la media dei vicini sarà vicina a $f(x_i)$ e il bias è piccolo.
- Se invece f cambia velocemente attorno a x_i (ad esempio f fa una curva ripida), allora i vicini possono avere valori molto diversi e la media può allontanarsi da $f(x_i)$, creando bias.
- Bias ↑ quando k è grande e si prendono vicini lontani (si fa una media troppo "larga").

Varianza della stima kNN

Dato che $\hat{f}(x_i)$ è una media di k osservazioni rumorose $Y_j = f(x_j) + \varepsilon_j$, la sua varianza (sotto assunzioni standard tipo errori indipendenti con stessa varianza σ^2) è:

$$\mathrm{Var}(\hat{f}(x_i)) = \mathrm{Var}\left(\frac{1}{k} \sum_{j \in \mathcal{N}_k(x_i)} Y_j \right) = \frac{1}{k^2} \sum_{j \in \mathcal{N}_k(x_i)} \mathrm{Var}(Y_j) = \frac{1}{k^2} \cdot k \cdot \sigma^2 = \frac{\sigma^2}{k}.$$

Quindi:

$$\boxed{\mathrm{Var}(\hat{f}(x_i)) = \frac{\sigma^2}{k}.}$$

Interpretazione:

- Se k è piccolo, stiamo mediando pochissime osservazioni ⇒ la varianza è grande.
- Se k è grande, stiamo mediando tante osservazioni ⇒ la varianza scende (perché la media "stabilizza" il rumore).

Questa è la parte "varianza" del trade-off.

Errore di previsione atteso

Nel Fixed-X, l'errore di previsione medio (prediction error atteso sul test set negli stessi punti x_i) si può scrivere come:

$$\mathrm{Err}_F = \sigma^2 + \frac{1}{n} \sum_{i=1}^{n} \underbrace{\left(\frac{1}{k} \sum_{j \in \mathcal{N}_k(x_i)} f(x_j) - f(x_i) \right)}_{\text{Bias in } x_i}^2 + \frac{\sigma^2}{k}.$$

Vediamolo pezzo per pezzo:

- σ^2 è l'errore **irriducibile** (il rumore intrinseco nei dati: anche se conoscessimo f perfettamente, rimarrebbe).
- Il termine medio sui punti $i = 1, \ldots, n$ è il **bias** al quadrato: quanto la media dei vicini si discosta dalla vera $f(x_i)$.

- $\frac{\sigma^2}{k}$ è la **varianza** della stima kNN.

Questa formula è la versione "bias^2 + var + rumore" applicata al kNN:

$$\mathrm{Err}_F = \underbrace{\sigma^2}_{\text{irriducibile}} + \underbrace{\mathrm{Bias}^2}_{\text{dipende da quanto è liscio } f \text{ e da } k} + \underbrace{\mathrm{Var}}_{\sigma^2/k,\ \text{dipende da } k}.$$

Messaggi importanti da ricordare

- kNN è un metodo **non parametrico**: non impone una forma a f. Impara la forma di f direttamente dai dati vicini al punto in cui vuoi prevedere.
- La scelta di k è cruciale:
  - k piccolo: modello molto flessibile $\Rightarrow$ basso bias, alta varianza;
  - k grande: modello più rigido $\Rightarrow$ alto bias, bassa varianza.
- In pratica, k si sceglie di solito con **cross-validation**: provi diversi valori di k, calcoli l'errore di validazione, e scegli quello con l'errore più basso.
- kNN soffre in alta dimensione (troppe variabili): in spazi di dimensione alta è difficile trovare "vicini" davvero vicini. Questo è legato alla **curse of dimensionality** e spiega perché la qualità di kNN cala quando il numero di feature è grande.

4.3 Curse of Dimensionality and k-NN

Il metodo k-NN funziona bene quando i dati stanno in uno spazio con poche variabili (povera dimensionalità), ma diventa molto meno efficace quando il numero di variabili/predittori cresce. Questo problema generale è chiamato **curse of dimensionality**.

Idee di base

Nella regressione k-NN, per fare una previsione in un nuovo punto x:

$$\hat{y}(x) = \frac{1}{k} \sum_{i \in \mathcal{N}_k(x)} y_i,$$

cioè facciamo la media dei valori y_i dei k punti più vicini a x (i suoi k vicini più prossimi).

In teoria, quando il numero di osservazioni n diventa molto grande ($n \to \infty$), la stima k-NN può avvicinarsi alla funzione vera $f(x)$. Quindi, con tantissimi dati, potremmo pensare: basta scegliere un k abbastanza grande per ridurre la varianza, ma comunque abbastanza "locale" da non aumentare troppo il bias.

Questa intuizione funziona **in bassa dimensione** (per esempio 1D, 2D). Ma **in alta dimensione** fallisce in modo pesante.

Perché l'alta dimensione è un problema?

Ci sono tre motivi principali.

1. I vicini non sono davvero vicini. Quando lo spazio ha tante dimensioni (tante feature), anche il punto più vicino può essere comunque lontano in valore assoluto.

Tradotto:

- in 1D: se cerco punti vicini a x, ne trovo davvero attaccati a x;
- in 100D: anche il "più vicino" è spesso ancora molto distante lungo alcune delle 100 direzioni.

Risultato: i k vicini usati per la media potrebbero **non rappresentare davvero** il comportamento locale di f attorno a x. Quindi l'errore di previsione aumenta.

2. Il trade-off bias-varianza peggiora. Ricorda:

- k piccolo $\Rightarrow$ bassa distorsione (bias basso) ma alta varianza.
- k grande $\Rightarrow$ bassa varianza ma alto bias (perché sto mediando anche punti lontani).

In alta dimensione succede questo:

- Se usi k piccolo, i vicini sono pochi **e** rumorosi **e** comunque lontani in molte coordinate $\Rightarrow$ stima super instabile (varianza altissima).
- Se aumenti k, per riempire il vicinato devi prendere punti sempre più lontani da x $\Rightarrow$ stai facendo una media di punti che non assomigliano più a x $\Rightarrow$ bias enorme.

Quindi succede il peggio di entrambi i mondi:

$$\text{Alta varianza con } k \text{ piccolo} \quad \text{e} \quad \text{alto bias con } k \text{ grande.}$$

3. Perdi la "località". L'idea di base di k-NN è locale: "per predire in x, guarda i punti *vicini* ad x". In alta dimensione, però, il concetto di "vicino" si rompe:

- È difficile trovare abbastanza punti veramente simili a x in tutte le dimensioni.
- Per riuscire a trovare k punti, devi allargare troppo la regione attorno a x.

Visivamente: in 2D puoi prendere un piccolo cerchietto attorno a x e dentro ci sono già tanti punti del dataset. In 50D, per catturare lo stesso % di punti del dataset devi esplorare una regione enorme dello spazio. Quella regione non è più "locale", è praticamente tutto il mondo. Quindi la media che fai non è più una media di veri vicini: è una media globale camuffata da locale.

Perché non basta avere tanti dati?

Uno potrebbe dire: "Ok, basta raccogliere più dati". Il problema è che il numero di punti necessari per riempire bene lo spazio cresce in modo esplosivo con le dimensioni.

Esempio concettuale: per campionare bene un intervallo 1D $[0, 1]$ ti servono magari 100 punti. Per campionare bene un quadrato 2D $[0, 1]\times[0, 1]$, magari te ne servono $100\times100 = 10{,}000$. Per campionare bene un cubo 10D $[0, 1]^{10}$, i punti necessari esplodono. Questo è un altro modo di vedere la **curse of dimensionality**: lo spazio vuoto cresce più veloce dei dati che riesci a raccogliere.

Conclusione: anche se aumenti n, spesso rimani comunque "scarso" rispetto al volume dello spazio in alta dimensione. Quindi k-NN non riesce a trovare veri vicini locali.

Effetto finale sulla previsione con k-NN

Quando la dimensione è alta:

- i k vicini mediati non sono davvero simili al punto di interesse x;
- la stima diventa molto rumorosa oppure troppo liscia;
- l'errore di previsione cresce.

Tradotto brutalmente: **k-NN soffre in alta dimensione**. È ottimo per dati bassi-dimensionali con relazioni non lineari, ma crolla se hai tante feature.

Come possiamo mitigare il problema?

Ci sono alcune strategie classiche:

- **Riduzione della dimensionalità** Cerca di ridurre il numero di variabili prima di applicare k-NN. Esempi:
  - PCA (Principal Component Analysis): proietta i dati in poche componenti principali.
  - Autoencoder: rete neurale che comprime e poi ricostruisce i dati.
  - Feature selection: tieni solo le feature davvero utili.
- **Usare una distanza più adatta** La distanza euclidea può essere ingannevole in alta dimensione. A volte funzionano meglio:
  - distanza di Mahalanobis (tiene conto della correlazione tra variabili),
  - similarità coseno (angolo tra i vettori), se conta più la direzione che la scala.

  Non è una cura magica, ma può aiutare se le feature hanno struttura geometrica particolare.
- **Normalizzare / standardizzare le feature** Se una variabile ha numeri molto più grandi delle altre, domina la distanza e rompe la nozione di "vicino". Standardizzare (togli media e dividi per la deviazione standard) è praticamente obbligatorio.
- **Usare modelli più adatti all'alta dimensione** In regressione, metodi come:
  - regressione lineare con regolarizzazione (ridge, lasso),
  - alberi e foreste random,
  - gradient boosting,
  - reti neurali,

  spesso funzionano meglio di k-NN quando il numero di feature è grande, perché non dipendono solo sul concetto geometrico di "vicinato locale" nello spazio delle feature grezze.

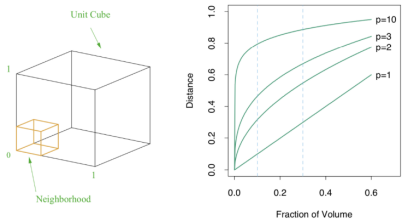


Figure 16: *One illustration of the curse of dimensionality. Left: A neighborhood cube Q_x (orange) as a subset of the unit cube Q. Right: The horizontal axis is the fraction of volume r that the neighborhood cube contains. The vertical axis is the required side length $e_d(r)$ of the neighborhood cube Q_x. Curves plotting $e_d(r)$ for $d = 1, 2, 3, 10$ are shown (note: in the figure, $p = d$).*

Riassunto da ricordare

- k-NN funziona bene quando i dati sono bassi-dimensionali e la relazione $x \mapsto y$ è anche fortemente non lineare.
- Quando la dimensione è alta, "vicino" smette di significare qualcosa di veramente locale.
- Questo fa esplodere l'errore di previsione:
  - **varianza alta** con k piccolo,
  - **bias alto** con k grande.
- Questa è una manifestazione tipica della **curse of dimensionality**.

Chapter 5

Ridge regression

5.1 Introduzione alla ridge regression

Negli ultimi vent'anni, le nuove tecnologie hanno profondamente cambiato il modo in cui i dati vengono raccolti in moltissimi ambiti: dalla finanza, al marketing, fino alla medicina. Sensori, piattaforme digitali, dispositivi indossabili e basi di dati elettroniche permettono oggi di misurare e registrare un'enorme quantità di informazioni per ogni individuo o unità di analisi.

È ormai normale lavorare con dataset che contengono un grande numero di **predittori** (o variabili esplicative), indicati con p. Queste variabili possono rappresentare, ad esempio, caratteristiche molecolari in biologia, indicatori economici in finanza, o tratti comportamentali in marketing.

Tuttavia, mentre il numero di variabili p cresce rapidamente, il numero di osservazioni n (cioè i campioni o i soggetti misurati) resta spesso limitato. Questo accade per vari motivi:

- **costi elevati** nella raccolta dei dati (ad esempio campioni clinici o analisi di laboratorio);
- **scarsa disponibilità di soggetti** (come nei casi di malattie rare);
- oppure **vincoli pratici o etici** che impediscono di ottenere molte repliche.

Il problema dell'alta dimensionalità

Quando il numero di predittori p è comparabile a n o addirittura maggiore ($p > n$), diciamo che i dati sono **ad alta dimensionalità**. In queste situazioni, i metodi classici di statistica (come la regressione lineare ai minimi quadrati ordinari) diventano inadeguati, per motivi sia matematici che pratici:

- Dal punto di vista matematico, se $p > n$, la matrice $X^\top X$ non è invertibile, quindi non è possibile calcolare la soluzione classica dei minimi quadrati.
- Dal punto di vista statistico, anche quando p è solo vicino a n, la stima diventa molto instabile e sensibile al rumore, portando a **overfitting** e a scarsa capacità di generalizzazione.

Un nuovo paradigma di analisi

L'analisi dei dati ad alta dimensionalità richiede quindi un nuovo approccio:

- tecniche di **selezione delle variabili** (per scegliere solo i predittori rilevanti);

- metodi di **regolarizzazione** (per stabilizzare le stime e ridurre la varianza);
- e strumenti di **validazione e controllo della complessità** (come la cross-validation o i criteri informativi).

In questo contesto, l'obiettivo non è più solo "trovare la miglior stima" sul dataset disponibile, ma costruire modelli che siano in grado di **generalizzare** bene a nuovi dati, anche quando la dimensionalità è elevata.

In sintesi:

Alta dimensionalità: $p \gg n \quad \Longrightarrow \quad$ metodi classici inadeguati, servono tecniche di regolarizzazione e selezione.

5.1.1 Notazione matriciale

Per rappresentare in modo compatto i modelli lineari, usiamo la **notazione matriciale**. Questa scrittura è molto utile quando lavoriamo con molti predittori, perché ci permette di manipolare le equazioni in forma vettoriale anziché una per volta.

Vettore delle risposte

Indichiamo con y il vettore (colonna) delle risposte osservate:

$$\mathbf{y}_{n\times 1} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_i \\ \vdots \\ y_n \end{bmatrix}$$

Ogni y_i rappresenta il valore della variabile dipendente (cioè la risposta osservata) per la i-esima osservazione.

Matrice dei predittori (design matrix)

Indichiamo con X la **matrice del disegno sperimentale** o **design matrix**, che raccoglie tutti i predittori (variabili esplicative).

$$\mathbf{X}_{n\times p} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1j} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2j} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{i1} & x_{i2} & \cdots & x_{ij} & \cdots & x_{ip} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nj} & \cdots & x_{np} \end{bmatrix}$$

- Ogni **riga** $x_i^\top$ (di dimensione $1 \times p$) rappresenta un'osservazione, cioè tutti i valori delle p variabili per il soggetto i.

- Ogni **colonna** x_j (di dimensione $n \times 1$) rappresenta un predittore, cioè i valori della stessa variabile per tutte le osservazioni.

Scrivendo in modo più compatto:

$$\mathbf{X} = \begin{bmatrix} x_1 & x_2 & \dots & x_p \end{bmatrix} = \begin{bmatrix} x_1^\top \\ x_2^\top \\ \vdots \\ x_n^\top \end{bmatrix}.$$

Interpretazione intuitiva

- n = numero di osservazioni (righe della matrice);
- p = numero di predittori o variabili esplicative (colonne della matrice);
- x_{ij} = valore della variabile j-esima per l'osservazione i-esima.

Esempio pratico (piccolo dataset con $n = 3$ osservazioni e $p = 2$ predittori):

$$\mathbf{X} = \begin{bmatrix} 1.2 & 3.4 \\ 2.1 & 4.8 \\ 0.7 & 2.9 \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} 10.2 \\ 12.5 \\ 9.4 \end{bmatrix}.$$

Qui ogni riga di $\mathbf{X}$ contiene le due variabili x_1, x_2 misurate su una specifica unità, e $\mathbf{y}$ contiene il valore della risposta corrispondente. Questa forma è la base per scrivere il modello lineare in modo compatto come:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}.$$

5.1.2 Linear regression

In questo paragrafo descriviamo la regressione lineare in notazione matriciale, con linguaggio semplice e formule essenziali.

Setup e assunzioni

Abbiamo:

$$\mathbf{y} \in R^{n\times 1}, \qquad \mathbf{X} \in R^{n\times p}, \qquad \boldsymbol{\beta} \in R^{p\times 1}, \qquad \boldsymbol{\varepsilon} \in R^{n\times 1}.$$

Il modello è

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}, \qquad E[\boldsymbol{\varepsilon}] = \mathbf{0}, \qquad \mathrm{Var}(\boldsymbol{\varepsilon}) = \sigma^2 \mathbf{I}_n.$$

Assumiamo $\mathrm{rank}(\mathbf{X}) = p$ (colonne linearmente indipendenti) e in genere $p < n$.

Problema ai minimi quadrati (OLS)

Stimiamo $\boldsymbol{\beta}$ minimizzando la somma dei residui al quadrato:

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}\in R^p} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 = \arg\min_{\boldsymbol{\beta}} \sum_{i=1}^{n} (y_i - \mathbf{x}_i^\top \boldsymbol{\beta})^2,$$

dove $\|\mathbf{v}\|_2 = \sqrt{\mathbf{v}^\top \mathbf{v}} = \left(\sum_{j=1}^{p} v_j^2\right)^{1/2}$.

Soluzione chiusa (normal equations)

Se $\text{rank}(\mathbf{X}) = p$,

$$\boxed{\hat{\boldsymbol{\beta}} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}}$$

Valori adattati e residui

$$\hat{\mathbf{y}} = \mathbf{X}\hat{\boldsymbol{\beta}}, \qquad \mathbf{e} = \mathbf{y} - \hat{\mathbf{y}}.$$

La **hat matrix** è $\mathbf{H} = \mathbf{X}(\mathbf{X}^\top \mathbf{X})^{-1}\mathbf{X}^\top$, quindi $\hat{\mathbf{y}} = \mathbf{H}\mathbf{y}$ e $\mathbf{e} = (\mathbf{I}_n - \mathbf{H})\mathbf{y}$. La sua traccia è $\text{tr}(\mathbf{H}) = p$ (gradi di libertà del modello).

Stima della varianza del rumore

La somma dei residui al quadrato è $\text{RSS} = \|\mathbf{e}\|_2^2 = \sum_{i=1}^n e_i^2$. La stima (non distorta) di σ^2 è:

$$\boxed{\hat{\sigma}^2 = \frac{\text{RSS}}{n-p}}$$

Incertezza dei coefficienti

Sotto le assunzioni classiche (errori omoschedastici e non correlati),

$$\text{Var}(\hat{\boldsymbol{\beta}}) = \sigma^2 (\mathbf{X}^\top \mathbf{X})^{-1} \quad \Rightarrow \quad \widehat{\text{Var}}(\hat{\boldsymbol{\beta}}) = \hat{\sigma}^2 (\mathbf{X}^\top \mathbf{X})^{-1}.$$

Intervalli di confidenza e test sui coefficienti si ottengono da queste varianze stimate.

Predizione su nuovi dati (test)

Dato un nuovo design $\mathbf{X}^* \in R^{m \times p}$,

$$\hat{\mathbf{y}}^* = \mathbf{X}^* \hat{\boldsymbol{\beta}}.$$

Varianza della **media predetta** nel punto $\mathbf{x}^{*\top}$:

$$\text{Var}\big(\mathbf{x}^{*\top} \hat{\boldsymbol{\beta}}\big) = \sigma^2 \, \mathbf{x}^{*\top} (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{x}^*.$$

Varianza della **predizione** (che include anche il rumore futuro):

$$\text{Var}\big(Y^* - \mathbf{x}^{*\top} \hat{\boldsymbol{\beta}}\big) = \sigma^2 \Big(1 + \mathbf{x}^{*\top} (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{x}^*\Big).$$

Proprietà chiave

- **Non distorsione (corretta specificazione):** se $E[\mathbf{y}] = \mathbf{X}\boldsymbol{\beta}$, allora $E[\hat{\boldsymbol{\beta}}] = \boldsymbol{\beta}$.
- **Gauss–Markov:** tra gli stimatori lineari non distorti, $\hat{\boldsymbol{\beta}}$ ha varianza minima (BLUE).

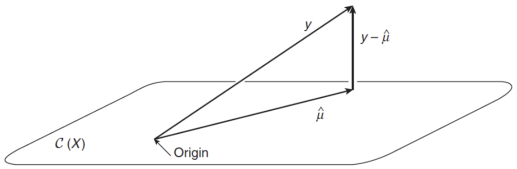
Note pratiche

- **Intercetta:** includerla aggiungendo una colonna di 1 in $\mathbf{X}$.
- **Scala delle variabili:** centrare/scalare può aiutare l'interpretazione e la stabilità numerica.
- **Collinearità:** se le colonne di $\mathbf{X}$ sono quasi collineari, $(\mathbf{X}^\top\mathbf{X})^{-1}$ è instabile $\Rightarrow$ grande varianza; valutare ridge/lasso.
- **Caso $p \geq n$: $\mathbf{X}^\top\mathbf{X}$** non è invertibile $\Rightarrow$ servono metodi con regolarizzazione o selezione di variabili.
- **Calcolo:** preferire decomposizioni numericamente stabili (QR/SVD) alle normal equations in implementazioni pratiche.

5.1.3 The Geometry of Least Squares

L'idea chiave: la regressione ai minimi quadrati è una **proiezione ortogonale** del vettore dei dati $\mathbf{y} \in R^n$ sullo **spazio delle colonne** di $\mathbf{X}$, indicato con $\mathcal{C}(\mathbf{X})$. Il risultato di questa proiezione è il vettore dei valori adattati

$$\hat{\boldsymbol{\mu}} = \hat{\mathbf{y}} = \mathbf{X}\hat{\boldsymbol{\beta}} = \mathbf{H}\mathbf{y}, \qquad \mathbf{H} = \mathbf{X}(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top.$$



Proiezione e residui

- $\hat{\mathbf{y}} \in \mathcal{C}(\mathbf{X})$: è il punto di $\mathcal{C}(\mathbf{X})$ più vicino a $\mathbf{y}$.
- I residui sono la parte "perpendicolare" a $\mathcal{C}(\mathbf{X})$:

$$\mathbf{e} = \mathbf{y} - \hat{\mathbf{y}} = (\mathbf{I}_n - \mathbf{H})\mathbf{y}.$$

- $\hat{\mathbf{y}}$ ed $\mathbf{e}$ sono **ortogonali**:

$$\hat{\mathbf{y}}^\top\mathbf{e} = 0.$$

  Equivalente: $\mathbf{X}^\top\mathbf{e} = \mathbf{0}$ (equazioni normali).

Proprietà della hat matrix H

$$\mathbf{H} = \mathbf{H}^\top \quad \text{(simmetrica)}, \qquad \mathbf{H}^2 = \mathbf{H} \quad \text{(idempotente)}, \qquad \text{rank}(\mathbf{H}) = p.$$

$\mathbf{H}$ proietta su $\mathcal{C}(\mathbf{X})$, mentre $\mathbf{I}_n - \mathbf{H}$ proietta sull'ortogonale $\mathcal{C}(\mathbf{X})^\perp$. Gli elementi diagonali h_{ii} sono le **leverages**: misurano "quanto" l'osservazione i influenza $\hat{y}_i$ ($0 \leq h_{ii} \leq 1$, $\sum_i h_{ii} = p$).

Decomposizione pitagorica

Dalla scomposizione ortogonale $\mathbf{y} = \hat{\mathbf{y}} + \mathbf{e}$ segue

$$\|\mathbf{y}\|_2^2 = \|\hat{\mathbf{y}}\|_2^2 + \|\mathbf{e}\|_2^2.$$

Se il modello include l'intercetta (quindi si lavora con variabili centrate), questa identità diventa la classica

$$\text{TSS} = \text{ESS} + \text{RSS},$$

dove $\text{RSS} = \|\mathbf{e}\|_2^2$ è la somma dei residui al quadrato e $\text{ESS} = \|\hat{\mathbf{y}} - \bar{y}\mathbf{1}\|_2^2$ è la parte spiegata.

Perché è una proiezione?

Il punto di $\mathcal{C}(\mathbf{X})$ più vicino a $\mathbf{y}$ minimizza $\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2$. La condizione di ottimo impone ortogonalità del residuo allo spazio: $\mathbf{X}^\top(\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}}) = \mathbf{0}$, da cui $\hat{\boldsymbol{\beta}} = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{y}$ e quindi $\hat{\mathbf{y}} = \mathbf{H}\mathbf{y}$.

Lettura del disegno

Nel grafico, il piano rappresenta $\mathcal{C}(\mathbf{X})$. Il vettore $\hat{\mathbf{y}}$ è la proiezione di $\mathbf{y}$ sul piano. Il segmento verticale è il residuo $\mathbf{y} - \hat{\mathbf{y}}$, perpendicolare al piano. La "triangolazione" visiva è esattamente la decomposizione pitagorica sopra.

5.1.4 Alternatives to Least Squares

La regressione ai minimi quadrati ordinari (OLS) funziona bene solo quando il numero di osservazioni n è molto più grande del numero di predittori p. In caso contrario, l'algoritmo diventa instabile o addirittura impossibile da applicare. Vediamo perché e in quali situazioni servono metodi alternativi (come ridge o lasso).

Motivazione

Nel caso **Fixed-X**, la varianza dello stimatore OLS cresce con la complessità del modello. In media possiamo scrivere:

$$\text{Var}(\hat{\boldsymbol{\beta}}) \approx \frac{\sigma^2 p}{n}.$$

- Se $n \gg p$: la varianza è piccola, OLS funziona bene.
- Se np: la varianza diventa grande $\rightarrow$ instabilità e rischio di overfitting.
- Se $n = p$: OLS si adatta perfettamente ai dati ($\hat{y}_i = y_i$), ma generalizza malissimo.
- Se $n < p$: il sistema è sottodeterminato, $\mathbf{X}^\top\mathbf{X}$ non è invertibile, e OLS non è definito.

In breve, **quando p cresce rispetto a n**, occorre introdurre vincoli o regolarizzazioni per ridurre la varianza.

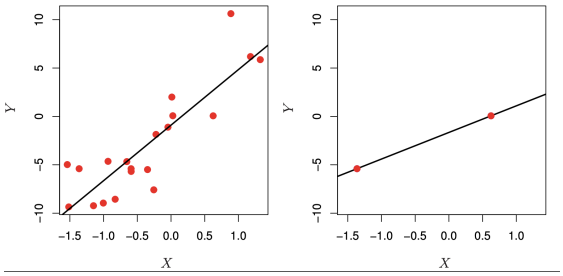
1) Caso $p < n$: soluzione unica

Quando le colonne di $\mathbf{X}$ sono linearmente indipendenti ($\text{rank}(\mathbf{X}) = p$), la soluzione OLS è univoca:

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}.$$

Questo è il caso classico in cui OLS è ben definito e stabile. Geometricamente, la soluzione è la proiezione di $\mathbf{y}$ nello spazio delle colonne di $\mathbf{X}$, di dimensione $p < n$.

A sinistra: regressione lineare classica in bassa dimensione. A destra: caso limite con due osser-



vazioni ($n = 2$) e due parametri da stimare (intercetta + coefficiente): la retta passa esattamente per entrambi i punti.

2) Caso $n \approx p$: instabilità numerica

Anche se $\mathbf{X}^\top \mathbf{X}$ è invertibile, può essere quasi singolare, cioè mal condizionata. In tal caso:

$$\text{Var}(\hat{\boldsymbol{\beta}}) = \sigma^2 (\mathbf{X}^\top \mathbf{X})^{-1}$$

diventa molto grande. Piccole variazioni nei dati portano a enormi cambiamenti nei coefficienti stimati.

Esempio: con $n = 10$, $p = 9$, e vero vettore $\boldsymbol{\beta} = (1, 0, 0, \ldots, 0)^\top$, otteniamo stime dei coefficienti con valori molto grandi (in valore assoluto), segno di overfitting.

$$Y = \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_9 X_9 + \varepsilon$$

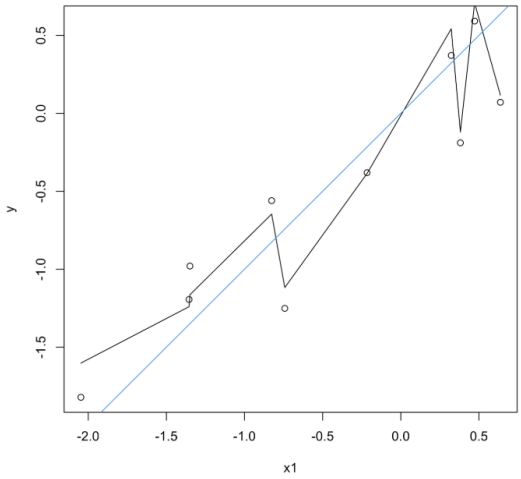
```
n <- 10; p <- 9
set.seed(1793)
X <- matrix(rnorm(n * p), n, p)
y <- X[,1] + rnorm(n, 0, 0.5)
fit <- lm(y ~ 0 + X)
round(coef(fit), 3)
```

Output:

X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9
13.528	−5.466	−24.512	−6.611	2.732	−10.671	15.473	16.831	−11.331

I coefficienti hanno magnitudini elevate e irregolari: il modello ha memorizzato il rumore, non la relazione vera.

```
yhat <- predict(fit)
plot(X[ ,1], y, xlab = "x1")
ix <- sort(X[ ,1], index.return = T)$ix
lines(X[ix, 1], yhat[ix])
abline(a = 0, b = 1, col = 4)
```



3) Caso $p > n$: fallimento dei minimi quadrati

Quando ci sono più predittori che osservazioni ($p > n$), la matrice $\mathbf{X}^\top\mathbf{X}$ non è invertibile:

$$\text{rank}(\mathbf{X}) < p,$$

e quindi il sistema delle equazioni normali

$$\mathbf{X}^\top\mathbf{X}\boldsymbol{\beta} = \mathbf{X}^\top\mathbf{y}$$

ha infinite soluzioni. Lo stimatore OLS non è più definito.

Sia $\mathcal{C}(\mathbf{X})$ lo spazio delle colonne di $\mathbf{X}$ (di dimensione n) e $\mathcal{V} = \mathcal{C}(\mathbf{X})^\perp$ lo spazio ortogonale nullo (null space) di dimensione $p - n$. Per ogni vettore $\mathbf{v} \in \mathcal{V}$ vale:

$$\mathbf{X}\mathbf{v} = 0_p, \quad \mathbf{X}^\top\mathbf{X}\mathbf{v} = 0_n.$$

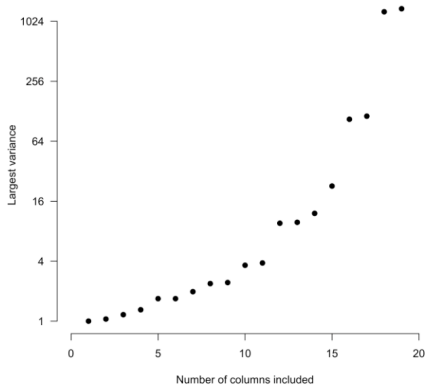
La soluzione generale delle equazioni normali è:

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^\top\mathbf{X})^-\mathbf{X}^\top\mathbf{y} + \mathbf{v}, \quad \forall \mathbf{v} \in \mathcal{V},$$

dove $(\mathbf{X}^\top\mathbf{X})^-$ è la pseudoinversa di Moore–Penrose.

Conseguenza pratica

Nel caso $p > n$, la varianza dei coefficienti cresce rapidamente e il modello OLS diventa privo di senso statistico. Nell'esempio seguente (matrice $\mathbf{X}$ con $n = 20$ e colonne generate casualmente



da una normale standard), la varianza dei coefficienti stimati aumenta senza limite man mano che p cresce.

Conclusione: OLS non è adatto ai contesti ad alta dimensionalità. Occorre introdurre metodi che riducano la varianza — come la **regolarizzazione** — ad esempio:

Ridge Regression (L2 penalty), Lasso Regression (L1 penalty), Elastic Net (L1 + L2).

Questi metodi introducono un compromesso tra bias e varianza per ottenere stime più stabili e generalizzabili.

5.1.5 Condition number

Nel modello lineare, lo stimatore OLS risolve le equazioni normali

$$\mathbf{X}^\top \mathbf{X}\,\boldsymbol{\beta} = \mathbf{X}^\top \mathbf{y}.$$

La **difficoltà numerica** (stabilità) nel risolvere questo sistema si riassume nel **numero di condizione**.

Definizione (norma 2)

Per una matrice A a rango pieno,

$$\kappa_2(A) = \frac{\sigma_{\max}(A)}{\sigma_{\min}(A)},$$

dove $\sigma_{\max}$ e $\sigma_{\min}$ sono le **singular values** massima e minima di A (da SVD). Nel nostro caso, se guardiamo $A = \mathbf{X}^\top \mathbf{X}$,

$$\kappa_2(\mathbf{X}^\top \mathbf{X}) = \frac{\lambda_{\max}(\mathbf{X}^\top \mathbf{X})}{\lambda_{\min}(\mathbf{X}^\top \mathbf{X})} = \left(\frac{\sigma_{\max}(\mathbf{X})}{\sigma_{\min}(\mathbf{X})}\right)^2 = \kappa_2(\mathbf{X})^2.$$

Nota pratica: è spesso più informativo calcolare $\kappa_2(\mathbf{X})$ (non di $\mathbf{X}^\top \mathbf{X}$), perché quest'ultimo **quadruplica** i problemi di condizionamento.

Interpretazione

- κ misura quanto **amplifica** gli errori (o piccole perturbazioni) nei dati/nei calcoli: se $\delta\mathbf{y}$ è un piccolo disturbo, l'errore relativo su $\hat{\boldsymbol{\beta}}$ può crescere di un fattore $\approx \kappa$ (stima di ordine).
- κ grande $\Rightarrow$ matrice **mal condizionata** (quasi singolare) $\Rightarrow$ soluzione instabile, varianze grandi:
$$\mathrm{Var}(\hat{\boldsymbol{\beta}}) = \sigma^2(\mathbf{X}^\top\mathbf{X})^{-1} \quad \text{esplode se} \quad \lambda_{\min}(\mathbf{X}^\top\mathbf{X}) \text{ è piccola.}$$
- Collegamento con **multicollinearità**: colonne quasi combinate linearmente $\Rightarrow \sigma_{\min}(\mathbf{X})\downarrow$ $\Rightarrow \kappa\uparrow$.

Regole pratiche (rule of thumb)

$\kappa_2(\mathbf{X})10$: ben condizionata; 10–30 : attenzione; > 30 (o > 100) : problemi seri di collinearità.

Ricorda: $\kappa_2(\mathbf{X}^\top\mathbf{X}) = \kappa_2(\mathbf{X})^2$; quindi soglie su $\mathbf{X}^\top\mathbf{X}$ vanno **al quadrato**.

Come si calcola

$$\kappa_2(\mathbf{X}) = \frac{\sigma_{\max}(\mathbf{X})}{\sigma_{\min}(\mathbf{X})} \quad \text{(via SVD).}$$

R (esempio):

```
# kappa basato su SVD (norma 2)
kappa_X  <- function(X) {
  s <- svd(X, nu = 0, nv = 0)$d
  max(s) / min(s)
}

# oppure: base R (attenzione: per default usa X'X)
kappa_xtx <- kappa(crossprod(X))   # ~ kappa_X(X)^2
kappa2    <- kappa_X(X)            # consigliato
```

Perché ci importa

- **Instabilità dei coefficienti**: piccoli cambi nei dati $\Rightarrow$ grandi cambi in $\hat{\boldsymbol{\beta}}$.
- **Intervalli di confidenza larghi** e **p-value fuorvianti** (alta varianza).
- **Predizioni instabili** fuori dal range dei dati.

Come mitigare

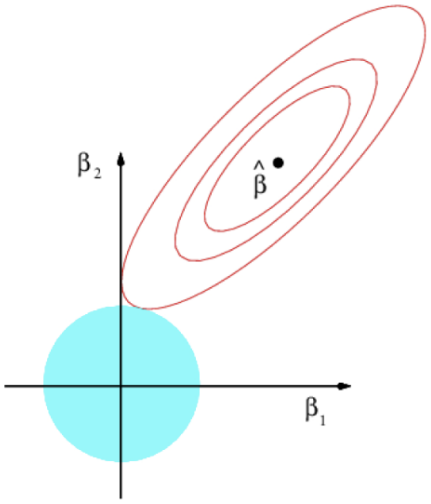
- **Standardizzare** le variabili (centering & scaling); usare **polinomi ortogonali** al posto dei "raw".
- **Eliminare/aggregare** predittori quasi duplicati; **feature selection**.
- **Ridge** (L2) o **Elastic Net**: aggiungono penalità che aumentano gli autovalori effettivi $\Rightarrow$ riducono κ e la varianza.
- **PCA/PLS**: usare componenti ben condizionate come regressori.

Geometria in breve

Colonne quasi allineate ⇒ lo spazio delle colonne di $\mathbf{X}$ è "schiacciato" lungo alcune direzioni: la proiezione (fit) diventa **molto sensibile** al rumore in quelle direzioni. Il numero di condizione quantifica proprio quanto "schiacciato" è questo spazio.

5.2 Ridge regression

L'idea: invece di scegliere un sottoinsieme di variabili e fare OLS sul sottoinsieme, teniamo **tutte** le p variabili ma **riduciamo** (shrink) i coefficienti verso 0 per stabilizzare la stima.



Definizione

Dato $\mathbf{X} \in R^{n \times p}$ e $\mathbf{y} \in R^n$, il ridge stima

$$\hat{\boldsymbol{\beta}}_\lambda = \arg\min_{\boldsymbol{\beta}} \ \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda \|\boldsymbol{\beta}\|_2^2 \qquad (\lambda \geq 0).$$

Soluzione in forma chiusa:

$$\hat{\boldsymbol{\beta}}_\lambda = (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}_p)^{-1} \mathbf{X}^\top \mathbf{y}.$$

Importante: si **standardizzano** i predittori (centra e scala le colonne di $\mathbf{X}$), e **non** si penalizza l'intercetta.

Perché aiuta

- Quando $n \approx p$ o c'è multicollinearità, OLS ha varianza alta. Il termine $\lambda\|\beta\|_2^2$ riduce la varianza (a costo di un piccolo bias).
- $(\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})$ è **meglio condizionata** di $\mathbf{X}^\top \mathbf{X}$: si evitano instabilità numeriche.
- Funziona anche quando $p > n$ (dove OLS non è definita).

Vista geometrica e SVD

Con la SVD $\mathbf{X} = \mathbf{U}\mathbf{D}\mathbf{V}^\top$, con $\mathbf{D} = \mathrm{diag}(d_1, \dots, d_r)$:

$$\hat{\boldsymbol{\beta}}_\lambda = \mathbf{V}\,\mathrm{diag}\left(\frac{d_j}{d_j^2 + \lambda}\right)\mathbf{U}^\top \mathbf{y}.$$

Ogni componente lungo una direzione poco informativa (singular value d_j piccolo) viene **schiacciata** più forte: $\frac{d_j}{d_j^2+\lambda} \ll 1$.

Hat matrix ed "effective degrees of freedom"

Le predizioni ridge sono $\hat{\boldsymbol{\mu}}_\lambda = \mathbf{S}_\lambda \mathbf{y}$ con

$$\mathbf{S}_\lambda = \mathbf{X}(\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^\top.$$

I **gradi di libertà effettivi** del fit sono

$$\mathrm{df}(\lambda) = \mathrm{tr}(\mathbf{S}_\lambda) = \sum_{j=1}^{r} \frac{d_j^2}{d_j^2 + \lambda},$$

decrescenti in λ: più penalizzazione $\Rightarrow$ modello più "rigido".

Bias–varianza

- $\lambda \uparrow \Rightarrow$ **varianza** delle predizioni $\downarrow$, **bias** $\uparrow$.
- Esiste un λ ottimale (tipicamente trovato con CV) che minimizza l'errore di test.

Ridge vs OLS vs Lasso

- **OLS**: nessuna penalità, varianza alta se collinearità o $n \approx p$.
- **Ridge** (ℓ_2): **non fa selezione esatta** (i coefficienti raramente diventano zero), ma riduce la varianza e gestisce bene multicollinearità e $p > n$.
- **Lasso** (ℓ_1): può annullare esattamente alcuni coefficienti (selezione di variabili), ma può essere meno stabile quando le variabili sono molto correlate.

Scelta di λ

Si usa di solito **K-fold cross-validation** (o GCV) sul training set:

$$\lambda^\star = \arg\min_\lambda \ \widehat{\mathrm{Err}}_{\mathrm{CV}}(\lambda).$$

Spesso si riporta anche la regola "*one-standard-error*" per preferire un modello più semplice: scegliere il valore più grande di λ il cui CV error è entro 1 s.e. dal minimo.

Note pratiche

- **Standardizza** i predittori (media 0, varianza 1). Non penalizzare l'intercetta.
- Ridge riduce gli effetti di collinearità distribuendo il peso tra predittori correlati.
- Con molte variabili deboli e rumore, ridge può dare predizioni sensibilmente più stabili di OLS.

R: implementazione essenziale

Soluzione in forma chiusa (per studio).

```
# X: predittori (senza intercetta), y: risposta
Xs <- scale(X, center = TRUE, scale = TRUE)
ys <- y - mean(y)

lambda <- 0.5
beta_ridge <- solve(crossprod(Xs) + lambda * diag(ncol(Xs)),
                    crossprod(Xs, ys))
yhat <- drop(Xs %*% beta_ridge + mean(y))
```

Con `glmnet` e K-fold CV.

```
library(glmnet)
# glmnet standardizza di default; alpha=0 -> ridge
fit <- glmnet(X, y, alpha = 0)
cv  <- cv.glmnet(X, y, alpha = 0, nfolds = 10)

lambda_min <- cv$lambda.min      # minimo CV error
lambda_1se <- cv$lambda.1se      # più parsimonioso (1-SE rule)

coef_min <- coef(cv, s = "lambda.min")
coef_1se <- coef(cv, s = "lambda.1se")
yhat_cv  <- predict(cv, newx = X, s = "lambda.min")
```

Messaggio chiave

Ridge **non elimina** variabili ma **stabilizza** la stima dei coefficienti. Questo controllo della complessità (via λ) riduce l'overfitting e, in molti scenari pratici, **migliora l'errore di test** rispetto a OLS.

5.2.1 Constrained estimation

Il metodo dei minimi quadrati stima i coefficienti $\beta_0, \ldots, \beta_p$ minimizzando la somma dei quadrati dei residui:

$$RSS = \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 = (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^\top (\mathbf{y} - \mathbf{X}\boldsymbol{\beta}).$$

Problema

Se la matrice $\mathbf{X}^\top\mathbf{X}$ è mal condizionata o quasi singolare, la stima OLS

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{y}$$

diventa numericamente instabile: piccoli cambiamenti nei dati producono grandi variazioni nei coefficienti. Questo accade tipicamente quando i predittori sono fortemente correlati (*multicollinearità*) o quando $p \approx n$.

Idea di Ridge regression

Si "stabilizzano" le equazioni normali aggiungendo un termine di penalità diagonale:

$$(\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I}_p)\boldsymbol{\beta} = \mathbf{X}^\top\mathbf{y}, \qquad \lambda > 0.$$

La soluzione è:

$$\hat{\boldsymbol{\beta}}_\lambda = (\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I}_p)^{-1}\mathbf{X}^\top\mathbf{y}.$$

Questa è la **stima Ridge** proposta da Hoerl e Kennard (1970). Il termine $\lambda\mathbf{I}_p$ impedisce che $\mathbf{X}^\top\mathbf{X}$ sia singolare e riduce la varianza dei coefficienti.

Effetto sul numero di condizione

Il numero di condizione della matrice modificata è:

$$\kappa(\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I}_p) = \frac{d_{\max} + \lambda}{d_{\min} + \lambda},$$

dove $d_{\max}$ e $d_{\min}$ sono i valori singolari massimo e minimo di $\mathbf{X}^\top\mathbf{X}$. Anche se $d_{\min} = 0$, il numero di condizione resta **finito** per ogni $\lambda > 0$, garantendo stabilità numerica. Questa procedura è nota come **regolarizzazione di Tikhonov** (dal matematico Andrey Tikhonov).

Interpretazione geometrica

Ridge può essere interpretato come un problema di **ottimizzazione vincolata**: invece di minimizzare solo l'errore di training, imponiamo un vincolo sulla grandezza dei coefficienti:

$$\min_{\boldsymbol{\beta}\in R^p} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 \quad \text{soggetto a} \quad \|\boldsymbol{\beta}\|_2^2 \le t,$$

dove $t > 0$ controlla la quantità di "shrinkage": più piccolo è t, più i coefficienti sono costretti verso 0.

Relazione con la forma penalizzata: attraverso il metodo dei moltiplicatori di Karush–Kuhn–Tucker (KKT), si dimostra che il problema vincolato è equivalente al problema penalizzato:

$$\min_{\boldsymbol{\beta}} \left\{ \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_2^2 \right\},$$

dove il parametro di penalità λ è in relazione biunivoca con il raggio t del vincolo.

Interpretazione intuitiva:

- OLS sceglie $\boldsymbol{\beta}$ che minimizza solo l'errore di fit;
- Ridge sceglie $\boldsymbol{\beta}$ che bilancia **bassa varianza** e **coefficienti piccoli**;
- il termine $\lambda\|\boldsymbol{\beta}\|_2^2$ agisce come un **ammortizzatore** che previene valori estremi dovuti alla collinearità.

Visualizzazione geometrica

Nel piano bidimensionale ($p = 2$):

- le ellissi del RSS rappresentano le regioni a uguale errore di fit;
- il vincolo $\|\boldsymbol{\beta}\|_2^2 \leq t$ rappresenta un cerchio (o sfera in più dimensioni);
- la soluzione Ridge è il punto di tangenza tra la più piccola ellisse del RSS e il cerchio del vincolo.

Riassunto:

- Ridge regolarizza la soluzione aggiungendo $\lambda \mathbf{I}_p$.
- Migliora la condizione numerica di $\mathbf{X}^\top \mathbf{X}$.
- Riduce la varianza dei coefficienti (a costo di un bias controllato).
- È equivalente a imporre $\|\boldsymbol{\beta}\|_2^2 \leq t$.

5.2.2 Penalized estimation

La Ridge regression può essere vista come un problema di **stima penalizzata**: invece di minimizzare soltanto l'errore di adattamento, aggiungiamo una penalità sulla grandezza dei coefficienti per controllare la complessità del modello.

Problema di minimizzazione penalizzata

$$\min_{\boldsymbol{\beta} \in R^p} \left\{ \sum_{i=1}^{n} (y_i - \mathbf{x}_i^\top \boldsymbol{\beta})^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \right\} = \min_{\boldsymbol{\beta} \in R^p} \left\{ \underbrace{\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2}_{\text{RSS}} + \underbrace{\lambda \|\boldsymbol{\beta}\|_2^2}_{\text{penalty}} \right\}.$$

Il parametro $\lambda \in [0, \infty)$ controlla la forza della penalità:

- $\lambda = 0 \Rightarrow$ stima OLS classica;
- $\lambda \to \infty \Rightarrow$ tutti i coefficienti si avvicinano a 0.

Quindi λ governa il compromesso **bias–varianza** e la **quantità di shrinkage** dei coefficienti.

Il minimo del termine RSS si ottiene in $\boldsymbol{\beta} = \hat{\boldsymbol{\beta}}_{\text{OLS}}$, mentre il minimo della penalità si ottiene in $\boldsymbol{\beta} = \mathbf{0}_p$. Il termine di penalità "tira" le stime verso lo zero, riducendone la varianza.

Derivazione analitica

Scriviamo la funzione obiettivo:

$$L(\boldsymbol{\beta}) = \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda \|\boldsymbol{\beta}\|_2^2.$$

Calcoliamo il gradiente rispetto a $\boldsymbol{\beta}$:

$$\frac{\partial L}{\partial \boldsymbol{\beta}} = -2\mathbf{X}^\top (\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) + 2\lambda \boldsymbol{\beta}.$$

Imponendo la condizione di ottimalità ($\nabla L = 0$):

$$-2\mathbf{X}^\top \mathbf{y} + 2\mathbf{X}^\top \mathbf{X}\boldsymbol{\beta} + 2\lambda\boldsymbol{\beta} = 0,$$

$$(\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}_p)\boldsymbol{\beta} = \mathbf{X}^\top \mathbf{y}.$$

La soluzione è la stima **Ridge**:

$$\boxed{\hat{\boldsymbol{\beta}}_\lambda = (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}_p)^{-1}\mathbf{X}^\top \mathbf{y}.}$$

Convessità e unicità della soluzione

La matrice Hessiana di $L(\boldsymbol{\beta})$ è

$$\nabla^2 L = 2(\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}_p).$$

Poiché $\mathbf{X}^\top \mathbf{X}$ è semidefinita positiva e $\lambda \mathbf{I}_p$ è definita positiva per $\lambda > 0$, la loro somma è **definita positiva**. Questo implica che $L(\boldsymbol{\beta})$ è **strettamente convessa** e quindi:

- esiste un unico minimo globale;
- la soluzione $\hat{\boldsymbol{\beta}}_\lambda$ è ben definita per ogni $\lambda > 0$, anche se $\mathbf{X}^\top \mathbf{X}$ non è invertibile.

Questa proprietà garantisce stabilità numerica e l'esistenza di una soluzione chiusa anche in presenza di multicollinearità o $p > n$.

Relazione con il problema vincolato

Il problema penalizzato è **equivalente** a un problema con vincolo:

$$\min_{\boldsymbol{\beta}} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 \quad \text{soggetto a} \quad \|\boldsymbol{\beta}\|_2^2 \leq t.$$

Per ogni valore del vincolo $t > 0$ esiste un corrispondente $\lambda > 0$ tale che le soluzioni coincidono:

$$\hat{\boldsymbol{\beta}}_\lambda \text{ risolve il problema vincolato con } t = \|\hat{\boldsymbol{\beta}}_\lambda\|_2^2.$$

Questa corrispondenza (detta **duale**) lega la forza della penalità (λ) alla dimensione della regione ammissibile (t).

Interpretazione intuitiva

- La penalità $\lambda\|\boldsymbol{\beta}\|_2^2$ "regolarizza" il problema, prevenendo coefficienti estremi.
- Geometricamente, si cerca la minima ellisse di RSS che tocca la sfera di raggio $\sqrt{t}$.
- La soluzione Ridge è più "centrata" e stabile, con meno varianza ma un po' di bias.

Messaggio chiave

Ridge regression bilancia **errore di fit** e **stabilità numerica**:

$$\text{Bias} \uparrow \Rightarrow \text{ Varianza} \downarrow \Rightarrow \text{ Errore di test totale ridotto.}$$

5.2.3 Solution path (traccia delle soluzioni)

Cosa succede al variare di λ.

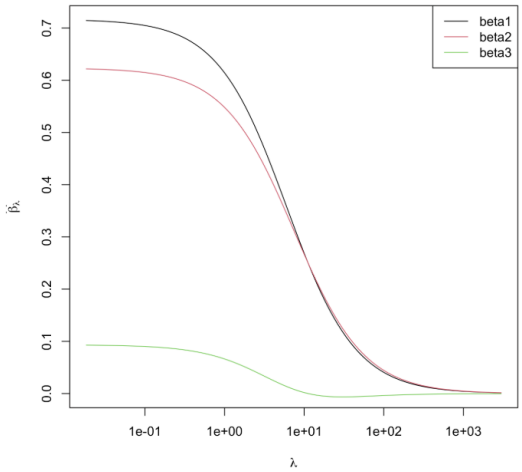
$$\lambda \to 0 \;\Rightarrow\; \hat{\boldsymbol{\beta}}_\lambda \to \hat{\boldsymbol{\beta}}_{\text{OLS}} \qquad\qquad \lambda \to \infty \;\Rightarrow\; \hat{\boldsymbol{\beta}}_\lambda \to \mathbf{0}_p$$

Aumentando λ, tutti i coefficienti stimati vengono **shrinkati** (spinti) verso lo zero. L'insieme $\{\hat{\boldsymbol{\beta}}_\lambda : \lambda \in [0, \infty)\}$ si chiama **solution path** (o **ridge trace** quando lo tracciamo graficamente).

Importante. La riduzione non è necessariamente **monotona** per ogni singolo coefficiente:

$$\lambda_a > \lambda_b \;\Rightarrow\; |\hat{\beta}_{j,\lambda_a}| < |\hat{\beta}_{j,\lambda_b}|$$

Questo può accadere per via delle **correlazioni** tra le variabili: il modo in cui la penalità ridistribuisce il peso tra coefficienti può farli "incrociare" lungo il percorso.



Esempio: ridge trace in R

```
X <- matrix(c(1,1,1,1,-1,0,2,1,2,1,-1,0), byrow=FALSE, ncol=3)
p <- ncol(X)
y <- c(1.3, -0.5, 2.6, 0.9)

lambdas <- exp(seq(-4, 8, length.out = 100))
hatbetas <- sapply(lambdas, function(lambda)
  solve(t(X) %*% X + lambda * diag(p)) %*% t(X) %*% y
)

plot(lambdas, hatbetas[1,], type = "l", log = "x",
     xlab = expression(lambda),
     ylab = expression(widehat(beta)[lambda]))
lines(lambdas, hatbetas[2,], col = 2)
lines(lambdas, hatbetas[3,], col = 3)
legend("topright", c("beta1","beta2","beta3"), col = 1:3, lty = 1)
```

Come scegliere λ in pratica

- **Ridge traces (scelta "visiva"):** si tracciano i coefficienti $\hat{\beta}_{j,\lambda}$ in funzione di λ e si sceglie un punto dove le curve sono stabili e con segni "sensati". È un criterio soggettivo e oggi poco usato da solo.
- **Cross-validation (standard):** si sceglie λ minimizzando l'errore di validazione (K-fold CV o LOOCV). È la pratica consigliata perché fornisce una scelta **data-driven** e meno soggettiva.

Messaggio chiave. Il *solution path* mostra come la penalizzazione redistribuisce la "massa" dei coefficienti al crescere di λ:

$$\text{piccola } \lambda \Rightarrow \text{ basso bias, alta varianza} \quad ; \quad \text{grande } \lambda \Rightarrow \text{ più bias, varianza ridotta.}$$

L'**ottimo** si cerca con la **CV**, puntando al miglior errore di test atteso.

5.2.4 Dettagli importanti (applicativi)

Termine di intercetta

Nella regressione ridge **l'intercetta non viene penalizzata**. Il modello si stima come

$$\min_{\alpha,\, \boldsymbol{\beta}\in R^p} \left\{ \sum_{i=1}^{n} \Big(y_i - \alpha - \sum_{j=1}^{p} x_{ij}\beta_j \Big)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \right\}.$$

Se prima di applicare la ridge **centriamo le colonne** di $\mathbf{X}$ (cioè rendiamo la media di ciascuna colonna pari a 0), ossia:

$$\tilde{\mathbf{x}}_j = \mathbf{x}_j - \tfrac{1}{n}\bar{x}_j \mathbf{1}_n, \qquad \bar{x}_j = \tfrac{1}{n}\sum_{i=1}^{n} x_{ij},$$

allora la stima dell'intercetta risulta semplicemente

$$\hat{\alpha} = \bar{y}, \qquad \bar{y} = \tfrac{1}{n}\sum_{i=1}^{n} y_i.$$

Questo avviene perché, una volta che X è centrata, la miglior costante per adattare i dati in media è proprio la media di y.

Perché standardizzare i predittori

Nel modello lineare classico OLS, il risultato non cambia se moltiplichiamo le variabili per una costante (è **invariante alla scala**); invece la ridge **non lo è**. La penalità

$$\|\boldsymbol{\beta}\|_2^2 = \sum_{j=1}^{p} \beta_j^2$$

tratta tutti i coefficienti come se fossero comparabili. Se le variabili hanno scale diverse (es. anni, kg, euro), la penalità diventa sbilanciata e produce stime distorte.

Buona pratica. Prima di applicare la regressione ridge si procede con:

1. Centrare ogni predittore (media = 0) per renderlo ortogonale all'intercetta.
2. Scalare ogni predittore a una scala comune (varianza unitaria o norma Euclidea $\sqrt{n}$).

Questo assicura che la penalità sia equa per tutte le variabili e stabilizza il modello.

Ricalcolo sulla scala originale

Definiamo:

$$\tilde{\mathbf{y}} = \mathbf{y} - \tfrac{1}{n}\bar{y}\mathbf{1}_n, \qquad \tilde{\mathbf{X}} = \Big(\mathbf{X} - \tfrac{1}{n}\mathbf{1}_n\bar{\mathbf{x}}^\top\Big)\mathrm{diag}(1/\mathbf{s}),$$

dove $\bar{\mathbf{x}} = \frac{1}{n}\mathbf{X}^\top\mathbf{1}_n$ contiene le medie delle colonne e $\mathbf{s} = (s_1, \ldots, s_p)^\top$ rappresenta le scale (ad esempio, le deviazioni standard).

Stima sui dati standardizzati:

$$\hat{\boldsymbol{\beta}}_\lambda^{(\mathrm{std})} = \big(\tilde{\mathbf{X}}^\top\tilde{\mathbf{X}} + \lambda\mathbf{I}_p\big)^{-1}\tilde{\mathbf{X}}^\top\tilde{\mathbf{y}}.$$

Ritorno alla scala originale:

$$\hat{\boldsymbol{\beta}}_\lambda = \mathrm{diag}(1/\mathbf{s})\,\hat{\boldsymbol{\beta}}_\lambda^{(\mathrm{std})}, \qquad \hat{\alpha} = \bar{y} - \bar{\mathbf{x}}^\top\hat{\boldsymbol{\beta}}_\lambda.$$

In questo modo si può lavorare su dati centrati e scalati, per poi riconvertire i coefficienti nella scala originale. Non si perde nessuna informazione.

Nota sui software

I diversi pacchetti statistici implementano standardizzazioni diverse:

- `glmnet` (impostazione predefinita) centra e scala automaticamente X, include un'intercetta e, per modelli gaussiani, può anche standardizzare y.
- Se si implementa la ridge manualmente, è importante essere espliciti su come si centra e si scala X e y per ottenere risultati coerenti.

Procedura passo-passo

1. Centrare X e $y \Rightarrow$ evita di penalizzare l'intercetta, che poi sarà $\hat{\alpha} = \bar{y}$ dopo la riconversione.
2. Scalare le colonne di $X \Rightarrow$ rende la penalità equa per tutti i predittori.
3. Risolvere $(\tilde{X}^\top\tilde{X} + \lambda I)\hat{\beta}^{(\mathrm{std})} = \tilde{X}^\top\tilde{y} \Rightarrow$ stima ridge sui dati standardizzati.
4. Tornare alla scala originale con $\hat{\beta} = \mathrm{diag}(1/s)\hat{\beta}^{(\mathrm{std})}$ e $\hat{\alpha} = \bar{y} - \bar{x}^\top\hat{\beta}$.

Messaggi chiave

- **Non penalizzare l'intercetta.** Dopo la centratura, l'intercetta stimata coincide con la media di y.
- **Standardizzare sempre i predittori** prima di applicare la ridge; successivamente riconvertire i coefficienti alla scala originale.
- **Controllare le impostazioni dei software** per sapere se la centratura e la standardizzazione vengono applicate automaticamente.

5.2.5 Ridge: calcoli con la SVD

Per calcolare l'intero **percorso di regolarizzazione** (le soluzioni $\hat{\beta}_\lambda$ per molti valori di λ) in modo efficiente, conviene usare la **Singular Value Decomposition (SVD)**.

Richiamo di SVD

Per una matrice dei predittori $\mathbf{X} \in R^{n\times p}$ centrata per colonna, la SVD (in forma "piena") è

$$\mathbf{X} = \mathbf{U}\,\mathbf{D}\,\mathbf{V}^\top,$$

dove:

- $\mathbf{U} \in R^{n\times n}$ è ortogonale ($\mathbf{U}^\top\mathbf{U} = \mathbf{I}_n$),
- $\mathbf{V} \in R^{p\times p}$ è ortogonale ($\mathbf{V}^\top\mathbf{V} = \mathbf{I}_p$),
- $\mathbf{D} \in R^{n\times p}$ è diagonale "rettangolare" con valori singolari $d_1 \geq \cdots \geq d_m \geq 0$, $m = \min\{n, p\}$, sugli elementi diagonali (gli altri sono zero).

Soluzione ridge via SVD

La soluzione ridge

$$\hat{\boldsymbol{\beta}}_\lambda = \big(\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I}_p\big)^{-1}\mathbf{X}^\top\mathbf{y}$$

si riscrive con la SVD come

$$\hat{\boldsymbol{\beta}}_\lambda = \mathbf{V}\big(\mathbf{D}^\top\mathbf{D} + \lambda\mathbf{I}_p\big)^{-1}\mathbf{D}^\top\mathbf{U}^\top\mathbf{y} = \mathbf{V}\,\mathrm{diag}\left(\frac{d_1}{d_1^2+\lambda}, \ldots, \frac{d_p}{d_p^2+\lambda}\right)\mathbf{U}^\top\mathbf{y} = \sum_{j:\,d_j>0} \mathbf{v}_j\,\frac{d_j}{d_j^2+\lambda}\,\mathbf{u}_j^\top\mathbf{y},$$

dove $\mathbf{v}_j$ e $\mathbf{u}_j$ sono le j-esime colonne di $\mathbf{V}$ e $\mathbf{U}$.

Osservazioni rapide

- Se $n > p$, per $\lambda = 0$ ritroviamo OLS.
- Se $p > n$ (più variabili che campioni), per $\lambda \downarrow 0$ otteniamo la **soluzione a norma minima** tra le infinite OLS: la SVD seleziona in modo univoco quella con $\|\beta\|_2$ più piccola.

Valori adattati e fattori di contrazione

I valori fittati ridge sono

$$\hat{\mathbf{y}}_\lambda = \mathbf{X}\hat{\boldsymbol{\beta}}_\lambda = \mathbf{U}\,\mathrm{diag}\left(\frac{d_1^2}{d_1^2+\lambda}, \ldots, \frac{d_p^2}{d_p^2+\lambda}, \underbrace{0, \ldots, 0}_{n-p}\right)\mathbf{U}^\top\mathbf{y} = \sum_{j:\,d_j>0} \mathbf{u}_j\,\frac{d_j^2}{d_j^2+\lambda}\,\mathbf{u}_j^\top\mathbf{y}.$$

Quindi ridge:

- proietta $\mathbf{y}$ sulle direzioni $\mathbf{u}_j$ (come OLS),
- poi **restringe** ciascuna coordinata tramite il fattore

$$\frac{d_j^2}{d_j^2+\lambda} \in (0, 1],$$

più piccolo quando d_j è piccolo (cioè quando la direzione spiega poca varianza in $\mathbf{X}$).

Legame con PCA

La SVD è collegata alla PCA: i **componenti principali** sono $\mathbf{Z} = \mathbf{XV} = [\mathbf{z}_1, \ldots, \mathbf{z}_p]$, con

$$\mathbf{z}_j = \mathbf{X}\mathbf{v}_j = \mathbf{u}_j d_j, \qquad \mathrm{Var}(z_j) = \frac{d_j^2}{n}.$$

La **PCR** (Principal Components Regression) con q componenti stima OLS su $\mathbf{z}_1, \ldots, \mathbf{z}_q$:

$$\hat{\boldsymbol{\beta}}^{(q)} = \mathbf{V}\,\mathrm{diag}\Big(\frac{1}{d_1}, \ldots, \frac{1}{d_q}, 0, \ldots, 0\Big)\,\mathbf{U}^\top \mathbf{y}.$$

Confronto sintetico (usando la SVD):

$$\begin{aligned}
\text{OLS:} \quad & \hat{\boldsymbol{\beta}} = \mathbf{V}\,\mathrm{diag}\Big(\tfrac{1}{d_1}, \ldots, \tfrac{1}{d_p}\Big)\,\mathbf{U}^\top \mathbf{y}, \\
\text{Ridge:} \quad & \hat{\boldsymbol{\beta}}_\lambda = \mathbf{V}\,\mathrm{diag}\Big(\tfrac{d_1}{d_1^2+\lambda}, \ldots, \tfrac{d_p}{d_p^2+\lambda}\Big)\,\mathbf{U}^\top \mathbf{y}, \\
\text{PCR } (q): \quad & \hat{\boldsymbol{\beta}}^{(q)} = \mathbf{V}\,\mathrm{diag}\Big(\tfrac{1}{d_1}, \ldots, \tfrac{1}{d_q}, 0, \ldots, 0\Big)\,\mathbf{U}^\top \mathbf{y}.
\end{aligned}$$

Idea chiave: PCR applica una **soglia dura** (tiene i primi q e azzera gli altri); ridge applica una **contrazione continua** dipendente da d_j (mai azzera del tutto). Se l'informazione utile è concentrata nelle componenti con varianza grande (d_j grandi), **ridge funziona bene** perché "schiaccia" soprattutto le direzioni instabili (con d_j piccoli).

Take-away pratici

- La SVD consente di ottenere **tutte** le soluzioni ridge per **tutti** i λ in modo molto efficiente (riusando $\mathbf{U}, \mathbf{D}, \mathbf{V}$).
- In alta dimensionalità ($p \gg n$), ridge seleziona automaticamente la soluzione a norma minima e stabilizza il problema.
- I fattori $\frac{d_j^2}{d_j^2+\lambda}$ spiegano chiaramente il meccanismo: più piccola è la varianza spiegata da una direzione, maggiore sarà la contrazione su quella direzione.

5.2.6 Ridge: compromesso bias–varianza (Fixed-X)

Assumiamo il modello vero $\mathbf{y} = \mathbf{X}\beta + \varepsilon$ con $\varepsilon \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I}_n)$, $n > p$ e $\mathbf{X}$ a rango pieno. Il **ridge** stima

$$\hat{\beta}_\lambda = (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}_p)^{-1} \mathbf{X}^\top \mathbf{y}, \qquad \lambda \geq 0.$$

Operatore di restringimento

Definiamo

$$\mathbf{W}_\lambda = \Big(\mathbf{I}_p + \lambda (\mathbf{X}^\top \mathbf{X})^{-1}\Big)^{-1}.$$

Allora $\hat{\beta}_\lambda = \mathbf{W}_\lambda\, \hat{\beta}$ (cioè il ridge applica uno **shrinkage** alla stima OLS $\hat{\beta}$).

Bias (distorsione)

$$E[\hat{\beta}_\lambda] = \mathbf{X}^\top \mathbf{X}\,(\lambda \mathbf{I}_p + \mathbf{X}^\top \mathbf{X})^{-1} \beta, \qquad \Rightarrow \quad \mathrm{Bias}(\hat{\beta}_\lambda) = E[\hat{\beta}_\lambda] - \beta \neq \mathbf{0} \text{ se } \lambda > 0.$$

Limiti: $E[\hat{\beta}_0] = \beta$ (nessun bias per OLS) e $\lim_{\lambda \to \infty} E[\hat{\beta}_\lambda] = \mathbf{0}$ (tutto schiacciato verso 0).

Vista con la SVD. Se $\mathbf{X} = \mathbf{U}\mathbf{D}\mathbf{V}^\top$ (valori singolari $d_1 \geq \cdots \geq d_p > 0$),

$$\text{Bias}(\hat{\beta}_\lambda) = \mathbf{V}\,\text{diag}\Big(\tfrac{\lambda}{d_1^2+\lambda}, \ldots, \tfrac{\lambda}{d_p^2+\lambda}\Big)\mathbf{V}^\top\beta = \sum_{j=1}^{p} \mathbf{v}_j\,\frac{\lambda}{d_j^2+\lambda}\,\mathbf{v}_j^\top\beta.$$

Lettura: le direzioni con d_j piccolo (più "instabili") subiscono più bias.

Varianza

$$\text{Var}[\hat{\beta}_\lambda] = \sigma^2\,\mathbf{W}_\lambda(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{W}_\lambda^\top = \sigma^2\,\mathbf{V}\,\text{diag}\Big(\tfrac{d_1^2}{(d_1^2+\lambda)^2}, \ldots, \tfrac{d_p^2}{(d_p^2+\lambda)^2}\Big)\mathbf{V}^\top.$$

Confronto: per $\lambda = 0$, $\text{Var}[\hat{\beta}] = \sigma^2(\mathbf{X}^\top\mathbf{X})^{-1} \succeq \text{Var}[\hat{\beta}_\lambda]$; e $\lim_{\lambda\to\infty}\text{Var}[\hat{\beta}_\lambda] = \mathbf{0}$.

Intuizione SVD. Ogni direzione j è ridotta dal fattore $\dfrac{d_j^2}{(d_j^2+\lambda)^2}$: più piccolo è d_j, più forte è la riduzione di varianza.

MSE (errore quadratico medio) della stima

$$\text{MSE}(\hat{\beta}_\lambda) = E\Big[\|\hat{\beta}_\lambda - \beta\|_2^2\Big] = \text{tr}\big(\text{Var}[\hat{\beta}_\lambda]\big) + \|\text{Bias}(\hat{\beta}_\lambda)\|_2^2.$$

Trade-off: aumentando λ **cresce** il bias ma **diminuisce** la varianza. Spesso esiste un $\lambda > 0$ con MSE più basso di OLS ($\lambda = 0$), soprattutto quando alcune direzioni di $\mathbf{X}$ sono deboli (valori singolari piccoli).

Riassunto operativo

- Ridge "spalma" la stima lungo le direzioni principali di $\mathbf{X}$: fattore di contrazione per i fitted $\frac{d_j^2}{d_j^2+\lambda} \in (0, 1]$.
- Direzioni con d_j piccolo (alta instabilità numerica / multicollinearità): **tanta** riduzione di varianza ma **più** bias (utile per evitare overfitting).
- La scelta di λ (tipicamente via CV) bilancia bias e varianza per minimizzare l'errore.

5.2.7 Ridge migliora davvero OLS? (Teorema di Theobald, 1974)

Setting: Fixed-X, modello vero $\mathbf{y} = \mathbf{X}\beta + \varepsilon$ con $\varepsilon \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I}_n)$.

Enunciato (idea): Esiste sempre un $\lambda > 0$ tale che l'errore quadratico medio (MSE) sulla stima dei **parametri** è **minore** per ridge che per OLS:

$$\text{MSE}(\hat{\beta}_\lambda) = \underbrace{\|E[\hat{\beta}_\lambda] - \beta\|_2^2}_{\text{bias}^2} + \underbrace{\text{tr}\{\text{Var}(\hat{\beta}_\lambda)\}}_{\text{varianza}} \;<\; \text{MSE}(\hat{\beta}).$$

Intuizione: aumentando leggermente λ si introduce un po' di bias, ma la **forte** riduzione di varianza (soprattutto nelle direzioni "deboli"/mal condizionate di $\mathbf{X}$) più che compensa il nuovo bias.

Implicazione pratica. Anche se il modello lineare è corretto, una **piccola** penalizzazione può dare una stima dei coefficienti più accurata (MSE più basso) di OLS.

Perché non possiamo fissare subito il λ "ottimo"?

Il λ che minimizza l'MSE dipende da quantità sconosciute (β e σ^2). In pratica:

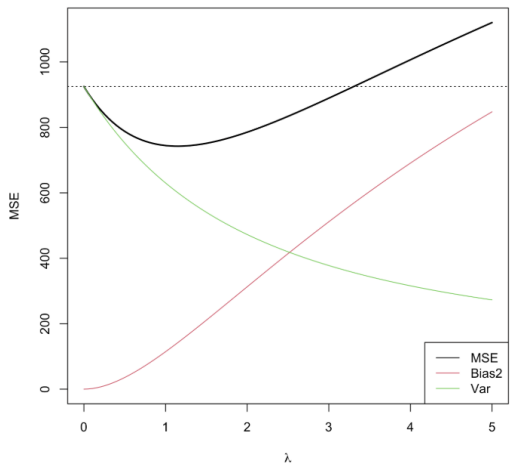
- si stima λ con **cross-validation** (minimizzando l'errore di previsione);
- oppure con criteri Bayesiani/empirici (interpretazione ridge = prior gaussiano su β).

Codice R: cosa fa il blocco proposto

1. **Dati** `diabetes`: costruisce $\mathbf{X}$ (predittori) e y (risposta).
2. **Standardizza $\mathbf{X}$** in $\mathbf{Z}$ per lavorare su scale comparabili (buona pratica per ridge).
3. **Stima** un $\hat{\beta}$ "proxy del vero" con OLS su $\mathbf{Z}$ centrata; stima anche $\hat{\sigma}^2$ (varianza del rumore).
4. **Funzione** `ridge_MSE`: usa la SVD di $\mathbf{X}$ per calcolare in chiuso, per ogni λ,

$$\mathrm{Bias}(\hat{\beta}_\lambda) = \mathbf{V}\,\mathrm{diag}\Big(\tfrac{\lambda}{d_j^2+\lambda}\Big)\mathbf{V}^\top\beta, \quad \mathrm{Var}(\hat{\beta}_\lambda) = \sigma^2\,\mathbf{V}\,\mathrm{diag}\Big(\tfrac{d_j^2}{(d_j^2+\lambda)^2}\Big)\mathbf{V}^\top,$$

   e poi l'MSE $= \mathrm{tr}(\mathrm{Var}) + \|\mathrm{Bias}\|^2$.
5. **Grafico**: traccia MSE (linea nera), Bias^2 (rosso) e Var (verde) al variare di λ, con la linea orizzontale al livello OLS ($\lambda = 0$).



Come leggere il grafico. Di solito la curva nera (MSE) scende sotto la linea tratteggiata di OLS per un intervallo di $\lambda > 0$: **esiste** quindi un λ che migliora OLS (confermando Theobald). A sinistra ($\lambda \approx 0$): varianza alta, bias quasi nullo. A destra (λ grande): varianza piccola, ma bias cresce e l'MSE risale.

Take-away operativi

- Ridge riduce la varianza dove $\mathbf{X}$ è mal condizionata (multicollinearità, $n \approx p$), spesso abbassando l'MSE dei **coefficienti**.
- La scelta di λ si fa in pratica con **CV** (obiettivo: errore di test minimo).
- Se il vero β è molto **sparso**, il **lasso** può dare vantaggi ulteriori (selezione di variabili).

5.2.8 Errore di previsione atteso

Quando usiamo la stima ridge $\hat{\beta}_\lambda$ per predire una nuova osservazione con predittori $\mathbf{x}_i$, la previsione è

$$\hat{y}_i = \mathbf{x}_i^\top \hat{\beta}_\lambda.$$

L'errore quadratico medio di previsione per il punto $\mathbf{x}_i$ è dato da:

$$\mathrm{MSE}(\hat{y}_i) = E\big[(\mathbf{x}_i^\top \hat{\beta}_\lambda - \mathbf{x}_i^\top \beta)^2\big] = \mathbf{x}_i^\top \mathrm{Var}(\hat{\beta}_\lambda)\mathbf{x}_i + \big[\mathbf{x}_i^\top \mathrm{Bias}(\hat{\beta}_\lambda)\big]^2.$$

Interpretazione

- Il primo termine $\mathbf{x}_i^\top \mathrm{Var}(\hat{\beta}_\lambda)\mathbf{x}_i$ misura quanto le previsioni variano da campione a campione.
- Il secondo termine $[\mathbf{x}_i^\top \mathrm{Bias}(\hat{\beta}_\lambda)]^2$ è il contributo dovuto al fatto che $\hat{\beta}_\lambda$ tende a sottostimare i coefficienti reali per effetto dello **shrinkage**.
- All'aumentare di λ, la varianza diminuisce ma il bias cresce.

Errore di previsione medio atteso

Considerando n punti indipendenti, l'errore quadratico medio atteso sulle previsioni è:

$$E\left[\frac{1}{n}\sum_{i=1}^{n}(y_i - y_i^*)^2\right] = \frac{1}{n}\sum_{i=1}^{n}\mathrm{MSE}(\hat{y}_i) + \sigma^2,$$

dove σ^2 è la varianza del rumore (che rappresenta il limite inferiore raggiungibile da qualsiasi metodo predittivo).

Lettura intuitiva

- L'**errore di previsione** si compone di tre parti:

$$\underbrace{\text{bias}^2}_{\text{modello troppo rigido}} + \underbrace{\text{varianza}}_{\text{modello troppo flessibile}} + \underbrace{\sigma^2}_{\text{rumore intrinseco}}.$$

- Nessuna scelta di λ può eliminare σ^2, ma scegliere λ bene permette di minimizzare il compromesso tra bias e varianza, ottenendo la miglior accuratezza predittiva possibile.

5.2.9 Matrice dei disegni ortonormale

Consideriamo il caso in cui la matrice del disegno $\mathbf{X}$ sia **ortonormale**, cioè

$$\mathbf{X}^\top \mathbf{X} = \mathbf{I}_p = (\mathbf{X}^\top \mathbf{X})^{-1}.$$

Ad esempio:

$$\mathbf{X} = \frac{1}{2} \begin{bmatrix} -1 & -1 & -1 & 1 \\ 1 & -1 & 1 & 1 \\ -1 & 1 & -1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}.$$

Stima ridge in questo caso

Se $\mathbf{X}$ è ortonormale, la stima ridge assume una forma molto semplice:

$$\hat{\beta}_\lambda = \frac{1}{1+\lambda}\, \hat{\beta},$$

cioè ogni coefficiente è ridotto di un fattore costante $\frac{1}{1+\lambda}$ rispetto a OLS. In altre parole, ridge applica una **contrazione uniforme** a tutti i coefficienti.

Varianza della stima ridge

Poiché i coefficienti sono ridotti di un fattore $\frac{1}{1+\lambda}$, la varianza si riduce quadraticamente:

$$\mathrm{Var}(\hat{\beta}_\lambda) = \frac{\sigma^2}{(1+\lambda)^2}\, \mathbf{I}_p.$$

Errore quadratico medio (MSE)

Combinando bias e varianza otteniamo:

$$\mathrm{MSE}(\hat{\beta}_\lambda) = \frac{p\sigma^2}{(1+\lambda)^2} + \frac{\lambda^2 \, \|\beta\|^2}{(1+\lambda)^2}.$$

Il valore di λ che minimizza l'MSE è:

$$\lambda^* = \frac{p\sigma^2}{\|\beta\|^2}.$$

Interpretazione intuitiva

- Quando $\mathbf{X}$ è ortonormale, tutte le direzioni dello spazio dei predittori hanno la stessa importanza statistica (stessa scala e stessa correlazione nulla).
- Ridge riduce in modo uniforme tutti i coefficienti, quindi il compromesso bias–varianza dipende solo da λ e non dalla struttura di $\mathbf{X}$.
- Il valore ottimale λ^* cresce con la varianza del rumore σ^2 e diminuisce con la dimensione del segnale $\|\beta\|^2$: più rumore $\Rightarrow$ più penalizzazione; segnale più forte $\Rightarrow$ meno shrinkage.

5.2.10 Ridge e leave-one-out cross validation

Per la validazione incrociata leave-one-out (LOO) esiste un risultato elegante che vale per la ridge regression e, più in generale, per gli stimatori lineari.

Matrice "hat" di ridge

La matrice dei proiettori (o **hat matrix**) per ridge è

$$H_\lambda = X\,(X^\top X + \lambda I_p)^{-1} X^\top.$$

Identità LOO

Sia $\widehat{\beta}^{\lambda}_{(-i)}$ lo stimatore ridge ottenuto escludendo l'osservazione i. Allora l'errore LOO può essere scritto senza rifittare n volte:

$$\mathrm{LOO}_\lambda = \frac{1}{n}\sum_{i=1}^{n}\Big[y_i - x_i^\top \widehat{\beta}^{\lambda}_{(-i)}\Big]^2 = \frac{1}{n}\sum_{i=1}^{n}\left[\frac{y_i - \hat{y}_i^{\lambda}}{1 - \{H_\lambda\}_{ii}}\right]^2,$$

dove $\hat{y}_i^{\lambda} = (H_\lambda y)_i$ e $\{H_\lambda\}_{ii}$ è l'elemento diagonale i-esimo di H_λ.

Calcolo rapido di H_λ via SVD

Se $X = UDV^\top$ è la SVD (con valori singolari $d_1 \geq \cdots \geq d_m \geq 0$, $m = \min\{n, p\}$), allora

$$H_\lambda = U\,\mathrm{diag}\left(\frac{d_1^2}{d_1^2 + \lambda}, \ldots, \frac{d_p^2}{d_p^2 + \lambda}\right) U^\top.$$

Questa forma consente di ottenere $\{H_\lambda\}_{ii}$ per molti valori di λ senza rifare i fit.

Idea chiave (perché funziona)

Se rimuovo (x_i, y_i) e rifitto, la previsione su x_i coincide con una correzione della previsione a campione intero:

$$\hat{y}_i^{\lambda} - y_i^{\star} = \frac{y_i - \hat{y}_i^{\lambda}}{1 - \{H_\lambda\}_{ii}},$$

dove $y_i^{\star} = x_i^\top \widehat{\beta}^{\lambda}_{(-i)}$. Questa identità porta alla formula compatta della LOO_λ sopra.

Generalized Cross-Validation (GCV)

Usando $\mathrm{tr}(H_\lambda) = \sum_{i=1}^{n}\{H_\lambda\}_{ii}$ e approssimando $\{H_\lambda\}_{ii} \approx \mathrm{tr}(H_\lambda)/n$, otteniamo

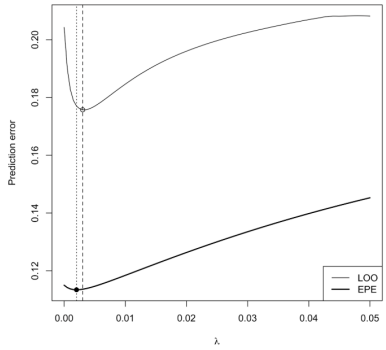
$$\mathrm{GCV}_\lambda = \frac{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i^{\lambda})^2}{\big(1 - \frac{1}{n}\mathrm{tr}(H_\lambda)\big)^2}.$$

La GCV evita di calcolare i n termini LOO e fornisce una stima efficiente dell'errore di previsione.

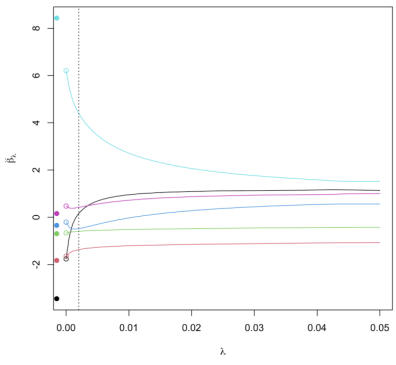
Codice R (ridge, LOO e GCV)

```
rm(list=ls())

library(glmnet)

y <- longley[, "Employed"]
X <- data.matrix(longley[, c(2:6,1)])
n <- nrow(X)
p <- ncol(X)

# Standardizzazione "population-like" delle colonne di X
Z <- scale(X, center = TRUE,
           scale = sqrt(diag(var(X) * (n-1)/n)))[,]

# Coefficienti OLS su Z (y centrata)
beta   <- matrix(coef(lm(I(y - mean(y)) ~ 0 + Z)), ncol = 1)
sigma2 <- summary(lm(I(y - mean(y)) ~ 0 + Z))$sigma^2

# EPE teorico per ridge (Fixed-X) via SVD
ridge_EPE <- function(X, beta, sigma2, lambda){
  n <- nrow(X); p <- ncol(X)
  beta <- matrix(beta, ncol = 1)
  SVD  <- svd(X); d <- SVD$d; U <- SVD$u; V <- SVD$v
  Bias <- V %*% diag(lambda/(d^2 + lambda)) %*% t(V) %*% beta
  Var  <- sigma2 * V %*% diag((d^2)/(d^2 + lambda)^2) %*% t(V)
  EPE  <- mean(apply(X, 1, function(x) t(x) %*% Var %*% x + (x %*% Bias)^2)) +
          sigma2
  return(EPE)
}

# Griglia di lambda
l <- seq(0, .05, length.out = 100)

set.seed(123)
# Simulo y da un modello lineare "vero" con varianza sigma2, poi centro
y <- Z %*% beta + rnorm(n, 0, sd = sqrt(sigma2))
y <- y - mean(y)

# LOO-CV esatta con glmnet (nfolds = n, grouped = FALSE)
cv_fit <- cv.glmnet(Z, y, alpha = 0, standardize = FALSE,
                    nfolds = n, grouped = FALSE, lambda = l)
l   <- cv_fit$lambda
fit <- glmnet(Z, y, alpha = 0, standardize = FALSE, lambda = l)

# Curve EPE (teorica) e LOO (stimata)
EPE <- sapply(l, function(lam) ridge_EPE(Z, beta, sigma2, lambda = lam))
LOO <- cv_fit$cvm
```

```
# Grafico errori di predizione
plot(l, EPE, xlab = expression(lambda), ylab = "Prediction error",
     type = "l", lwd = 2, ylim = c(min(EPE), max(LOO)))
lines(l, LOO)
legend("bottomright", c("LOO", "EPE"), lwd = 1:2)
abline(v = l[which.min(EPE)],  lty = 3)
abline(v = l[which.min(LOO)],  lty = 2)
points(l[which.min(LOO)], LOO[which.min(LOO)])
points(l[which.min(EPE)], EPE[which.min(EPE)], pch = 19)
```




```
# Tracce dei coefficienti ridge lungo il path di lambda
matplot(l, t(coef(fit)[-1,]), type = "l", lty = 1,
        xlab = expression(lambda),
        ylab = expression(hat(beta)[lambda]),
        col = 1:p, ylim = range(beta))
abline(v = l[which.min(EPE)], lty = 3)
points(rep(0, p), coef(fit)[-1, length(l)], col = 1:p)
points(rep(-.0015, p), beta, col = 1:p, pch = 19)
```



5.3 Applicazione: predizione di una variabile gaussiana

Consideriamo una variabile casuale gaussiana

$$Y = \beta + \varepsilon, \qquad \varepsilon \sim N(0, \sigma^2),$$

e un campione di training

$$Y_1, \ldots, Y_n.$$

Media campionaria e decomposizione bias–varianza

La media campionaria è

$$\bar{Y} = \frac{1}{n} \sum_{i=1}^{n} Y_i.$$

Per la previsione di una nuova osservazione Y, la decomposizione bias–varianza dà

$$E\big[(Y - \bar{Y})^2\big] = \sigma^2 + \big(\mathrm{Bias}(\bar{Y})\big)^2 + \mathrm{Var}(\bar{Y}) = \sigma^2 + 0 + \frac{\sigma^2}{n}.$$

Il teorema di Rao–Blackwell afferma che $\bar{Y}$ ha varianza minore di qualunque altro stimatore non distorto di β.

Domanda: la media campionaria $\bar{Y}$ è anche la **migliore** predizione per Y in termini di errore quadratico medio?

Media campionaria "shrunken"

Consideriamo un predittore "shrunken"

$$\hat{Y} = \lambda \bar{Y}, \qquad \lambda \in [0, 1].$$

La decomposizione bias–varianza diventa

$$E\big[(Y - \hat{Y})^2\big] = \sigma^2 + \big(\mathrm{Bias}(\hat{Y})\big)^2 + \mathrm{Var}(\hat{Y}) = \sigma^2 + (\beta - \lambda\beta)^2 + \lambda^2 \frac{\sigma^2}{n}.$$

Derivando rispetto a λ e ponendo uguale a zero,

$$\frac{\mathrm{d}}{\mathrm{d}\lambda} E\big[(Y - \hat{Y})^2\big] = 0 \quad \Rightarrow \quad \lambda = \frac{\beta^2}{\beta^2 + \sigma^2/n},$$

che minimizza l'errore quadratico medio $E\big[(Y - \hat{Y})^2\big]$.

Naturalmente, il valore ottimale di λ dipende da β e σ^2, che sono sconosciuti nella pratica.

Commenti

- Il fattore di shrinkage λ controlla quanto ci fidiamo dei dati rispetto a una sorta di **prior** implicita secondo cui la vera media β potrebbe essere vicina a zero.
- Se $\lambda = 1$ ci affidiamo completamente ai dati (nessuno shrinkage); se $\lambda = 0$ ignoriamo i dati e prediciamo sempre 0.

- Per valori intermedi di λ introduciamo bias ma riduciamo la varianza: possiamo ottenere un MSE più basso rispetto a $\bar{Y}$, pur essendo $\bar{Y}$ non distorto. Questo illustra il classico compromesso bias–varianza alla base della ridge regression e di molti metodi di regolarizzazione.

Questo esempio è un caso unidimensionale di ridge regression: il termine di penalizzazione λ ha l'effetto di restringere lo stimatore verso zero, migliorando la performance predittiva quando la varianza è elevata rispetto all'intensità del segnale.

Esempio simulato: sottoinsieme di coefficienti piccoli

Consideriamo ora un modello lineare multivariato, in cui alcuni coefficienti sono grandi e altri sono piccoli; vogliamo confrontare regressione lineare semplice (OLS) e ridge regression in termini di bias, varianza ed errore di previsione.

Impostiamo:

$$n = 50, \qquad p = 30, \qquad \sigma^2 = 1.$$

Il vettore vero dei coefficienti $\beta^\star$ ha 10 componenti grandi (tra 0.5 e 1) e 20 piccole (tra 0 e 0.3). Indichiamo con $\mu = X\beta^\star$ il valore atteso del modello lineare.

Codice R

```
library(MASS)

set.seed(0)
n <- 50
p <- 30

# Matrice di design con covariate indipendenti N(0,1)
x <- matrix(rnorm(n * p), nrow = n)

# Coefficienti veri: 10 grandi, 20 piccoli
bstar <- c(runif(10, 0.5, 1),
           runif(20, 0, 0.3))

# Media vera del modello
mu <- as.numeric(x %*% bstar)

# ------------------------------------------------
# Confronto tra regressione lineare e ridge
# ------------------------------------------------

set.seed(1)
R    <- 100          # numero di repliche
nlam <- 60           # numero di valori di lambda
lam  <- seq(0, 25, length = nlam)

beta.ls  <- matrix(0, R, p)
beta.rid <- array(0, dim = c(R, nlam, p))
```

```
fit.ls   <- matrix(0, R, n)
fit.rid  <- array(0, dim = c(R, nlam, n))

err.ls   <- numeric(R)
err.rid  <- matrix(0, R, nlam)

for (i in 1:R) {
  # y di training e ynew di test (stessa mu, nuovo rumore)
  y    <- mu + rnorm(n, sd = 1)
  ynew <- mu + rnorm(n, sd = 1)

  # Regressione lineare senza intercetta
  a   <- lm(y ~ 0 + x)
  bls <- coef(a)

  beta.ls[i, ] <- bls
  fit.ls[i, ]  <- x %*% bls
  err.ls[i]    <- mean((ynew - fit.ls[i, ])^2)

  # Ridge regression per una griglia di lambda
  aa   <- lm.ridge(y ~ 0 + x, lambda = lam)
  brid <- coef(aa)

  beta.rid[i, , ] <- brid
  fit.rid[i, , ]  <- brid %*% t(x)

  # Errore di previsione ridge su ynew
  err.rid[i, ] <- rowMeans(
    scale(fit.rid[i, , ], center = ynew, scale = FALSE)^2
  )
}

# Medie sugli R esperimenti
aveerr.ls  <- mean(err.ls)
aveerr.rid <- colMeans(err.rid)

# Bias e varianza di OLS
bias.ls <- sum((colMeans(fit.ls) - mu)^2) / n
var.ls  <- sum(apply(fit.ls, 2, var)) / n

# Bias e varianza di ridge lungo il path di lambda
bias.rid <- rowSums(
  scale(apply(fit.rid, 2:3, mean),
        center = mu, scale = FALSE)^2
) / n
var.rid <- rowSums(apply(fit.rid, 2:3, var)) / n

# MSE (sul valore atteso mu) e errore di previsione (aggiungo sigma^2 = 1)
mse.ls      <- bias.ls + var.ls
```

```
mse.rid     <- bias.rid + var.rid
prederr.ls  <- mse.ls + 1
prederr.rid <- mse.rid + 1

# Grafico: errore di previsione vs. amount of shrinkage
par(mar = c(4.5, 4.5, 0.5, 0.5))
plot(lam, prederr.rid, type = "l",
     xlab = "Amount of shrinkage", ylab = "Prediction error")
abline(h = prederr.ls, lty = 2)
text(c(1, 24), c(1.48, 1.48), c("Low", "High"))
legend("topleft", lty = c(2, 1),
       legend = c("Linear regression", "Ridge regression"))
```

Risultati (scenario con coefficienti piccoli)

Nel caso con 10 coefficienti grandi e 20 piccoli:

- La regressione lineare ha bias al quadrato ≈ 0.006 e varianza ≈ 0.627, quindi errore di previsione
$$\sigma^2 + (\text{Bias})^2 + \text{Var} \approx 1 + 0.006 + 0.627 \approx 1.633.$$
- La ridge regression (per un λ ottimale scelto tramite simulazione) ha bias al quadrato ≈ 0.077 e varianza ≈ 0.403, con errore di previsione
$$\approx 1 + 0.077 + 0.403 \approx 1.48,$$
quindi migliore di OLS.

L'esempio mostra che, anche se introduciamo bias (shrinkage dei coefficienti), la riduzione della varianza può essere sufficiente a ridurre l'errore di previsione totale.

Altri scenari: coefficienti moderati e coefficienti nulli

Coefficienti tutti moderatamente grandi. Se ripetiamo l'esperimento mantenendo la stessa struttura ($n = 50$, $p = 30$, $\sigma^2 = 1$) ma ponendo **tutti** i coefficienti in un intervallo moderatamente grande (ad esempio tra 0.5 e 1), i valori di bias, varianza ed errore di previsione per la regressione lineare rimangono sostanzialmente invariati. La ridge regression può ancora ottenere un MSE leggermente inferiore, ma soltanto per valori relativamente piccoli di λ (ad esempio $\lambda 5$); per shrinkage più forte il bias diventa dominante.

Sottoinsieme di coefficienti esattamente nulli. All'estremo opposto, se 10 coefficienti sono grandi (tra 0.5 e 1) e i restanti 20 sono esattamente uguali a zero, abbiamo un tipico problema di **selezione di variabili**. Anche qui la regressione lineare mantiene circa lo stesso bias e la stessa varianza di prima, mentre la ridge regression migliora l'MSE grazie allo shrinkage. Tuttavia:

- I coefficienti corrispondenti alle variabili irrilevanti (quelle con vero coefficiente zero) vengono solo **shrunken** ma non portati esattamente a zero.
- Di conseguenza, ridge regression non effettua vera selezione di variabili: tutti i predittori restano nel modello, anche se con coefficienti molto piccoli.

Questo mette in luce un limite interpretativo di ridge: è molto efficace per ridurre la varianza e migliorare la previsione, ma non è adatto quando l'obiettivo principale è individuare un sottoinsieme ristretto di variabili davvero rilevanti (per quello serviranno metodi come il lasso o procedure di selezione di modello).

Chapter 6

Best subset selection

6.1 Principio *bet on sparsity*

Nel contesto dell'alta dimensionalità è ragionevole assumere che solo un numero ristretto di predittori abbia un effetto reale sulla risposta, cioè che la maggior parte dei coefficienti veri sia nulla. Questa idea è spesso riassunta nel principio

"Bet on sparsity"

e si traduce nell'ipotesi che lo stimatore dei coefficienti $\hat{\beta}$ sia **sparso**, ossia che molti componenti $\hat{\beta}_j$ siano esattamente uguali a zero.

La ridge regression produce uno stimatore

$$\hat{\beta}_\lambda = \arg\min_\beta \Big\{ \|y - X\beta\|_2^2 + \lambda \|\beta\|_2^2 \Big\},$$

che in generale non è sparso: tutti i coefficienti vengono solo **shrinkati** verso zero, ma raramente diventano esattamente nulli. I metodi di **variable selection**, invece, usano i dati per decidere quali predittori devono avere $\hat{\beta}_j \neq 0$ e quali invece possono essere esclusi dal modello.

6.2 Modello lineare e guadagno potenziale della sparsità

Consideriamo il modello lineare

$$y = X\beta_0 + \varepsilon, \qquad E(\varepsilon) = 0, \quad \mathrm{Var}(\varepsilon) = \sigma^2 I_n,$$

dove X è una matrice $n \times p$ di predittori e β_0 è il vettore dei coefficienti veri.

In molti dataset moderni ci sono forti motivazioni per ritenere che solo una parte dei predittori sia effettivamente rilevante. Definiamo quindi l'insieme degli indici dei predittori realmente attivi

$$S = \{k \in \{1, \dots, p\} : \beta_{0k} \neq 0\},$$

e poniamo $s = |S| \ll p$.

La stima OLS classica

$$\hat{\beta}_{\mathrm{OLS}} = (X^\top X)^{-1} X^\top y$$

ha errore quadratico medio di previsione

$$\frac{1}{n} E\|X\beta_0 - X\hat{\beta}_{\text{OLS}}\|_2^2.$$

Sviluppando:

$$\begin{aligned}
\frac{1}{n} E\|X\beta_0 - X\hat{\beta}_{\text{OLS}}\|_2^2 &= \frac{1}{n} E\left\{(\beta_0 - \hat{\beta}_{\text{OLS}})^\top X^\top X(\beta_0 - \hat{\beta}_{\text{OLS}})\right\} \\
&= \frac{1}{n} E\left[\text{tr}\left\{(\beta_0 - \hat{\beta}_{\text{OLS}})(\beta_0 - \hat{\beta}_{\text{OLS}})^\top X^\top X\right\}\right] \\
&= \frac{1}{n} \text{ tr}\left[E\left\{(\beta_0 - \hat{\beta}_{\text{OLS}})(\beta_0 - \hat{\beta}_{\text{OLS}})^\top\right\} X^\top X\right] \\
&= \frac{1}{n} \text{ tr}\left[\text{Var}(\hat{\beta}_{\text{OLS}})\, X^\top X\right].
\end{aligned}$$

Poiché

$$\text{Var}(\hat{\beta}_{\text{OLS}}) = \sigma^2 (X^\top X)^{-1},$$

segue che

$$\frac{1}{n} E\|X\beta_0 - X\hat{\beta}_{\text{OLS}}\|_2^2 = \frac{1}{n} \text{ tr}\left[\sigma^2 I_p\right] = \frac{p}{n}\sigma^2.$$

Se riuscissimo a conoscere l'insieme S e stimare un modello lineare che utilizza solo le colonne di X corrispondenti a tali variabili, otterremmo analogamente

$$\frac{1}{n} E\|X_S\beta_{0,S} - X_S\hat{\beta}_{\text{OLS},S}\|_2^2 = \frac{s}{n}\sigma^2,$$

cioè un errore di previsione pari a $s\sigma^2/n$ invece di $p\sigma^2/n$.

Dunque:

- il modello più piccolo che coinvolge solo le variabili in S avrebbe un errore di previsione minore;
- gli stimatori dei parametri sarebbero più accurati;
- il modello sarebbe anche molto più semplice da interpretare.

Questo motiva la ricerca dell'insieme S tramite procedure di selezione di variabili.

6.3 Idea di base del best subset selection

Un approccio naturale consiste nel considerare tutti i possibili sottoinsiemi di predittori e scegliere il modello "migliore" in base a un opportuno criterio.

Per ogni sottoinsieme $M \subset \{1, \ldots, p\}$, indichiamo con X_M la matrice che contiene solo le colonne di X corrispondenti agli indici in M. Il numero totale di sottoinsiemi possibili è

$$2^p,$$

quindi ci sono 2^p modelli di regressione lineare della forma

$$y = X_M\beta_M + \varepsilon.$$

L'idea del **best subset selection** è:

- stimare tutti i modelli associati ai possibili sottoinsiemi M;
- confrontarli tramite un criterio di bontà (RSS, AIC, BIC, CV, ...);
- scegliere il sottoinsieme che ottimizza tale criterio.

Per matrici di progetto generali, ciò richiede una **ricerca esaustiva** su tutti i sottoinsiemi, e diventa rapidamente impraticabile quando p è anche solo moderatamente grande (tipicamente $p50$).

6.4 Algoritmo di ricerca per dimensione del modello

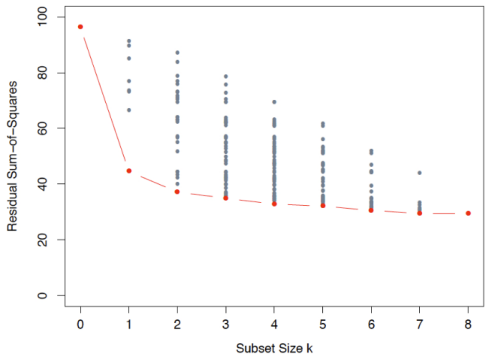
Un modo più strutturato per implementare il best subset selection è procedere per dimensione del modello:

1. Si parte dal **modello nullo** B_0, che contiene solo l'intercetta.
2. Per $k = 1, \dots, p$:
   (a) si considerano tutti i modelli che contengono esattamente k predittori; il loro numero è
   $$\binom{p}{k};$$
   (b) per ciascuno di questi modelli si stima la regressione lineare e si calcola la **residual sum of squares** (RSS);
   (c) si sceglie, tra i $\binom{p}{k}$ modelli, quello con RSS minima; lo si denota con B_k.
3. Alla fine si ottiene una collezione ordinata di modelli
   $$B_0, B_1, \dots, B_p,$$
   uno per ogni dimensione k.
4. Si seleziona il modello "finale" tra $B_0, \dots, B_p$ usando un criterio di scelta del modello, ad esempio AIC, BIC, C_p, oppure un errore di previsione stimato tramite cross-validation.

Osserviamo che, anche organizzando la ricerca per dimensione del modello, il costo computazionale complessivo rimane dell'ordine di 2^p, poiché

$$\sum_{k=0}^{p} \binom{p}{k} = 2^p.$$

Per valori di p moderati (ad esempio $p \leq 20$) questo è fattibile; per p più grandi è necessario ricorrere a strategie approssimate (forward/backward stepwise, lasso, ecc.) che realizzano un compromesso tra costo computazionale e qualità della soluzione.



6.5 Stepwise selection

6.5.1 Backward stepwise selection

La *backward stepwise selection* può essere vista come una versione "greedy" del *best subset selection*. A differenza della ricerca esaustiva, che valuta tutti i 2^p possibili modelli, lo stepwise considera solo una sequenza ristretta di modelli ottenuti rimuovendo una variabile alla volta.

Questo approccio è molto più efficiente dal punto di vista computazionale, ma inevitabilmente sub-ottimale rispetto al best subset selection.

Notiamo inoltre che la backward selection è applicabile solo quando $n > p$, perché richiede di stimare inizialmente il modello completo.

Algoritmo

1. Si parte dal modello completo
$$S_p = \{1, \dots, p\}.$$
2. Per $k = p, p-1, \dots, 1$:
   (a) si considerano tutti i modelli che contengono $k-1$ predittori, ottenuti rimuovendo uno dei k predittori in S_k;
   (b) si valutano i k modelli così ottenuti;
   (c) si seleziona il "migliore" tra questi modelli, dove "migliore" significa avere il valore minimo del **residual sum of squares** (RSS) oppure il miglior valore secondo AIC, BIC, CV, ecc.;
   (d) si denota tale modello con S_{k-1}.
3. Si ottiene quindi una sequenza di modelli
$$S_p, S_{p-1}, \dots, S_1, S_0,$$
   ciascuno con un numero decrescente di predittori.
4. Infine, si sceglie il miglior modello tra questi usando AIC, BIC, o cross-validation.

Perché la backward selection è greedy e sub-ottimale?

La backward stepwise selection parte dal modello completo e rimuove una variabile alla volta. A ogni passo viene scelta la variabile la cui rimozione produce il miglior miglioramento (o il minor peggioramento) del criterio di bontà del modello.

Questo comportamento è **greedy** perché:

- considera solo la **migliore decisione locale** a ogni passo;
- non valuta le conseguenze future della scelta (ad esempio come la rimozione influenzi le scelte successive);
- esplora solo una piccolissima parte dei 2^p modelli possibili.

Il best subset selection, invece, valuta **tutti** i possibili modelli e individua quindi il sottoinsieme globalmente ottimale.

Di conseguenza:

- la backward stepwise selection può rimanere bloccata in un ottimo locale;
- può non recuperare il modello globalmente migliore;
- tende a fornire soluzioni meno accurate rispetto al best subset selection.

In sintesi:

Backward stepwise è veloce ma può perdere il modello ottimale.
Best subset è ottimale ma troppo costoso per p grandi.

6.5.2 Forward stepwise selection

La *forward stepwise selection* è un algoritmo greedy, computazionalmente molto efficiente, ma sub-ottimale rispetto al *best subset selection*. A differenza dell'approccio backward, qui si parte dal modello nullo e si aggiungono predittori uno alla volta.

Un grande vantaggio è che può essere applicato anche quando $p > n$, perché non richiede di stimare il modello completo.

Produce una sequenza di modelli:

$$S_0,\ S_1,\ S_2,\ \ldots,\ S_{n-1},$$

ciascuno dei quali aggiunge un predittore al modello precedente.

Algoritmo

1. Si parte dal modello nullo:
$$S_0 = .$$
2. Per $k = 0, \ldots, \min(n-1,\, p-1)$:
   (a) si considerano tutti i $p - k$ modelli che aggiungono un singolo predittore a quelli già presenti in S_k;
   (b) si stima ciascuno di questi modelli;

(c) si sceglie il modello "migliore" secondo RSS, AIC, BIC, o cross-validation, e si definisce:

$$S_{k+1}.$$

3. Si seleziona infine un modello dalla sequenza

$$S_0,\ S_1,\ S_2,\ \ldots$$

usando un criterio di selezione (AIC, BIC, CV, ecc.).

Perché la forward stepwise selection è greedy e sub-ottimale?

La forward selection inizia dal modello nullo e aggiunge un predittore alla volta. A ogni passo sceglie il predittore che dà il miglior miglioramento immediato del criterio scelto (RSS, AIC, BIC,...).

Questo comportamento è **greedy** perché:

- ottimizza solo a livello locale, migliorando il modello un passo alla volta;
- prende decisioni che non considerano gli effetti futuri (una variabile non scelta inizialmente potrebbe diventare utile solo se combinata con altre);
- esplora una porzione estremamente ridotta dello spazio dei modelli.

Infatti, considera solo:

- 1 modello di dimensione 0,
- al più p modelli di dimensione 1,
- al più $p-1$ modelli di dimensione 2,
- ecc.

Invece, il best subset selection valuta tutti i modelli:

$$\binom{p}{k}$$

per ogni k, esplorando l'intero spazio dei sottoinsiemi.

Di conseguenza, la forward selection può:

- ignorare il sottoinsieme ottimale,
- restare intrappolata in soluzioni sub-ottimali,
- produrre modelli meno accurati rispetto al best subset selection.

In sintesi:

Forward stepwise è rapido ma rischia di perdere la combinazione ottimale di predittori.
Best subset è ottimale ma troppo costoso per p elevati.

6.5.3 Forward selection with AIC-based stopping rule

L'idea è simile alla forward stepwise selection, ma il processo si arresta automaticamente quando l'AIC smette di migliorare. In questo modo si evita di costruire l'intera sequenza di modelli fino a dimensione $\min(n-1, p-1)$.

Algoritmo

1. Si parte dal modello nullo:
$$S_0 =, \qquad k = 0.$$
2. Si considerano tutti i $p-k$ modelli che aggiungono un singolo predittore ai predittori già presenti in S_k:
$$S_k \cup \{j\}, \qquad j \in \{1, \dots, p\} \setminus S_k.$$
3. Si stima ognuno di questi modelli e si sceglie il migliore secondo il criterio AIC, definendo:
$$S_{k+1} = \arg \min_{\text{modelli } M} \text{AIC}(M).$$
4. Si confrontano i valori di AIC:
$$\text{AIC}(S_{k+1}) < \text{AIC}(S_k).$$
   - Se la disuguaglianza è vera, si aggiorna $k \leftarrow k+1$ e si ritorna al passo 2.
   - Se non è soddisfatta, **l'algoritmo si ferma**.

Interpretazione

L'algoritmo procede finché aggiungere una nuova variabile **migliora** il compromesso tra bontà dell'adattamento e complessità del modello, misurato dall'AIC. Quando l'AIC inizia ad aumentare, significa che una variabile in più non giustifica la maggiore complessità, e il processo si interrompe automaticamente.

In questo modo l'algoritmo:

- evita l'overfitting,
- costruisce modelli più compatti rispetto alla forward stepwise pura,
- è computazionalmente efficiente,
- è più interpretabile perché produce un punto di arresto naturale basato sul criterio statistico.

6.6 Variable selection and cross-validation

La selezione delle variabili è parte integrante del processo di costruzione del modello. Per questo motivo, quando utilizziamo la cross-validation (CV) per stimare l'errore di previsione, dobbiamo prestare particolare attenzione a **quando** avviene la selezione dei predittori.

Best subset selection *fuori* dalla cross-validation

Una strategia ingenua è:

1. eseguire la best subset selection sull'intero dataset, ottenendo i modelli

$$B_0, B_1, \ldots, B_p;$$

2. usare la cross-validation per scegliere la dimensione ottimale k e stimare l'errore di previsione.

Tuttavia questa procedura è scorretta. Il problema è che il **training set viene usato per tre scopi contemporaneamente**:

1. selezionare i predittori,
2. stimare i coefficienti,
3. valutare le prestazioni (tramite CV).

Se il numero di osservazioni non è molto grande rispetto al numero di predittori, il training set non è sufficiente per svolgere correttamente tutti questi compiti contemporaneamente. Di conseguenza, le stime dell'errore ottenute dalla cross-validation sono troppo ottimistiche e **non rappresentano accuratamente l'errore su un vero test set**.

Perché questa procedura è sbagliata

Se la best subset selection viene eseguita sull'intero dataset, allora la cross-validation non riflette la vera variabilità del processo di selezione. Di fatto, la CV valuta un modello già selezionato su tutti i dati, quindi ignora la parte più instabile del processo: **la scelta del sottoinsieme di variabili**.

Poiché la selezione delle variabili è parte del processo di costruzione del modello, **la cross-validation deve tener conto della sua variabilità**.

Best subset selection *dentro* la cross-validation

La soluzione corretta è:

> **Eseguire la best subset selection separatamente dentro ogni iterazione della cross-validation.**

Questo significa che per ogni fold:

1. si considera solo il training del fold,
2. si esegue da zero la best subset selection su quel training,
3. si stima il modello di ogni dimensione,
4. si valuta ogni modello sul validation fold.

Alla fine, raccogliamo gli errori ottenuti per ogni dimensione k e scegliamo la dimensione che minimizza l'errore medio di cross-validation.

Importante: non esiste un unico "miglior" sottoinsieme

Quando si effettua ricerca di subset, bisogna ricordare che:

> **possono esistere più sottoinsiemi diversi che producono lo stesso livello di performance.**

Pertanto, durante la cross-validation, è normale che a ciascuna iterazione emergano subset differenti:

$$B_k^{(1)},\ B_k^{(2)},\ \ldots,\ B_k^{(K)},$$

anche se il valore di k è lo stesso.

Questo è un comportamento normale e riflette la variabilità intrinseca del processo di selezione delle variabili.

Messaggio chiave

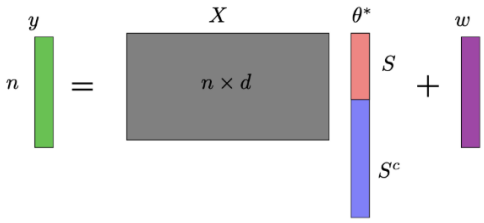
> **Per ottenere una stima corretta dell'errore di test, la selezione delle variabili deve essere rifatta da zero in ogni iterazione della cross-validation.**

Ignorare questo principio porta a valutazioni troppo ottimistiche e potenzialmente fuorvianti delle prestazioni del modello.

Chapter 7

Lasso regression

7.1 Sparse linear regression



7.1.1 Another look at best subset

Quando il segnale è realmente sparso, la best subset regression può sembrare la scelta più naturale: se solo poche variabili hanno un effetto, perché non cercare direttamente il sottoinsieme migliore?

Ricordiamo però che la best subset può essere vista come il problema di trovare il vettore dei coefficienti che minimizza l'errore di regressione, imponendo che **al più** s coefficienti siano diversi da zero. In forma matematica:

$$\min_{\beta \in R^p} \|y - X\beta\|_2^2 \qquad \text{s.t.} \qquad \|\beta\|_0 \leq s,$$

dove, con un abuso di terminologia, definiamo la "norma" ℓ_0 come:

$$\|\beta\|_0 = \sum_{j=1}^{p} \mathbf{1}\{\beta_j \neq 0\},$$

cioè il numero di coefficienti non nulli nel vettore β.

Problema: la non convessità. Il vincolo $\|\beta\|_0 \leq s$ rende il problema non convesso. Questo significa che per risolverlo dobbiamo considerare **tutte** le combinazioni di s predittori su p disponibili.

Il numero di modelli da valutare è:

$$\binom{p}{s} = \frac{p!}{(p-s)!\,s!},$$

che cresce in modo combinatoriale con p. Per esempio, con $p = 40$ e $s = 10$, abbiamo:

$$\binom{40}{10} = 847{,}660{,}528.$$

Questo rende la best subset impraticabile già per valori moderati di p.

Motivazione per metodi alternativi. La non convessità — e quindi l'impossibilità di risolvere efficientemente il problema ℓ_0 — è esattamente uno dei motivi principali per cui nasce la regressione lasso.

L'idea chiave:

rimpiazzare il vincolo $\|\beta\|_0 \leq s$ *con una penalizzazione convessa* $\|\beta\|_1$.

Questo porta a un problema ottimizzabile in modo efficiente, pur incoraggiando soluzioni sparse.

Le sezioni successive sviluppano proprio questa idea.

7.1.2 Another formulation for ridge regression

La regressione ridge può essere vista da due punti di vista: (1) come un problema penalizzato, (2) come un problema con vincolo.

Entrambe le formulazioni sono equivalenti.

1. Formulazione penalizzata (standard):

$$\min_{\beta \in R^p} \quad \|y - X\beta\|_2^2 + \lambda \|\beta\|_2^2,$$

dove $\lambda \geq 0$ controlla la quantità di shrinkage: più grande è λ, più piccoli diventano i coefficienti.

Indichiamo con $\hat{\beta}_\lambda^R$ la soluzione di questo problema.

2. Formulazione vincolata:

Sappiamo che la stessa soluzione $\hat{\beta}_\lambda^R$ risolve anche:

$$\min_{\beta \in R^p} \quad \|y - X\beta\|_2^2 \quad \text{s.t.} \quad \|\beta\|_2^2 \leq \|\hat{\beta}_\lambda^R\|_2^2.$$

In generale, ridge equivale a risolvere:

$$\min_{\beta \in R^p} \quad \|y - X\beta\|_2^2 \quad \text{s.t.} \quad \|\beta\|_2^2 \leq s,$$

per un opportuno parametro $s \geq 0$.

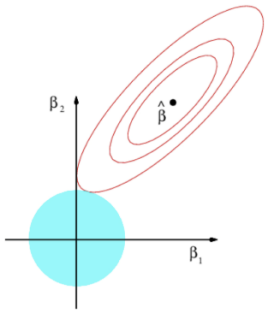
Equivalenza tra i due problemi

- Per ogni $\lambda \geq 0$ esiste un valore $s \geq 0$ tale che $\hat{\beta}_\lambda^R$ è anche soluzione del problema vincolato.

- Per ogni $s \geq 0$ esiste un $\lambda \geq 0$ tale che la soluzione del problema vincolato coincide con quella penalizzata.

Questa equivalenza è alla base dell'interpretazione geometrica della regressione ridge: le ellissi di livello dell'errore quadratico $\|y - X\beta\|_2^2$ intersecano la palla euclidea $\|\beta\|_2^2 \leq s$.

Le ellissi (rosse) rappresentano i livelli di $\|y - X\beta\|_2^2$; il vincolo $\beta_1^2 + \beta_2^2 \leq s$ (regione blu) impone che la soluzione rimanga vicino all'origine. Ridge "spinge" i coefficienti verso zero, ma **non** li annulla mai completamente.



Tre norme

Consideriamo ora tre norme fondamentali per descrivere la sparsità o la dimensione di un vettore β:

$$\|\beta\|_0 = \sum_{j=1}^{p} \mathbf{1}\{\beta_j \neq 0\}, \qquad \|\beta\|_1 = \sum_{j=1}^{p} |\beta_j|, \qquad \|\beta\|_2 = \left(\sum_{j=1}^{p} \beta_j^2 \right)^{1/2}.$$

La norma ℓ_0 **non è una vera norma** (non soddisfa proprietà di subadditività), ma è un conteggio del numero di coefficienti non nulli: è ciò che vorremmo minimizzare per ottenere un modello sparso.

Le tre norme sono casi particolari della norma ℓ_q:

$$\|\beta\|_q = \left(\sum_{j=1}^{p} |\beta_j|^q \right)^{1/q}.$$

- ℓ_2: induce shrinkage (ridge), ma non seleziona variabili.
- ℓ_1: induce soluzioni sparse (lasso), ponendo molti $\beta_j = 0$.
- ℓ_0: selezione perfetta, ma problema non convesso e computazionalmente impossibile.

Questa distinzione prepara il terreno all'introduzione del lasso, che usa la norma ℓ_1 come compromesso tra la sparsità ideale della norma ℓ_0 e la trattabilità della norma ℓ_2.

7.1.3 Penalized and constrained problems

La scelta della norma nella funzione di penalizzazione determina il tipo di regressione che otteniamo. Consideriamo tre penalizzazioni fondamentali: ℓ_0, ℓ_1, ℓ_2.

Formulazione penalizzata

Usare le norme ℓ_0, ℓ_1 e ℓ_2 porta ai seguenti problemi di ottimizzazione:

Best subset selection

$$\min_{\beta \in R^p} \quad \|y - X\beta\|_2^2 + \lambda \|\beta\|_0$$

Lasso regression

$$\min_{\beta \in R^p} \quad \|y - X\beta\|_2^2 + \lambda \|\beta\|_1$$

Ridge regression

$$\min_{\beta \in R^p} \quad \|y - X\beta\|_2^2 + \lambda \|\beta\|_2^2$$

dove $\lambda \geq 0$ è il parametro di tuning.

Formulazione vincolata

Le stesse penalizzazioni possono essere interpretate come vincoli:

Best subset selection

$$\min_{\beta \in R^p} \quad \|y - X\beta\|_2^2 \quad \text{s.t.} \quad \|\beta\|_0 \leq k$$

Lasso regression

$$\min_{\beta \in R^p} \quad \|y - X\beta\|_2^2 \quad \text{s.t.} \quad \|\beta\|_1 \leq s$$

Ridge regression

$$\min_{\beta \in R^p} \quad \|y - X\beta\|_2^2 \quad \text{s.t.} \quad \|\beta\|_2^2 \leq s$$

dove k e s sono parametri di tuning.

Interpretazione dei parametri

Nel caso ℓ_0, il parametro naturale è k (un intero): stiamo cercando il miglior sottoinsieme di k variabili in termini di errore di training.

Per lasso e ridge, invece, scegliere s o scegliere λ è equivalente: le due formulazioni portano allo stesso insieme di soluzioni.

Equivalenza (lasso e ridge)

Per le penalizzazioni ℓ_1 e ℓ_2 vale:

- per ogni $s \geq 0$ e ogni soluzione $\hat{\beta}$ del problema vincolato esiste un $\lambda \geq 0$ tale che $\hat{\beta}$ è anche soluzione del problema penalizzato; - e viceversa: ogni soluzione del problema penalizzato corrisponde a un valore s.

Questa equivalenza permette di passare liberamente tra interpretazione "penalizzata" e interpretazione "vincolata".

Non equivalenza (best subset)

Per la norma ℓ_0 l'equivalenza **non** vale:

- per ogni $\lambda \geq 0$ e ogni soluzione del problema penalizzato esiste un k tale che la stessa soluzione risolve il problema vincolato; - **ma non vale il viceversa**: una soluzione del problema vincolato per un dato k **non** è in generale ottenibile da nessun valore di λ.

Questo riflette la natura profondamente non convessa della penalizzazione ℓ_0.

7.1.4 Signal sparsity

L'ipotesi di **signal sparsity** afferma che solo un numero "piccolo" di predittori abbia realmente effetto sulla risposta, cioè:

$$\beta_j \neq 0 \quad \text{solo per pochi } j.$$

L'obiettivo è ottenere un estimatore $\hat{\beta}$ che sia esso stesso **sparso**, cioè con la maggior parte dei coefficienti esattamente nulli.

- L'estimatore ridge **non** è sparso: tutti i coefficienti vengono solo "shrinkati" verso zero, ma mai portati esattamente a zero.
- Gli stimatori ottenuti tramite **best subset selection** e **lasso** sono invece sparsi: forzano molti coefficienti a diventare esattamente zero.

Perché la norma ℓ_1 induce sparsità, mentre la norma ℓ_2 no?

La risposta si vede esaminando le regioni di vincolo dei due metodi.

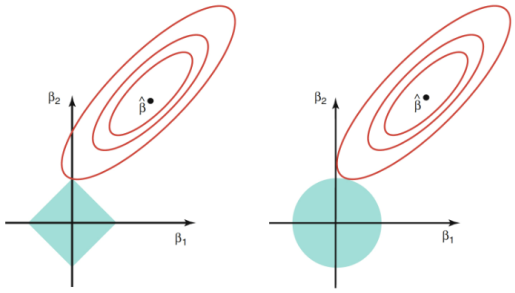
- Il vincolo del ridge è:
$$\|\beta\|_2^2 \leq s,$$
che rappresenta una **sfera (o ellissoide)** nello spazio dei coefficienti.
- Il vincolo del lasso è:
$$\|\beta\|_1 \leq s,$$
che rappresenta un **poliedro a forma di rombo (ottagono in dimensione 3, poliedro con "spigoli" in generale)**.

Lasso e ridge trovano la soluzione nel punto di contatto tra la regione di vincolo e le ellissi della funzione di perdita (RSS).

Geometria intuitiva

- Le ellissi del RSS tendono a "toccare" il vincolo del lasso **nei vertici** del poliedro.
- I vertici del vincolo ℓ_1 corrispondono a punti in cui molti coefficienti sono esattamente nulli.
- Il vincolo ℓ_2 è invece liscio e rotondo: non ha spigoli, quindi la soluzione è quasi sempre interna, con tutti i coefficienti diversi da zero.

Conclusione geometrica fondamentale:
Il lasso induce sparsità perché la forma "spigolosa" del vincolo ℓ_1 favorisce soluzioni sui vertici, mentre il ridge non induce sparsità perché il vincolo ℓ_2 è rotondo e non presenta punti preferenziali.



In sintesi

- ℓ_1 crea soluzioni sparse ($\hat{\beta}_j = 0$ per molti j);
- ℓ_2 produce shrinkage ma nessun coefficiente esattamente nullo;
- la differenza nasce interamente dalla geometria delle regioni di vincolo.

7.2 Lasso

Il metodo Lasso è stato proposto da Robert Tibshirani nel 1996 nel contesto della regressione lineare, con il nome di *Least Absolute Shrinkage and Selection Operator*. L'idea alla base è introdurre una penalizzazione di tipo ℓ_1 che permetta non solo di ridurre (shrinkare) i coefficienti come nel ridge, ma anche di annullarli esattamente, ottenendo dunque selezione automatica delle variabili.

Il Lasso stima β_0 tramite la coppia

$$(\hat{\mu}^L, \hat{\beta}^L_\lambda)$$

che minimizza il problema di ottimizzazione

$$\frac{1}{2n}\|y - \mu 1 - X\beta\|_2^2 + \lambda\|\beta\|_1, \qquad (\mu, \beta) \in R \times R^p.$$

Analogamente alla ridge regression, l'effetto della penalizzazione è di spingere i coefficienti verso lo zero. Tuttavia, a differenza della penalizzazione ℓ_2, la norma ℓ_1 può rendere **esattamente nulli** alcuni coefficienti, producendo un modello sparso e realizzando contemporaneamente stima e selezione delle variabili.

Come per il ridge, è possibile centrare e scalare le colonne di X e centrare y, eliminando così il termine di intercetta μ. In tal caso, il problema penalizzato del Lasso può essere scritto come:

$$\min_{\beta \in R^p} \left\{ \frac{1}{2n}\|y - X\beta\|_2^2 + \lambda\|\beta\|_1 \right\}.$$

Un'altra formulazione equivalente è quella *vincolata*, che corrisponde a limitare l'ampiezza della norma ℓ_1 del vettore dei coefficienti:

$$\min_{\beta \in R^p} \|y - X\beta\|_2^2 \quad \text{soggetto a} \quad \|\beta\|_1 \leq s.$$

Qui il parametro di tuning $s \geq 0$ controlla la dimensione della regione del vincolo. Per ogni valore di s esiste un valore di $\lambda \geq 0$ tale che le soluzioni dei due problemi coincidono, e viceversa.

Il punto chiave è che la norma ℓ_1 introduce una geometria del vincolo "spigolosa", che favorisce soluzioni su vertici in cui alcuni coefficienti sono esattamente zero. È proprio questa struttura che consente al Lasso di ottenere modelli sparsi, migliorando interpretabilità ed efficienza predittiva quando il segnale è realmente parziale o concentrato in poche variabili.

7.2.1 Shrinkage and selection

Il Lasso combina in modo unico due effetti fondamentali: **shrinkage** e **selezione**. Questi due aspetti sono alla base della potenza del metodo e ne caratterizzano il comportamento.

Shrinkage. La stima lasso

$$\hat{\beta}_\lambda^L$$

è una versione "shrunken", ossia contratta verso lo zero, dei coefficienti OLS. Come nella ridge regression, la penalizzazione scoraggia valori grandi dei coefficienti riducendone l'ampiezza:

$$|\hat{\beta}_{\lambda,j}^L| < |\hat{\beta}_j^{OLS}| \quad \text{per molti } j.$$

Selection. A differenza della ridge regression, il Lasso può annullare esattamente alcuni coefficienti. Si definisce insieme attivo:

$$\hat{S}_\lambda = \{\, k : \hat{\beta}_{\lambda,k}^L \neq 0 \,\},$$

ovvero l'insieme degli indici dei predittori effettivamente inclusi dal modello. Tutti gli altri predittori con coefficiente esattamente nullo vengono esclusi.

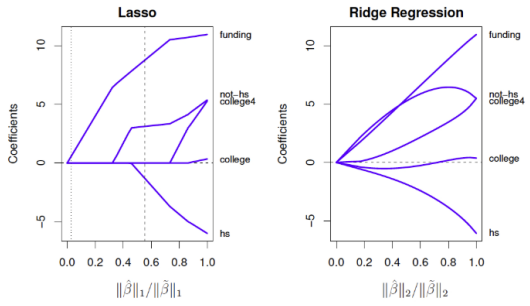
Perché questo è importante? Il nome "*Least Absolute Shrinkage and Selection Operator*" descrive esattamente ciò che accade:

- **Shrinkage:** il Lasso penalizza i coefficienti tramite la norma ℓ_1, riducendone la magnitudine e controllando l'overfitting;
- **Selection:** la natura "angolosa" della palla ℓ_1 induce soluzioni sparse: alcuni coefficienti sono esattamente zero, quindi il modello effettua una vera e propria selezione delle variabili.

Perché la sparsità è utile? La sparsità introdotta dal Lasso produce due benefici cruciali:

1. **Velocità.** Molti algoritmi e procedure statistiche possono essere notevolmente velocizzati quando il vettore dei coefficienti è sparso. Il modello risultante è più leggero, più rapido da stimare e più semplice da utilizzare, anche in contesti ad alta dimensionalità.
2. **Interpretabilità.** Quando p è molto grande (centinaia o migliaia di predittori), avere un modello che seleziona automaticamente solo alcune variabili è un enorme vantaggio. La sparsità permette di focalizzarsi sui predittori realmente importanti e di comprendere più chiaramente la struttura del fenomeno.

In sintesi, il Lasso non si limita a ridurre l'ampiezza dei coefficienti, ma crea modelli più semplici, più interpretabili e più efficienti. È proprio questa combinazione di **shrinkage** e **selezione** a renderlo uno strumento fondamentale nella regressione ad alta dimensionalità.



7.2.2 Convessità

Una delle differenze fondamentali tra la penalizzazione ℓ_1 e quella ℓ_0 riguarda la **convessità** dell'obiettivo di ottimizzazione.

La penalizzazione ℓ_0 porta a un problema di ottimizzazione non convesso e discontinuo, che richiede di esplorare tutte le combinazioni di variabili: un approccio computazionalmente proibitivo per p anche moderatamente grande.

Al contrario, la penalizzazione ℓ_1 utilizzata dal Lasso è **continua** e soprattutto **convessa**.

Definizione di funzione convessa. Una funzione

$$f : R^p \to R$$

è convessa se, per qualunque coppia di punti $b_1, b_2 \in R^p$ e per ogni $t \in [0, 1]$, vale la disuguaglianza:

$$f(tb_1 + (1-t)b_2) \;\leq\; tf(b_1) + (1-t)f(b_2).$$

Se la disuguaglianza è stretta per ogni $t \in (0, 1)$, la funzione si dice **strettamente convessa**.

Convessità del Lasso. La funzione obiettivo del Lasso,

$$\frac{1}{2n}\|y - X\beta\|_2^2 + \lambda\|\beta\|_1,$$

è la somma di:

- una funzione quadratica convessa ($\|y - X\beta\|_2^2$),
- e una funzione convessa ma non strettamente convessa ($\|\beta\|_1$).

Poiché la somma di funzioni convesse è ancora convessa, il problema del Lasso è un **problema convesso**.

Questo rappresenta un enorme vantaggio rispetto alla best subset selection: benché il Lasso non abbia una soluzione in forma chiusa come la ridge regression, il fatto che l'obiettivo sia convesso garantisce:

- **unico minimo globale** dell'obiettivo (anche se la soluzione $\hat{\beta}$ può non essere unica quando X non ha rango pieno),
- **ottimizzazione efficiente** tramite algoritmi come coordinate descent o LARS,
- **stabilità computazionale** anche per p molto grande.

Osservazione importante: La funzione obiettivo del Lasso non è strettamente convessa — questo significa che:

- possono esistere più soluzioni ottimali diverse,
- le soluzioni non sono funzioni lineari di y (a differenza della ridge regression),
- non esiste una formula esplicita (closed-form) per $\hat{\beta}_\lambda^L$.

Tuttavia, la convessità rende il problema ben posto e computazionalmente trattabile, ed è una delle ragioni per cui il Lasso è diventato uno degli strumenti fondamentali dell'analisi ad alta dimensionalità.

7.2.3 Convex optimization

Una proprietà fondamentale delle funzioni convesse è l'assenza di minimi locali che non siano anche minimi globali. In altre parole, se un punto b_0 è un minimo locale della funzione f, allora deve necessariamente essere anche un minimo globale.

Dimostrazione

Supponiamo, per assurdo, che esista un punto b_1 che è un minimo locale ma **non** è un minimo globale di f. Sia b_2 un punto dove f raggiunge il suo minimo globale.

Per qualsiasi $t \in [0, 1)$ vale:

$$tf(b_1) + (1-t)f(b_2) \;<\; tf(b_1) + (1-t)f(b_1) \;=\; f(b_1),$$

poiché, per ipotesi, $f(b_2) < f(b_1)$.

Dalla convessità di f abbiamo:

$$f\left(tb_1 + (1-t)b_2\right) \;\leq\; tf(b_1) + (1-t)f(b_2) \;<\; f(b_1).$$

Quindi il valore della funzione nel punto

$$tb_1 + (1-t)b_2$$

è strettamente minore di $f(b_1)$.

Per ogni intorno (vicinato) di b_1 possiamo scegliere t sufficientemente vicino a 1 in modo che $tb_1 + (1-t)b_2$ appartenga a quell'intorno.

Questo dimostra che b_1 non può essere un minimo locale, perché esistono sempre punti arbitrariamente vicini a b_1 in cui la funzione assume valori più bassi.

Conclusione

Una funzione convessa non può avere minimi locali diversi dal minimo globale. L'assenza di ottimi locali "spuri" è ciò che rende l'ottimizzazione convessa particolarmente potente.

Implicazioni pratiche

Grazie a questa proprietà, l'ottimizzazione di funzioni convesse può essere affrontata in modo efficiente tramite algoritmi iterativi come:

- **coordinate descent** (implementato in `glmnet`),
- **gradient descent** e varianti,
- **proximal gradient methods** (utilizzati per il Lasso),
- **LARS** in alcuni casi particolari.

Nel caso del Lasso, il coordinate descent è particolarmente efficace perché:

- la funzione obiettivo è convessa;
- la penalizzazione ℓ_1 permette aggiornamenti di coordinate molto semplici (soft-thresholding);
- non esistono minimi locali da cui rimanere "intrappolati".

La convessità è dunque uno dei motivi fondamentali per cui il Lasso è computazionalmente trattabile anche in regime ad alta dimensionalità.

7.2.4 The one standard error rule

In genere il parametro di tuning λ viene scelto come quello che minimizza l'errore di cross–validation (CV). Tuttavia, questa scelta spesso porta a modelli più complessi del necessario per fini interpretativi.

Per ottenere modelli più piccoli e più semplici, ma con prestazioni predittive comparabili, si utilizza una strategia molto efficace chiamata *one standard error rule*.

Cross–validation standard errors

Per ciascun fold di validazione incrociata, indichiamo con

$$Err_{-k}, \qquad k = 1, \ldots, K$$

l'errore di predizione ottenuto sul fold k lasciato fuori.

Possiamo allora stimare la deviazione standard degli errori di CV come:

$$SE(CVErr) = \frac{1}{K}\, sd(Err_{-1}, \ldots, Err_{-K})\,.$$

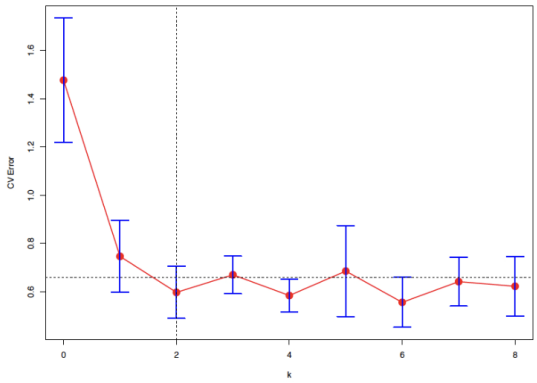
One standard error rule

Sia $\lambda_{\min}$ il valore di λ che minimizza l'errore di CV. La **one standard error rule** seleziona invece il valore di λ che corrisponde al modello più semplice tra quelli la cui CV error è entro una deviazione standard dal minimo, cioè:

$$CVErr(\lambda) \;\leq\; CVErr(\lambda_{\min}) + SE(CVErr).$$

In altre parole, scegliamo il modello più parsimonioso che non sia statisticamente peggiore del modello ottimale in termini di errore di CV.

Questa regola produce modelli:



- più piccoli,
- più interpretabili,
- con prestazioni predittive praticamente equivalenti al minimo CV.

7.2.5 Bias and variance of the lasso

Consideriamo lo stimatore lasso definito come

$$\hat{\beta}(\lambda) = \arg\min_{\beta \in R^p} \left\{ \|y - X\beta\|_2^2 + \lambda \|\beta\|_1 \right\}.$$

A differenza della regressione ridge, per il lasso non esistono formule chiuse semplici per descrivere bias e varianza dello stimatore. Tuttavia, il comportamento qualitativo è ben noto e segue la logica generale degli stimatori penalizzati.

Andamento del bias

All'aumentare di λ cresce la penalizzazione sulle componenti di β e quindi:

$$\lambda \uparrow \quad \Rightarrow \quad \text{bias}(\hat{\beta}(\lambda)) \uparrow .$$

Per $\lambda = 0$ non c'è penalizzazione e il lasso coincide con OLS:

$$\hat{\beta}(0) = \hat{\beta}^{OLS} \quad \Rightarrow \quad \text{bias} = 0.$$

Andamento della varianza

L'effetto della penalizzazione è quello di "stabilizzare" lo stimatore riducendo la variabilità:

$$\lambda \uparrow \quad \Rightarrow \quad \text{var}(\hat{\beta}(\lambda)) \downarrow .$$

Nel limite $\lambda \to \infty$, tutti i coefficienti vengono spinti verso zero:

$$\lim_{\lambda \to \infty} \hat{\beta}(\lambda) = 0,$$

e quindi la varianza dello stimatore tende a zero.

Conclusione: trade-off bias–varianza

Come per molti metodi di regolarizzazione:

- bias cresce con λ,
- varianza diminuisce con λ.

Il valore ottimale di λ bilancia questi due effetti per minimizzare l'errore quadratico medio (MSE).

Prestazioni predittive

In termini di prediction error, il lasso si comporta in modo molto simile alla regressione ridge: entrambi migliorano notevolmente rispetto a OLS quando il problema è ad alta varianza e/o p è grande rispetto a n.

La principale differenza tra ridge e lasso non è quindi nella capacità predittiva media, ma nella **sparsità** prodotta dal lasso (coefficenti esattamente nulli), con benefici interpretativi e computazionali.

7.3 Elastic Net

L'elastic net nasce dall'idea di combinare i punti di forza di due metodi complementari: la sparsità indotta dalla penalizzazione ℓ_1 del lasso e la stabilità statistica della penalizzazione ℓ_2 del ridge regression.

Come visto, la lasso può produrre soluzioni non uniche ed è particolarmente instabile quando i predittori sono fortemente correlati. Al contrario, il ridge non produce soluzioni sparse, ma è sempre ben posto e restituisce stime più stabili. L'elastic net mette insieme i due approcci.

Definiamo la funzione obiettivo per un dato $\lambda > 0$ e un parametro di mescolamento $\alpha \in [0, 1]$:

$$f(\beta; \lambda, \alpha) = \frac{1}{2n}\|y - X\beta\|_2^2 + \lambda \left((1-\alpha)\frac{1}{2}\|\beta\|_2^2 + \alpha\|\beta\|_1 \right).$$

L'estimatore elastic net è quindi

$$\widehat{\beta}^{EN}_{\lambda,\alpha} = \arg\min_{\beta} f(\beta; \lambda, \alpha).$$

Impostazioni particolari:

$$\alpha = 1 \Rightarrow \text{Lasso}, \qquad \alpha = 0 \Rightarrow \text{Ridge}.$$

L'elastic net conserva la capacità di effettuare selezione di variabili, grazie alla componente ℓ_1, ma eredita dal ridge una maggiore stabilità e unicità della soluzione.

Perché serve una penalizzazione ridge aggiuntiva?

La penalizzazione ℓ_1 del lasso favorisce la sparsità, ma introduce tre problemi importanti:

- **Soluzioni non uniche:** in presenza di predittori altamente correlati, il problema non è strettamente convesso e il lasso può restituire soluzioni diverse a seconda dell'algoritmo o dell'ordine delle variabili.
- **Instabilità nella selezione:** tra variabili molto correlate, il lasso tende a sceglierne una sola in modo arbitrario, rendendo la selezione poco interpretabile.

- **Alta varianza:** la soluzione può fluttuare molto al variare del campione.

Aggiungere un piccolo termine ℓ_2 (ridge) risolve questi problemi:

- il problema diventa **strettamente convesso** se $(1-\alpha)\lambda > 0$, garantendo l'unicità della soluzione;
- il modello diventa più stabile nei dati ad alta correlazione;
- la varianza delle stime si riduce in modo significativo.

Questo giustifica l'introduzione del termine ridge anche quando l'obiettivo principale è la selezione di variabili.

Effetto della combinazione L1 + L2

Il termine di penalizzazione dell'elastic net è:

$$\lambda \left((1-\alpha)\frac{1}{2}\|\beta\|_2^2 + \alpha\|\beta\|_1 \right),$$

che combina due effetti:

- **Penalizzazione ℓ_1:** induce sparsità, spegne alcuni coefficienti e permette la selezione delle variabili.
- **Penalizzazione ℓ_2:** stabilizza la soluzione, distribuisce il peso tra predittori correlati e riduce la varianza.

Il parametro α controlla il bilanciamento:

$$\alpha = 1 \Rightarrow \text{modello totalmente sparso}, \qquad \alpha = 0 \Rightarrow \text{modello totalmente stabile ma non sparso.}$$

Per $0 < \alpha < 1$ otteniamo un compromesso spesso molto più efficace della lasso pura.

Quando l'elastic net è preferibile alla lasso?

L'elastic net risulta vantaggioso nei seguenti scenari:

- presenza di **forte correlazione** tra variabili (gruppi di predittori correlati);
- numero di predittori molto più grande di n;
- necessità di **soluzioni uniche** e stabili;
- desiderio di un modello **sparse ma non troppo aggressivo**.

Mentre la lasso tende a selezionare una sola variabile da un gruppo di variabili fortemente correlate, l'elastic net tende a selezionare l'intero gruppo, preservando informazione ed evitando scelte arbitrarie.

Proprietà di unicità

Poiché il termine ridge è strettamente convesso, la soluzione elastic net è unica ogni volta che:

$$(1-\alpha)\lambda > 0.$$

Questo elimina completamente il problema della non unicità della lasso e rende la stima molto più affidabile dal punto di vista computazionale.

Interpretazione geometrica

La regione di vincolo dell'elastic net è un'interpolazione tra:

- il rombo della penalizzazione ℓ_1 (lasso),
- la sfera della penalizzazione ℓ_2 (ridge).

Il risultato è una figura "a cuscino" che conserva i vertici della lasso (che inducono sparsità), ma smussa gli angoli grazie alla componente ridge, rendendo la soluzione più stabile.

7.4 Relaxed Lasso

Un limite importante del Lasso è che la stessa penalizzazione che impone sparsità (ponendo molti coefficienti esattamente a zero) induce anche una forte contrazione dei coefficienti non nulli. Questo fenomeno, noto come *shrinkage*, può introdurre un bias significativo nelle stime. Una strategia semplice per ridurre questo bias consiste nel procedere in due fasi. Si considera innanzitutto

$$\widehat{A}_\lambda = \{k : \widehat{\beta}_k^{L,\lambda} \neq 0\},$$

ossia l'insieme dei predittori selezionati dal Lasso con parametro di penalizzazione λ. Si ricalcola poi la stima dei coefficienti mediante OLS utilizzando solo le variabili in $\widehat{A}_\lambda$. Questo produce una stima più "sbloccata":

$$\widehat{\beta}^{OLS,\lambda}_{\widehat{A}_\lambda},$$

che mantiene la sparsità del Lasso ma remove parte dello shrinkage. Un'alternativa più flessibile è il *Relaxed Lasso*, introdotto originariamente da Meinshausen (2006). L'idea è di combinare la stima Lasso e la stima OLS limitata alle variabili selezionate:

$$\widehat{\beta}_\lambda^{RELAX} = \gamma\,\widehat{\beta}^{L,\lambda} + (1-\gamma)\,\widehat{\beta}^{OLS,\lambda}, \qquad \gamma \in [0,1].$$

Il parametro γ controlla quanta "rigidità Lasso" mantenere:

- $\gamma = 1$: si recupera la stima Lasso standard;
- $\gamma = 0$: si ottiene la stima OLS sul supporto selezionato dal Lasso;
- valori intermedi di γ offrono un compromesso tra riduzione della varianza (via shrinkage) e riduzione del bias (via OLS).

Il parametro γ viene tipicamente selezionato tramite cross-validation.

7.5 Adaptive Lasso

Un approccio alternativo per mitigare il bias introdotto dalla penalizzazione ℓ_1 è l'*Adaptive Lasso*, proposto da Zou (2006). L'idea è di ripesare la penalizzazione utilizzando un set iniziale di stime, in modo da penalizzare meno i coefficienti che sembrano essere davvero rilevanti. Si parte da una stima iniziale $\widehat{\beta}^{init}$ di β_0, ad esempio ottenuta dal Lasso, e si definisce

$$\widehat{S}_{init} = \{k : \widehat{\beta}_k^{init} \neq 0\}.$$

Si risolve quindi un problema di Lasso pesato:

$$\widehat{\beta}_\lambda^{adapt} = \arg \min_{\beta:\beta_{(\widehat{S}_{init})^c}=0} \left\{ \frac{1}{2n}\|y - X\beta\|_2^2 + \lambda \sum_{k\in\widehat{S}_{init}} \frac{|\beta_k|}{|\widehat{\beta}_k^{init}|} \right\}.$$

Le variabili con coefficienti iniziali grandi ricevono una penalizzazione più debole, mentre quelle con coefficienti iniziali piccoli vengono penalizzate più duramente. In questo modo l'Adaptive Lasso:

- riduce ulteriormente il bias delle stime,
- migliora la stabilità della selezione,
- gode della proprietà dell'*oracle*: in condizioni regolari, identifica il set corretto di variabili rilevanti con probabilità che tende a 1.

7.6 Group Lasso

In molti contesti i predittori non sono indipendenti tra loro, ma possono essere naturalmente organizzati in gruppi. Esempi tipici includono:

- variabili derivanti dalla codifica di un predittore categoriale (dummy coding),
- espansioni in basi (splines, wavelets, polinomi),
- gruppi di variabili biologicamente correlate, come geni appartenenti alla stessa via metabolica.

Per gestire questa struttura **a gruppi**, Yuan e Lin (2006) propongono il *Group Lasso*. Si considera una partizione dell'insieme degli indici dei predittori:

$$G_1, G_2, \ldots, G_q \quad \text{con} \quad \bigcup_{k=1}^{q} G_k = \{1, \ldots, p\}.$$

Indichiamo con β_{G_k} il vettore dei coefficienti appartenenti al gruppo G_k. La penalizzazione del Group Lasso è definita come:

$$\lambda \sum_{k=1}^{q} m_k \, \|\beta_{G_k}\|_2,$$

dove i pesi $m_k > 0$ vengono utilizzati per compensare gruppi di dimensioni diverse. Una scelta standard è:

$$m_k = \sqrt{|G_k|},$$

in modo che la penalizzazione sia bilanciata tra gruppi piccoli e gruppi grandi.

Proprietà fondamentali

La penalizzazione a norma ℓ_2 sui gruppi induce un comportamento molto diverso dal Lasso classico:

- il Group Lasso tende a **selezionare interi gruppi** di predittori;
- o un gruppo G_k è completamente escluso dal modello ($\widehat{\beta}_{G_k} = 0$),
- oppure tutti i coefficienti del gruppo sono inclusi ($\widehat{\beta}_j \neq 0$ per ogni $j \in G_k$).

Questo è utile quando la rilevanza delle variabili è naturalmente legata alla rilevanza dell'intero gruppo.

Motivazioni

La selezione a gruppi risolve vari problemi del Lasso standard:

- evita la selezione arbitraria di una singola dummy per una variabile categoriale;
- mantiene assieme le componenti di una base funzionale;
- permette interpretazioni più coerenti quando le variabili hanno una struttura gerarchica o funzionale.

In sintesi, il Group Lasso introduce una forma di sparsità più strutturata:

sparsità sui gruppi, non sui singoli coefficienti.

Questa proprietà lo rende particolarmente adatto a problemi complessi in cui i predittori non sono naturalmente indipendenti ma organizzati in blocchi informativi.

Chapter 8

Linear smoothers

In molte situazioni, la relazione tra x e y non è lineare. A seguito, vengono riportati e approfonditi gli approcci più comuni per affrontare questo tipo di relazioni non lineari.

8.1 Local Regression

8.1.1 Local Linear Regression

L'idea alla base della regressione locale è che, se la funzione di regressione $f(x)$ è derivabile in un punto x_0, allora in un intorno sufficientemente piccolo essa può essere ben approssimata da una funzione lineare. In particolare, usando l'espansione di Taylor al primo ordine otteniamo:

$$f(x) \approx f(x_0) + f'(x_0)(x - x_0) = \underbrace{f(x_0)}_{\alpha} + \underbrace{f'(x_0)}_{\beta}(x - x_0).$$

Per stimare questa retta locale utilizziamo una regressione weighted least squares: le osservazioni vicine a x_0 pesano di più, mentre quelle lontane di meno. Definiamo quindi il problema:

$$\min_{\alpha,\,\beta} \sum_{i=1}^{n} \big(y_i - \alpha - \beta(x_i - x_0)\big)^2 w_h(x_i - x_0),$$

dove:

- $h > 0$ è il **bandwidth** o parametro di smoothing;
- $w_h(\cdot)$ è un **kernel** simmetrico (per esempio una densità normale centrata in 0);
- $w_h(x_i - x_0)$ assegna pesi decrescenti al crescere della distanza da x_0.

La stima della funzione in x_0 è quindi il valore previsto dalla retta locale nel punto x_0:

$$\hat{f}(x_0) = \hat{\alpha}.$$

Ripetendo questo procedimento per ogni punto x, si ottiene un'intera curva stimata $\hat{f}(x)$, che è un **smoother** della relazione tra x e y.

Teorema (Local Linear Smoothing)

La stima locale può essere espressa in forma esplicita tramite un "filtro lineare". Per un generico punto x vale:

$$\hat{f}(x) = \frac{\frac{1}{n}\sum_{i=1}^{n}\Big\{a_2(x;h) - a_1(x;h)(x_i - x)\Big\}\, w(x_i - x; h)\, y_i}{a_2(x;h)a_0(x;h) - a_1(x;h)^2}.$$

I coefficienti $a_r(x;h)$ sono definiti come:

$$a_r(x;h) = \frac{1}{n}\sum_{i=1}^{n}(x_i - x)^r\, w(x_i - x; h), \qquad r = 0, 1, 2.$$

Interpretazione

- Il kernel w controlla quanto "vicine" devono essere le osservazioni per contribuire alla stima in x.
- Il bandwidth h è il parametro più importante:
  - h piccolo $\rightarrow$ curva molto flessibile (potenziale overfitting);
  - h grande $\rightarrow$ curva molto liscia (potenziale underfitting).

La scelta del kernel è meno rilevante: una scelta comune è la densità normale

$$w(x;h) = \exp\left(-\frac{x^2}{2h^2}\right),$$

che assegna pesi grandi ai punti vicini e pesi molto piccoli ai punti lontani.

In sintesi, la regressione locale lineare produce uno smoother che combina flessibilità e stabilità, ed è ampiamente utilizzata in statistica non parametrica per stimare relazioni non lineari.

8.1.2 Local Quadratic Regression

La regressione locale lineare fornisce buone stime in regioni in cui la funzione vera $f(x)$ è approssimativamente lineare. Tuttavia, quando $f(x)$ presenta **curvatura**, il modello locale lineare introduce un bias non trascurabile.

Per ridurre questo bias, si può passare ad una **regressione locale quadratica**, in cui la funzione è approssimata tramite un polinomio di secondo ordine nell'intorno di un punto x_0:

$$f(x) \approx f(x_0) + f'(x_0)(x - x_0) + \tfrac{1}{2}f''(x_0)(x - x_0)^2.$$

Più in generale, stimiamo i coefficienti del polinomio locale risolvendo:

$$\min_{\alpha,\beta,\gamma}\sum_{i=1}^{n}\big(y_i - \alpha - \beta(x_i - x_0) - \gamma(x_i - x_0)^2\big)^2\, w_h(x_i - x_0),$$

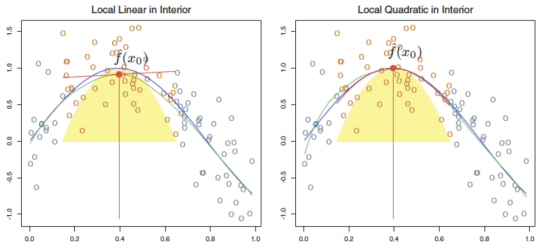
dove:

- α approssima $f(x_0)$,

- β approssima $f'(x_0)$,
- γ approssima $\frac{1}{2}f''(x_0)$,
- $w_h(\cdot)$ è il kernel pesato,
- h è il bandwidth.

Stima nel punto x_0:

$$\hat{f}(x_0) = \hat{\alpha}.$$



Perché usare la regressione locale quadratica?

- La regressione locale lineare elimina il bias di bordo ed è meno variabile della regressione locale costante (Nadaraya–Watson), ma mantiene bias significativo quando la funzione ha curvatura.
- La regressione locale quadratica approssima meglio la funzione in presenza di concavità o convessità locali, riducendo notevolmente il bias.
- A parità di bandwidth, il modello quadratico riesce a seguire meglio le variazioni locali del segnale.

Limiti

- La regressione locale quadratica è più flessibile, ma comporta **maggiore varianza** rispetto a quella lineare: tre parametri da stimare invece di due.
- In presenza di rumore elevato, potrebbe introdurre oscillazioni indesiderate se il bandwidth h è troppo piccolo.

In sintesi, la regressione locale quadratica è un metodo efficace per ridurre il bias in presenza di curvatura della funzione vera, al costo di una maggiore variabilità. La scelta tra regressione locale lineare o quadratica dipende quindi dal trade-off bias–varianza e dalla struttura della funzione sottostante.

8.1.3 On the choice of the kernel

La scelta del kernel non è particolarmente critica: numerosi studi mostrano che, a parità di bandwidth, kernel diversi producono stime molto simili. Un kernel con bandwidth h può essere scritto nella forma

$$w(t;h) = \frac{1}{h}\, w_0\left(\frac{t}{h}\right),$$

dove w_0 è una funzione di densità simmetrica attorno a zero.

Una scelta comune per w_0 è la densità normale standard $N(0,1)$, che porta al kernel gaussiano

$$w(t;h) = \frac{1}{\sqrt{2\pi}h} \exp\left(-\frac{t^2}{2h^2}\right) = N(0,h^2).$$

Tuttavia, molte altre scelte sono possibili, in particolare i kernel a **supporto limitato**, come il biquadratico o il tricubico:

$$w_0(t) = \begin{cases} (1-t^2)^2, & |t|<1, \\ 0, & \text{altrimenti}, \end{cases} \qquad w_0(t) = \begin{cases} (1-|t|^3)^3, & |t|<1, \\ 0, & \text{altrimenti}. \end{cases}$$

Il vantaggio principale dei kernel a supporto limitato è che annullano molti pesi, riducendo significativamente il costo computazionale.

Common kernel choices

Kernel	$w(z)$	**Supporto**
Gaussian	$\frac{1}{\sqrt{2\pi}} e^{-z^2/2}$	R
Rectangular	$\frac{1}{2}$	$[-1,1]$
Epanechnikov	$\frac{3}{4}(1-z^2)$	$[-1,1]$
Biquadratic	$\frac{15}{16}(1-z^2)^2$	$[-1,1]$
Tricubic	$\frac{70}{81}(1-\lvert z\rvert^3)^3$	$[-1,1]$

Considerazioni teoriche

Risultati asintotici mostrano che il kernel di Epanechnikov è "ottimale" in un senso formale, in quanto minimizza l'errore quadratico medio asintotico. Tuttavia, nella pratica, differenze tra kernel sono spesso trascurabili rispetto alla scelta di h.

I kernel a supporto limitato offrono vantaggi computazionali, perché richiedono di considerare soltanto i punti che cadono in un intervallo ristretto attorno a x. Di contro, questi kernel possono generare derivate discontinue della curva stimata $\hat{f}(x)$, cosa potenzialmente indesiderabile in applicazioni che richiedono grande regolarità della stima.

In conclusione, la scelta del kernel è poco rilevante; la scelta del bandwidth è cruciale.

8.1.4 Bias-variance tradeoff

Una componente cruciale della regressione locale è la scelta del parametro di smoothing h. Per n sufficientemente grande possiamo approssimare:

$$E\Big(\hat{f}(x)\Big) \approx f(x) + \frac{h^2}{2}\,\sigma_w^2\,f''(x), \qquad \mathrm{Var}\Big(\hat{f}(x)\Big) \approx \frac{\sigma_w^2}{nh}\,\frac{g(x)}{\alpha(w)},$$

dove:

$$\sigma_w^2 = \int z^2 w(z)\,dz, \qquad \alpha(w) = \int w(z)^2 dz,$$

e $g(x)$ è la densità da cui provengono gli x_i.

Interpretazione

Queste espressioni ci permettono di osservare che:

- il **bias** cresce come h^2;
- la **varianza** cresce come $(nh)^{-1}$.

Se scegliessimo $h \to 0$, il bias si ridurrebbe ma la varianza esploderebbe; se scegliessimo $h \to \infty$, accadrebbe il contrario.

Esiste quindi un naturale **compromesso bias–varianza** che determina la scelta ottimale di h.

Bandwidth ottimale

Minimizzando l'errore quadratico medio asintotico otteniamo:

$$h_{\mathrm{opt}}(x) = \left(\frac{1}{n}\,\frac{\sigma^2\,\alpha(w)}{\sigma_w^4\,f''(x)^2\,g(x)}\right)^{1/5}.$$

Questa espressione è poco utile in pratica, perché richiede quantità ignote come $f''(x)$, $g(x)$ e σ^2. Tuttavia mette in evidenza due aspetti fondamentali:

- il bandwidth ottimale decresce come $n^{-1/5}$, quindi **molto lentamente**;
- sostituendo $h_{\mathrm{opt}}(x)$ nelle formule si ottiene un MSE che converge come $n^{-4/5}$, molto più lento rispetto al caso **parametrico** (n^{-1}).

Questo riflette la maggiore flessibilità della regressione locale rispetto ai modelli parametrici: la stima $\hat{f}(x)$ è più adattiva, ma paga in termini di velocità di convergenza statistica.

8.1.5 Loess regression

In molti casi è vantaggioso utilizzare un **bandwidth non costante** lungo l'asse delle x, adattandolo al livello di densità o rarità delle osservazioni.

Motivazione

Se i punti x_i sono molto sparsi in una certa regione, è ragionevole utilizzare un valore di h più grande; al contrario, quando gli x_i sono molto densi, possiamo permetterci un bandwidth più piccolo. Questa idea porta naturalmente all'uso di un **bandwidth variabile**.

LOWESS / LOESS

La procedura *loess* (o *lowess*, **locally weighted estimated scatterplot smoothing**), introdotta da Cleveland (1979), implementa in modo sistematico questa filosofia.

Il meccanismo principale è il seguente:

- la quantità di smoothing è definita tramite un parametro chiamato **span**, che indica la **frazione di osservazioni effettive** utilizzate per stimare $f(x)$ in un punto x_0;
- questa frazione è mantenuta costante lungo tutto l'asse delle x;
- se si utilizza un kernel a supporto limitato, mantenere una frazione fissa implica **automaticamente un bandwidth variabile**, proprio come accade nei metodi KNN:

$$h = h(x_0) \quad \text{dipende dalla densità degli } x_i.$$

Interpretazione

Questa scelta comporta che:

- nelle regioni più popolate, la finestra locale è più stretta (bandwidth piccolo);
- nelle regioni meno popolate, la finestra si allarga (bandwidth grande);
- il livello di smoothing rimane **coerente e comparabile** lungo tutto il dominio.

Robustezza

Un ulteriore vantaggio del metodo *loess* è che combina il bandwidth variabile con idee di **stima robusta**: le osservazioni considerate outlier ricevono un peso progressivamente più piccolo, così da ridurre la loro influenza sul valore finale di $\hat{f}(x)$.

Questa caratteristica rende *loess* particolarmente efficace nella pratica, soprattutto in presenza di:

- dati rumorosi,
- outlier sporadici,
- distribuzioni irregolari dei punti sull'asse delle x.

In sintesi, loess offre uno smoothing locale adattivo e robusto, capace di produrre curve stimate molto stabili e realistiche anche in situazioni di forte eterogeneità dei dati.

8.1.6 Variability bands

Una quantità quasi-pivotale che si utilizza spesso in regressione locale è

$$\frac{\hat{f}(x) - f(x) - b(x)}{\sqrt{\mathrm{Var}(\hat{f}(x))}} \;\approx\; N(0,1),$$

dove $b(x)$ è il bias della stima locale. A differenza del caso parametrico, qui il bias **non può essere ignorato**, poiché può essere grande rispetto alla varianza, specialmente quando il bandwidth non è ottimale.

Correggere il bias in modo esatto richiederebbe formule complesse e dipendenti da quantità ignote, come $f''(x)$ e la densità dei punti. Per questo motivo, una pratica moderna e largamente adottata non è costruire **confidence intervals**, bensì **variability bands**.

Variability bands

I variability bands hanno la forma:

$$\big(\hat{f}(x) - z_{\alpha/2}\,\mathrm{SE}(\hat{f}(x)),\ \hat{f}(x) + z_{\alpha/2}\,\mathrm{SE}(\hat{f}(x))\big),$$

dove:

- $z_{\alpha/2}$ è il quantile standard della normale;
- $\mathrm{SE}(\hat{f}(x))$ è la deviazione standard stimata della stima locale.

Queste bande rappresentano un'indicazione della **variabilità locale** di $\hat{f}(x)$, cioè quanto la curva stimata potrebbe oscillare attorno al valore previsto, tenuto conto dell'incertezza dovuta ai dati e alla scelta del bandwidth.

Importante

I variability bands:

- **non** sono intervalli di confidenza al livello nominale;
- **non** garantiscono copertura corretta, perché ignorano il bias;
- sono però molto utili per mostrare graficamente l'incertezza locale della stima.

In pratica, mentre gli intervalli di confidenza richiederebbero una correzione esplicita del bias (spesso impossibile), i variability bands forniscono una rappresentazione semplice e informativa della **variabilità** del metodo di smoothing.

8.2 Non-linearity

Una situazione in cui i modelli lineari iniziano a comportarsi in modo non ottimale è quando la relazione tra la risposta y e il predittore x non è lineare, né può essere approssimata in modo soddisfacente da una relazione lineare.

Come esempio di modello non lineare consideriamo

$$y_i = \cos(\beta_1 x_i) + e^{-x_i \beta_2} + \varepsilon_i,$$

dove ε_i è un termine di errore.

L'approccio naturale per stimare i parametri sconosciuti, dati i dati osservati, consiste nel minimizzare la somma dei quadrati dei residui: si parla di *non-linear least squares*. In generale, però, non esiste una soluzione in forma chiusa.

Spesso, nella pratica, ci limitiamo a supporre che

$$E(y_i) = f(x_i)$$

per qualche funzione sconosciuta f, che vogliamo stimare in modo flessibile a partire dai dati.

8.2.1 Basis expansion

L'idea della regressione polinomiale può essere generalizzata e migliorata tramite le *basis expansions*. Il principio è di arricchire o sostituire l'input x con trasformazioni non lineari di esso.

Ricordiamo che la regressione lineare richiede solo che il modello sia lineare rispetto ai parametri β_j, non rispetto alla variabile x.

Per esempio, per un predittore scalare x_i, possiamo utilizzare la base polinomiale

$$y_i = \sum_{k=1}^{K} x_i^{k-1} \beta_k + \varepsilon_i.$$

Più in generale, possiamo modellare la relazione come

$$y_i = \sum_{k=1}^{K} B_k(x_i) \beta_k + \varepsilon_i,$$

dove le funzioni $B_k(\cdot)$ costituiscono una base (polinomi, funzioni trigonometriche, spline, ecc.). Questo approccio è noto come *basis expansion*.

Limiti della regressione polinomiale

La regressione polinomiale, pur essendo semplice, può risultare problematica.

Quando si usano polinomi, la natura *globale* dell'approssimazione porta spesso a prestazioni scarse in presenza di:

- elevata varianza del rumore;
- funzioni f con molte irregolarità o punti critici.

Consideriamo, ad esempio, la funzione

$$f(x) = \sin\big(2(4x-2)\big) + 2e^{-(16)^2(x-0.5)^2}.$$

Anche usando un polinomio di grado 15, la stima può essere scarsa in molte regioni e mostrare "oscillazioni" spurie dove non sembrano necessarie.

In teoria, i polinomi possono approssimare una vasta classe di funzioni (per il teorema di Taylor). In pratica, tuttavia, la regressione polinomiale non è ben adatta a modellare relazioni complesse: l'approssimazione è globale e la stima è spesso instabile ai bordi dell'intervallo.

In un modello flessibile ci si aspetta che la previsione in x_i dipenda soprattutto dalle osservazioni vicine a x_i. Al contrario, nei polinomi, anche le osservazioni lontane hanno un forte impatto su $\hat{f}(x_i)$, producendo oscillazioni spurie e stime instabili.

Una possibile idea è dividere i dati in sotto-intervalli delimitati da nodi (*knots*) e adattare un modello polinomiale separato in ciascun intervallo.

Definiamo K nodi interni sull'intervallo di x:

$$\min(x) < \xi_1 < \cdots < \xi_K < \max(x),$$

che definiscono $K+1$ pezzi o segmenti.

Su ciascuno dei $K+1$ intervalli

$$(-\infty, \xi_1],\ (\xi_1, \xi_2],\ \ldots,\ (\xi_{K-1}, \xi_K],\ (\xi_K, +\infty)$$

possiamo adattare un modello polinomiale locale.

Step function

Un modello di regressione a tratti costante è un esempio di basis expansion, in cui utilizziamo funzioni a gradino.

Con K nodi $\xi_1, \ldots, \xi_K$, definiamo le funzioni base:

$$B_1(x) = \mathbf{1}\{x < \xi_1\}, \quad B_2(x) = \mathbf{1}\{\xi_1 \le x < \xi_2\}, \ \ldots, \ B_K(x) = \mathbf{1}\{\xi_{K-1} \le x < \xi_K\},$$

$$B_{K+1}(x) = \mathbf{1}\{x \ge \xi_K\}.$$

Un modello lineare nella base $\{B_j(x)\}$ produce una funzione $f(x)$ costante su ciascun intervallo determinato dai nodi.

Piecewise linear regression

Il modello a tratti costante può essere troppo rigido. Per aumentare la flessibilità possiamo utilizzare funzioni *lineari a tratti*.

Con K nodi definiamo, ad esempio, le funzioni base:

$$B_1(x) = \mathbf{1}\{x < \xi_1\}, \quad B_2(x) = x\,\mathbf{1}\{x < \xi_1\},$$

$$B_3(x) = \mathbf{1}\{\xi_1 \le x < \xi_2\}, \quad B_4(x) = x\,\mathbf{1}\{\xi_1 \le x < \xi_2\},$$

$$\ldots$$

$$B_{2K}(x) = \mathbf{1}\{x \ge \xi_K\}, \quad B_{2(K+1)}(x) = x\,\mathbf{1}\{x \ge \xi_K\}.$$

In questo modo otteniamo una funzione lineare in ciascun intervallo, ma in generale **non continua** nei nodi, a meno di imporre opportune condizioni.

Polinomi a tratti e spline

Un caso base di polinomio è il cubico:

$$B_1(x) = 1, \qquad B_2(x) = x, \qquad B_3(x) = x^2, \qquad B_4(x) = x^3.$$

Le **spline** sono funzioni polinomiali a tratti con vincoli di continuità e di regolarità.

Storicamente sono state introdotte, ad esempio, in costruzioni navali per ottenere curve regolari: si fissavano dei pesi (i nodi) e si faceva passare una barra flessibile (lo spline) attraverso questi punti.

Una spline di grado d con nodi $\xi_1, \ldots, \xi_K$ è una funzione f tale che:

- f è un polinomio di grado d su ciascun intervallo

$$(-\infty, \xi_1], \ [\xi_1, \xi_2], \ [\xi_2, \xi_3], \ \ldots, \ [\xi_{K-1}, \xi_K], \ [\xi_K, +\infty);$$

- f è continua e possiede derivate continue fino all'ordine $d-1$ in ogni nodo, cioè

$$f(\xi_k^-) = f(\xi_k^+), \quad f'(\xi_k^-) = f'(\xi_k^+), \ \ldots, \ f^{(d-1)}(\xi_k^-) = f^{(d-1)}(\xi_k^+), \quad k = 1, \ldots, K,$$

  dove ξ_k^- e ξ_k^+ indicano, rispettivamente, il limite da sinistra e da destra in ξ_k.

Le **cubic splines** ($d = 3$) sono le più utilizzate in pratica. Dati nodi ordinati $\xi_1 < \cdots < \xi_K$, una cubic spline $f(x; \beta)$ è una funzione polinomiale cubica a tratti, con derivate prima e seconda continue.

Il grado d controlla la regolarità della spline:

- $d = 0 \Rightarrow$ funzione costante a tratti;
- $d = 1 \Rightarrow$ funzione lineare a tratti (continua ma con derivate discontinue);
- valori più alti di d aumentano la regolarità, ma rendono la spline più simile a un polinomio globale.

In pratica si usa raramente $d > 3$.

Analisi più approfondita: un singolo nodo

Invece di adattare un polinomio di ordine elevato, immaginiamo di adattare due polinomi lineari ai dati: uno per i punti con $x \leq k$ e uno per i punti con $x > k$, dove k è un nodo.

Possiamo descrivere questo approccio tramite una basis expansion. Una scelta possibile è:

$$B_1(x) = \mathbf{1}(x \leq k), \quad B_2(x) = x\,\mathbf{1}(x \leq k),$$

$$B_3(x) = \mathbf{1}(x > k), \quad B_4(x) = x\,\mathbf{1}(x > k).$$

È utile riformulare questa base in termini di intercetta e pendenza di base per $x \leq k$ e cambiamenti per $x > k$:

$$B_1(x) = 1, \quad B_2(x) = x, \quad B_3(x) = \mathbf{1}(x > k), \quad B_4(x) = (x - k)\,\mathbf{1}(x > k).$$

In questa forma è chiaro che, in generale, la funzione stimata non è continua in k (a meno di imporre vincoli sui coefficienti). Forzare la continuità al nodo (eliminando, di fatto, il "salto" in k) riduce i gradi di libertà di uno (da 4 a 3).

Quadratici a tratti e continuità

Per adattare due polinomi quadratici sui due lati di k, possiamo considerare le funzioni base:

$$B_1(x) = 1, \quad B_2(x) = x, \quad B_3(x) = x^2,$$

$$B_4(x) = (x - k)\,\mathbf{1}(x > k), \quad B_5(x) = (x - k)^2\,\mathbf{1}(x > k).$$

In questo caso abbiamo due polinomi quadratici (2×3 parametri) meno un vincolo di continuità, per un totale di $6 - 1 = 5$ gradi di libertà. La funzione è continua in k, ma la derivata prima non è necessariamente continua.

Per imporre anche la continuità della derivata, si può eliminare la base $B_4(x)$, riducendo ancora i gradi di libertà di uno. Questo illustra come ogni vincolo di regolarità (continuità, derivata continua, ecc.) riduca i gradi di libertà.

Funzione parte positiva e base troncata

Definiamo la *funzione parte positiva*:

$$(x)_+ = \begin{cases} x, & x \geq 0, \\ 0, & \text{altrimenti.} \end{cases}$$

Possiamo generalizzare ad un polinomio di ordine M usando la base:

$$B_1(x) = 1, \qquad B_{j+1}(x) = x^j, \quad j = 1, \ldots, M,$$

$$B_{M+2}(x) = (x - k)_+^M.$$

Questa base produce una funzione con derivate continue di ordine da 0 a $M - 1$ nel nodo k.

Truncated power basis

Possiamo generalizzare ulteriormente introducendo P nodi $k_1, \ldots, k_P$ e definendo la *truncated power basis* di ordine M come:

$$B_1(x) = 1, \qquad B_{j+1}(x) = x^j, \quad j = 1, \ldots, M,$$

$$B_{M+p+1}(x) = (x - k_p)_+^M, \quad p = 1, \ldots, P.$$

Questa base definisce polinomi a tratti di ordine M con derivate continue fino all'ordine $M-1$ in ciascun nodo.

In totale:

- ci sono $P + 1$ polinomi di ordine M;
- ci sono P insiemi di vincoli (continuità delle derivate fino a ordine $M - 1$);

quindi il numero di parametri liberi è

$$(P + 1)(M + 1) - PM = 1 + M + P.$$

Questa rappresentazione è alla base delle spline in forma di *truncated power basis*, molto utilizzata nelle implementazioni pratiche delle cubic splines e delle regressioni flessibili.

8.2.2 Regression splines

Una volta definite le funzioni base, possiamo adattare un modello di regressione per stimare la funzione sconosciuta $f(x)$ minimizzando la somma dei residui quadrati su tutte le funzioni nello spazio generato da tale base.

Come in ogni basis expansion, costruiamo esplicitamente una matrice di progetto B definita da

$$B_{i,j} = B_j(x_i), \qquad i = 1, \ldots, n, \; j = 1, \ldots, 1 + M + P.$$

La regression spline può essere scritta come

$$\hat{f}(x) = \sum_{j=1}^{1+P+M} \hat{\beta}_j \, B_j(x),$$

dove il vettore dei coefficienti stimati è

$$\hat{\beta} = (B^\top B)^{-1} B^\top y.$$

In pratica, le funzioni base più utilizzate sono quelle della **truncated power basis** con grado $M = 3$, cioè spline cubiche.

Natural cubic splines

Un comportamento tipico delle spline costruite tramite basis expansion è che, al di fuori dell'intervallo compreso tra il più piccolo e il più grande dei nodi, la funzione stimata può diventare instabile o oscillare in modo innaturale.

Le **natural splines** risolvono questo problema imponendo un ulteriore vincolo alla truncated power basis: la funzione viene forzata ad avere ordine $(M - 1)/2$ al di fuori dei nodi estremi.

Per spline cubiche ($M = 3$), le natural cubic splines risultano **lineari** fuori dai boundary knots. Questo impedisce l'instabilità e permette una migliore extrapolazione.

Una possibile parametrizzazione delle natural cubic splines è la seguente:

$$B_1(x) = 1, \qquad B_2(x) = x,$$

$$B_{j+2}(x) = \frac{(x-k_j)_+^3 - (x-k_P)_+^3}{k_P - k_j} - \frac{(x-k_{P-1})_+^3 - (x-k_P)_+^3}{k_P - k_{P-1}}, \qquad j = 1, \ldots, P-2.$$

Questa base produce spline cubiche che sono lineari oltre i nodi estremi k_1 e k_P.

The number of knots and where to put them

Le natural cubic splines presentano un problema pratico: **decidere il numero e il posizionamento dei nodi**. Un numero troppo elevato di nodi porta ad overfitting; troppo pochi nodi portano ad una funzione eccessivamente rigida.

Una possibile alternativa è rappresentata dalle **smoothing splines**, che utilizzano come nodi tutti i punti del training set ma controllano la complessità tramite penalizzazione (non tramite la scelta esplicita dei nodi).

In generale:

- i nodi sono parametri di complessità e devono essere scelti tramite criteri come la cross-validation;
- in linea teorica si potrebbero posizionare manualmente i nodi in modo ottimale, ma ciò porterebbe ad un problema di ottimizzazione proibitivo.

Nella pratica, i nodi vengono quasi sempre selezionati in due modi:

1. **posizionamento uniforme**: nodi equispaziati tra $\min(x)$ e $\max(x)$;
2. **posizionamento sui quantili** (approccio usato da `bs()` in R): assicura una banda locale adattiva, più fitta dove i dati sono più densi.

Il grado d della spline influenza quanti nodi è possibile utilizzare dato un certo numero di gradi di libertà p. Per esempio:

- spline lineari ($d = 1$): con $p = 12$ si possono usare $k = p - d - 1 = 10$ nodi;
- spline quadratiche ($d = 2$): con $p = 12$ si hanno $k = 9$ nodi;
- spline cubiche ($d = 3$): con $p = 12$ si hanno $k = 8$ nodi.

8.2.3 Smoothing splines

Dato un insieme di osservazioni $\{(x_i, y_i)\}_{i=1}^n$ e un parametro di smoothing λ, una **smoothing spline cubica** è definita come la funzione f che risolve il problema di ottimizzazione

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - f(x_i))^2 \; + \; \lambda \int (f''(x))^2 \, dx,$$

dove f appartiene alla famiglia delle funzioni due volte derivabili. Si può dimostrare che la soluzione ottima è una **natural cubic spline con nodi in ogni diverso valore di** x_i.

Il secondo termine è la **roughness penalty** e controlla la "wiggliness" della curva:

- $f''(x)$ è la derivata seconda di f: sarebbe zero se f fosse una retta, quindi misura la curvatura locale; - il segno di $f''(x)$ indica concavità/convessità, ma non interessa ai fini della penalizzazione, perciò lo si eleva al quadrato; - l'integrale somma la curvatura su tutto il dominio.

Il parametro di smoothing $\lambda \geq 0$ media tra interpolazione e eccessiva rigidità:

- $\lambda = 0$: nessuna penalizzazione $\Rightarrow$ la spline interpola esattamente i dati;
- $\lambda = \infty$: la curvatura è vietata $\Rightarrow$ f diventa lineare.

Forma matriciale

L'obiettivo può essere riscritto tramite la base delle natural cubic splines. Sia B la matrice base:

$$B_{ij} = B_j(x_i),$$

e sia Ω la matrice che quantifica la curvatura delle funzioni base:

$$\Omega_{jk} = \int B_j''(x)\, B_k''(x)\, dx.$$

La funzione obiettivo diventa:

$$\|y - B\beta\|_2^2 + n\lambda\, \beta^\top \Omega \beta.$$

La soluzione esplicita è:

$$\hat{\beta}_\lambda = (B^\top B + n\lambda\Omega)^{-1} B^\top y.$$

Stima lineare e smoothing matrix

Poiché l'estimatore è lineare rispetto a y, vale:

$$\hat{y} = B\hat{\beta}_\lambda = B(B^\top B + \lambda\Omega)^{-1} B^\top y = H_\lambda y,$$

dove H_λ è la **smoothing matrix**. La sua diagonale e la sua traccia giocano un ruolo chiave nella selezione di λ.

Selection of λ

Esistono due criteri principali:

Leave-One-Out Cross-Validation (LOOCV)

LOOCV seleziona il valore di λ che minimizza:

$$\sum_{i=1}^{n} \left(y_i - \hat{f}^{\lambda}_{-i}(x_i) \right)^2 = \sum_{i=1}^{n} \left(\frac{y_i - \hat{f}^{\lambda}(x_i)}{1 - \{H_\lambda\}_{ii}} \right)^2.$$

L'identità sopra evita di rifare n volte il fitting, e sfrutta la struttura lineare dell'estimatore.

Generalized Cross-Validation (GCV)

GCV è una versione computazionalmente più leggera di LOOCV: sostituisce la diagonale di H_λ con la sua traccia media.

$$\mathrm{GCV}(\lambda) = \sum_{i=1}^{n} \left(\frac{y_i - \hat{f}^{\lambda}(x_i)}{1 - \mathrm{trace}(H_\lambda)/n} \right)^2.$$

Il valore di λ scelto è quello che minimizza tale quantità.

Interpretazione

- λ grande $\rightarrow$ curva molto liscia, vicino a una funzione lineare
- λ piccolo $\rightarrow$ curva molto flessibile, può adattarsi al rumore
- GCV e LOOCV trovano il punto di equilibrio ottimale tra varianza e bias

Le smoothing splines offrono una procedura potente perché:

1. non richiedono la scelta manuale dei nodi;
2. sfruttano automaticamente una base ricca (un nodo per ogni x_i);
3. controllano la complessità tramite la penalizzazione, non tramite il numero dei parametri;
4. forniscono stimatori stabili con ottime proprietà teoriche.

Chapter 9

Data splitting for variable selection

In questo capitolo consideriamo l'inferenza in alta dimensionalità dal punto di vista del multiple testing e della selezione di variabili. Il problema centrale è come interpretare evidenze come un p-value piccolo quando vengono testate molte ipotesi simultaneamente, e come controllare gli errori mantenendo allo stesso tempo una buona potenza. Ci concentriamo su modelli di regressione con $p \geq n$ e discutiamo metodi di data splitting per ottenere un'inferenza post-selezione valida.

9.1 Inferenza in alta dimensionalità

Consideriamo il modello lineare gaussiano

$$\mathbf{y} \sim N_n(\mathbf{1}_n\beta_0 + \mathbf{X}\boldsymbol{\beta}, \sigma^2\mathbf{I}_n),$$

dove $\mathbf{X}$ è una matrice di progetto $n \times p$ e $\boldsymbol{\beta}$ è un vettore di coefficienti $p \times 1$.

Quando $p \geq n$, i metodi classici di stima e inferenza su $\boldsymbol{\beta}$ (ad esempio OLS, test t classici) non sono direttamente applicabili. La domanda chiave diventa:

- Come eseguire inferenza su $\boldsymbol{\beta}$ (intervalli di confidenza e p-value per i singoli parametri β_j, $j = 1, \ldots, p$) in un contesto ad alta dimensionalità?

Metodi di regressione penalizzata come il Lasso producono stime sparse ed eseguono selezione di variabili, ma non forniscono automaticamente p-value o intervalli di confidenza validi. È quindi necessario definire esplicitamente tassi di errore e potenza rispetto a una nozione di "segnali veri".

Insieme di supporto

L'**insieme di supporto** del segnale è

$$S = \{j \in \{1, \ldots, p\} : \beta_j \neq 0\},$$

con cardinalità $s = |S|$. Il suo complementare è l'insieme nullo

$$N = \{j \in \{1, \ldots, p\} : \beta_j = 0\}.$$

Sia $\hat{S} \subseteq \{1, \dots, p\}$ un estimatore di S, tipicamente ottenuto tramite una procedura di selezione (ad esempio l'insieme attivo del Lasso). Allora:

$$|\hat{S} \cap N|$$

è il numero di selezioni errate (errori di tipo I, cioè inclusioni false), mentre

$$|S \setminus \hat{S}|$$

è il numero di deselezioni errate (errori di tipo II, cioè esclusioni false).

Nella regressione penalizzata (Lasso, elastic net, ...) la preoccupazione principale è spesso il controllo delle false inclusioni, perché portano a modelli sovra-adattati e instabili. Le false esclusioni possono comunque verificarsi, in particolare quando p è grande o i predittori sono fortemente correlati.

Tassi di errore

Un modo utile per valutare una procedura di selezione $\hat{S}$ è tramite le seguenti quantità.

False Discovery Proportion (FDP). Definiamo

$$\mathrm{FDP} = \frac{|\hat{S} \cap N|}{|\hat{S}|},$$

con la convenzione $\mathrm{FDP}(\emptyset) = 0$. Questa quantità misura, **dopo** la selezione, la proporzione di variabili selezionate che sono in realtà nulle (falsi positivi). È una variabile aleatoria che varia da campione a campione.

Family-Wise Error Rate (FWER). Il family-wise error rate è

$$\mathrm{FWER} = P\big(\mathrm{FDP}(\hat{S}) > 0\big) = P(\hat{S} \cap N \neq \emptyset).$$

È la probabilità di commettere **almeno** una falsa inclusione. Si tratta di un criterio molto conservativo: anche un solo falso positivo viene considerato una violazione del vincolo. Di conseguenza, le procedure che controllano la FWER riducono fortemente il rischio di errori di tipo I, al prezzo di un aumento degli errori di tipo II.

False Discovery Rate (FDR). La false discovery rate è

$$\mathrm{FDR} = E\big(\mathrm{FDP}(\hat{S})\big).$$

Controllare la FDR significa garantire che, **in media**, la proporzione di scoperte false tra le variabili selezionate rimanga piccola. Questo è il principio alla base di molte procedure per il multiple testing (ad esempio Benjamini–Hochberg).

Controllo dell'errore e potenza

Vorremmo controllare il tasso di errore scelto al livello α, cioè

$$P(\hat{S} \cap N \neq \emptyset) \leq \alpha \quad \text{oppure} \quad E\big(\mathrm{FDP}(\hat{S})\big) \leq \alpha,$$

massimizzando al contempo una misura di potenza, ad esempio la potenza media

$$\text{AvgPower} = \frac{1}{|S|} \sum_{j \in S} P(j \in \hat{S}).$$

Esiste un trade-off tra errori di tipo I e di tipo II, e la FWER è più stringente della FDR. Infatti,

$$E\big(\text{FDP}(\hat{S})\big) \leq P(\hat{S} \cap N \neq \emptyset).$$

Pertanto, i metodi che controllano la FWER controllano automaticamente anche la FDR, ma in modo molto più conservativo, e risultano quindi meno potenti.

In contesti ad alta dimensionalità si osserva quindi il seguente quadro:

- Un controllo più rigido dell'errore (FWER) riduce drasticamente la probabilità di includere anche una sola variabile irrilevante; ciò è desiderabile in applicazioni ad alto rischio, ma tipicamente porta a molte false esclusioni e a bassa potenza.
- Un controllo più permissivo (FDR) consente una piccola proporzione di scoperte false; questo porta spesso a maggiore potenza e a una selezione più stabile, in particolare quando i segnali veri sono deboli o i predittori sono fortemente correlati.

Nei moderni problemi di regressione ad alta dimensionalità, metodi come il Lasso combinato con stability selection o data splitting cercano di controllare un adeguato tasso di errore mantenendo al contempo una potenza elevata.

9.2 Esempio: Lasso e tassi di errore

Illustriamo queste idee usando un disegno di simulazione ispirato alla Sezione 3.1 di Hastie et al. (2020).

Disegno di simulazione

Dato n (numero di osservazioni), p (dimensione), s (livello di sparsità), ν (signal-to-noise ratio, SNR) e ρ (livello di correlazione tra predittori), generiamo i dati come segue.

1. Definiamo $\boldsymbol{\beta} \in R^p$ in funzione di s e di un certo pattern di sparsità. Ad esempio, "beta-type 2": i primi s componenti sono uguali a 1 e i restanti a 0.
2. Estraiamo le righe della matrice dei predittori $\mathbf{X} \in R^{n \times p}$ i.i.d. da

   $$N_p(\mathbf{0}, \boldsymbol{\Sigma}),$$

   dove $\boldsymbol{\Sigma} \in R^{p \times p}$ ha elementi $\Sigma_{ij} = \rho^{|i-j|}$ (struttura di Toeplitz).
3. Estraiamo il vettore risposta $\mathbf{y} \in R^n$ da

   $$N_n(\mathbf{1}_n \beta_0 + \mathbf{X}\boldsymbol{\beta}, \sigma^2 \mathbf{I}_n),$$

   con σ^2 scelto per ottenere il SNR desiderato:

   $$\sigma^2 = \boldsymbol{\beta}^\top \boldsymbol{\Sigma} \boldsymbol{\beta} / \nu.$$

4. Adattiamo il Lasso e definiamo l'insieme attivo

$$\hat{S} = \{j \in \{1, \dots, p\} : \hat{\beta}_j \neq 0\},$$

dove λ è scelto, per esempio, mediante la regola della one standard error.

Una tipica implementazione in R per un dataset con $n = 200$, $p = 1000$, $s = 10$, $\rho = 0$, $\nu = 2.5$ è:

```
library(glmnet)
alpha <- 0.05
n <- 200
p <- 1000
s <- 10

beta <- c(rep(1,s), rep(0,p-s))
S <- which(beta != 0)
varType <- rep("N", p)
varType[S] <- "S"

rho <- 0
Sigma <- toeplitz(rhoˆ(0:(p-1)))
SNR <- 2.5
sigma2 <- (t(beta) %*% Sigma %*% beta) / SNR

set.seed(123)
X <- as.matrix(matrix(rnorm(n * p), n, p) %*% chol(Sigma))
y <- X %*% beta + rnorm(n, mean = 0, sd = sqrt(sigma2))

fit <- cv.glmnet(X, y)
M_hat <- which(coef(fit, s = fit$lambda.1se)[-1] != 0)

table(varType[M_hat])

m_hat        <- length(M_hat)
M_hat_typeI  <- length(setdiff(M_hat, S))
M_hat_typeII <- length(setdiff(S, M_hat))
sensitivity  <- length(intersect(M_hat, S)) / length(S)
```

Possiamo riassumere una singola realizzazione in una tabella:

$$|\hat{S}| \;\Big|\; |\hat{S} \cap N| \;\Big|\; |S \setminus \hat{S}| \;\Big|\; |\hat{S} \cap N|/|\hat{S}| \;\Big|\; |\hat{S} \cap S|/|S|$$

e calcolare stime empiriche di FWER, FDR e potenza media su B repliche.

Una comoda funzione di supporto per una sola replica è:

```
one_rep <- function() {
  X <- as.matrix(matrix(rnorm(n * p), n, p) %*% chol(Sigma))
  y <- X %*% beta + rnorm(n, mean=0, sd=sqrt(sigma2))

  fit <- cv.glmnet(X, y)
```

```
  M_hat <- which(coef(fit, s = fit$lambda.1se)[-1] != 0)

  true_pos  <- length(intersect(M_hat, S))
  false_pos <- length(setdiff(M_hat, S))

  sensitivity <- true_pos / length(S)
  fwer <- as.numeric(false_pos > 0)
  fdr  <- ifelse(length(M_hat) > 0, false_pos / length(M_hat), 0)

  c(FWER = fwer, FDR = fdr, Power = sensitivity)
}
```

Aggregando su $B = 100$ repliche si ottengono stime empiriche di FWER, FDR e potenza. In esecuzioni tipiche con questa configurazione si osservano FWER e FDR molto alti, nonostante una potenza molto elevata. Questo sottolinea che la selezione basata sul Lasso in modo ingenuo non fornisce garanzie inferenziali valide.

9.3 Procedura ingenua a due stadi

Un approccio molto diffuso ma problematico è la seguente procedura **naïve a due stadi**:

1. Si esegue il Lasso sull'intero dataset e si ottiene l'insieme attivo
$$\hat{M} = \{j : \hat{\beta}_j \neq 0\}.$$
2. Si adatta una regressione OLS di $\mathbf{y}$ sulla matrice ridotta $\mathbf{X}_{\hat{M}}$, e si calcolano i p-value standard p_j per $H_j : \beta_j = 0$ in questo modello ridotto.
3. Si definisce l'insieme di variabili "significative"
$$\hat{S} = \{j \in \hat{M} : p_j \leq \alpha\}.$$

Una simulazione in R per una replica è:

```
sim_naive <- function(n, p, s, SNR, rho = 0, alpha = 0.05) {
  Sigma <- toeplitz(rho^(0:(p-1)))
  beta  <- c(rep(1, s), rep(0, p-s))
  S     <- which(beta != 0)
  sigma2 <- (t(beta) %*% Sigma %*% beta) / SNR

  X <- as.matrix(matrix(rnorm(n * p), n, p) %*% chol(Sigma))
  y <- X %*% beta + rnorm(n, mean = 0, sd = sqrt(sigma2))

  fit <- cv.glmnet(X, y)
  M_hat <- which(coef(fit, s = fit$lambda.1se)[-1] != 0)

  s_hat <- 0
  S_hat_typeI  <- 0
  S_hat_typeII <- s
```

```
  if (length(M_hat) > 0) {
    fit_M_hat <- lm(y ~ X[, M_hat])
    pval_M_hat <- summary(fit_M_hat)$coef[-1, 4]

    S_hat <- M_hat[pval_M_hat <= alpha]
    s_hat <- length(S_hat)

    S_hat_typeI  <- length(setdiff(S_hat, S))
    S_hat_typeII <- length(setdiff(S, S_hat))

    FDP        <- ifelse(s_hat > 0, S_hat_typeI / s_hat, 0)
    sensitivity <- length(intersect(S_hat, S)) / length(S)
  }

  c(s_hat = s_hat,
    typeI = S_hat_typeI,
    typeII = S_hat_typeII,
    FDP = FDP,
    sensitivity = sensitivity)
}
```

I risultati empirici mostrano che questa procedura porta a FWER e FDR estremamente elevati, anche quando la potenza è alta. Le ragioni principali sono:

- I dati vengono usati due volte: una prima volta per selezionare $\hat{M}$ (Lasso) e una seconda volta per testare le ipotesi nel modello ridotto come se $\hat{M}$ fosse fissato. La casualità di $\hat{M}$ viene ignorata.
- Esiste inoltre un problema di molteplicità, poiché vengono effettuati più test all'interno di $\hat{M}$ senza una correzione adeguata.

9.4 Data splitting

Un modo semplice per spezzare la dipendenza tra selezione e inferenza è il **data splitting** (Cox, 1975). L'idea di base è utilizzare sottoinsiemi disgiunti dei dati per la selezione delle variabili e per i test d'ipotesi.

9.4.1 Procedura single-split

L'approccio **single-split** (Wasserman e Roeder, 2009) procede come segue. Si suddivide il dataset in due parti L e I di dimensione uguale $n_L = n_I = n/2$.

1. Si applica la procedura di selezione (ad esempio il Lasso) su $(\mathbf{X}_L, \mathbf{y}_L)$ per ottenere un insieme di variabili selezionate
$$\hat{M}_L \subseteq \{1, \dots, p\}.$$
2. Si usa la seconda metà $(\mathbf{X}_I, \mathbf{y}_I)$ per costruire i p-value:
$$p_j = \begin{cases} p_j^I & \text{se } j \in \hat{M}_L, \\ 1 & \text{se } j \notin \hat{M}_L, \end{cases}$$

dove p_j^I è il p-value per testare $H_j : \beta_j = 0$ nel modello lineare che include solo le variabili selezionate, cioè nella regressione di $\mathbf{y}_I$ su $\mathbf{X}^I_{\hat{M}_L}$.

3. Si correggono i p-value per la molteplicità, ad esempio tramite Bonferroni:

$$\tilde{p}_j = \min\{|\hat{M}_L| \cdot p_j, 1\}, \qquad j = 1, \dots, p.$$

4. Si definisce l'insieme delle variabili selezionate

$$\tilde{S} = \{j \in \hat{M}_L : \tilde{p}_j \leq \alpha\}.$$

Una implementazione diretta in R è:

```
set.seed(123)
L <- as.logical(sample(rep(0:1, each = n/2)))
I <- !L

# Selezione su L
fit_L <- cv.glmnet(X[L, ], y[L])
M_hat <- which(coef(fit_L, s = fit_L$lambda.1se)[-1] != 0)

# Inferenza su I
fit_I <- lm(y[I] ~ X[I, M_hat])
pval_I <- rep(1, p)
pval_I[M_hat] <- summary(fit_I)$coef[-1, 4]

# Insieme selezionato non aggiustato
S_hat <- M_hat[pval_I[M_hat] <= alpha]

# p-value Bonferroni
pval_tilde <- rep(1, p)
pval_tilde[M_hat] <- p.adjust(pval_I[M_hat], "bonf")

# Insieme selezionato finale
S_tilde <- M_hat[pval_tilde[M_hat] <= alpha]
```

Teorema (controllo FWER per single-split)

Supponiamo che:

1. Valga il modello lineare

$$\mathbf{y} \sim N_n(\mathbf{1}_n\beta_0 + \mathbf{X}\boldsymbol{\beta}, \sigma^2\mathbf{I}_n).$$

2. La procedura di selezione su L soddisfi la **proprietà di screening**:

$$P(\hat{M}_L \supseteq S) \geq 1 - \delta,$$

per qualche $\delta \in (0, 1)$.

3. La matrice di progetto ridotta su I abbia rango pieno,

$$\operatorname{rank}\big(\mathbf{X}^I_{\hat{M}_L}\big) = |\hat{M}_L|.$$

Allora la procedura single-split garantisce il controllo della FWER fino a δ:

$$P(\tilde{S} \cap N \neq \emptyset) \leq \alpha + \delta.$$

In altre parole, se la fase di selezione sulla prima metà non perde (quasi mai) variabili veramente attive (screening) e se la seconda metà consente inferenza lineare standard, i p-value corretti con Bonferroni assicurano che la probabilità di includere almeno un predittore nullo sia limitata da $\alpha + \delta$.

P-value lottery

Un grosso difetto del single-split è che diversi split casuali producono insiemi $\hat{M}_L$ diversi e quindi p-value diversi. Questa variabilità casuale è spesso chiamata **p-value lottery**.

Ad esempio, ripetendo la procedura single-split B volte e salvando i p-value in una matrice:

```
B <- 50
pval_matrix       <- matrix(1, nrow = B, ncol = p)
pval_matrix_tilde <- matrix(1, nrow = B, ncol = p)

set.seed(123)
for (b in 1:B) {
  split <- as.logical(sample(rep(0:1, each = n/2)))
  fit   <- cv.glmnet(X[split, ], y[split])
  M_hat <- which(coef(fit, s = fit$lambda.1se)[-1] != 0)

  fit_I <- lm(y[!split] ~ X[!split, M_hat])
  pvals <- rep(1, p)
  pvals[M_hat] <- summary(fit_I)$coef[-1, 4]

  pval_matrix[b, ]       <- pvals
  pval_matrix_tilde[b, ] <- p.adjust(pvals, "holm")
}
```

Si osserva tipicamente una forte variabilità dei p-value tra diversi split, anche per variabili realmente attive.

9.5 Procedura multi-split

Per attenuare l'instabilità del single-split, Meinshausen et al. (2009) propongono l'approccio **multi-split**.

Algoritmo

1. Per $b = 1, \ldots, B$:
   - si effettua uno split casuale dei dati in (L_b, I_b);
   - si applica la procedura single-split su (L_b, I_b) ottenendo p-value aggiustati $\tilde{p}_j^b$ per tutti $j = 1, \ldots, p$.

2. Si aggregano i p-value aggiustati sui vari split usando uno statistico robusto, ad esempio la mediana:
$$\bar{p}_j = 2 \cdot \text{median}\big(\tilde{p}_j^1, \ldots, \tilde{p}_j^B\big), \qquad j = 1, \ldots, p.$$
3. Si definisce l'insieme dei predittori selezionati come
$$\hat{S} = \{j \in \{1, \ldots, p\} : \bar{p}_j \leq \alpha\}.$$

Questa aggregazione stabilizza i risultati sui diversi split casuali e riduce drasticamente l'effetto "p-value lottery".

In R si può utilizzare il pacchetto `hdi`:

```
library(hdi)
B <- 25
set.seed(123)

fit <- multi.split(
  x = X, y = y, B = B, fraction = 0.5,
  model.selector = lasso.cv,
  ci = TRUE, ci.level = 1 - alpha,
  gamma = 0.5
)

S_hat <- which(fit$pval.corr <= alpha)
table(varType[S_hat])

# Intervalli di confidenza simultanei per i predittori selezionati
confint(fit)[S_hat, ]
```

Le procedure multi-split non solo forniscono p-value più stabili, ma permettono anche di costruire intervalli di confidenza simultanei
$$P\big(\beta_j \in [\hat{L}_j, \hat{U}_j] \; \forall j = 1, \ldots, p\big) \geq 1 - \alpha,$$
ottenendo così un'inferenza post-selezione valida con controllo della FWER per le variabili selezionate.

9.6 Conclusioni

Le simulazioni e i metodi discussi in questo capitolo mettono in evidenza le principali difficoltà nell'eseguire un'inferenza valida **dopo** la selezione di variabili in contesti ad alta dimensionalità.

- La procedura ingenua a due stadi (selezione con Lasso seguita da test t standard sul modello selezionato) ha prestazioni molto scarse dal punto di vista del controllo dell'errore. Poiché gli stessi dati vengono usati sia per la selezione sia per l'inferenza, i p-value risultanti sono fortemente distorti, portando a FWER e FDR molto elevati anche in presenza di modello veramente sparso e segnali forti.
- Il data splitting offre una soluzione concettualmente semplice, usando sottoinsiemi disgiunti dei dati per selezione e inferenza. La procedura single-split può garantire (approssimativamente) il controllo della FWER sotto una proprietà di screening, ma soffre di instabilità: split diversi producono p-value molto diversi (**p-value lottery**).

- L'approccio multi-split stabilizza l'inferenza aggregando i risultati su molti split casuali. Riduce sostanzialmente la variabilità, migliora la stabilità della selezione e fornisce p-value e intervalli di confidenza simultanei con garanzie teoriche sul tasso di errore. Pur rimanendo leggermente conservativo, rappresenta un buon compromesso tra controllo dell'errore, potenza e fattibilità computazionale.

Il messaggio fondamentale dell'inferenza post-selezione è che **un'inferenza valida richiede metodi che tengano conto esplicitamente del fatto che la selezione delle variabili è data-driven**. Ignorare la fase di selezione (come nell'approccio ingenuo) porta a tassi di falsi positivi gonfiati, mentre approcci più strutturati come il multi-split ripristinano l'affidabilità al costo di una moderata perdita di potenza. Nei problemi di regressione ad alta dimensionalità, queste procedure forniscono un quadro robusto e pratico per identificare segnali realmente rilevanti controllando i falsi positivi in modo statisticamente significativo.

Chapter 10

Stability selection

Quando lavoriamo in contesti ad alta dimensionalità, specialmente quando $p \gg n$, metodi come la regressione lasso vengono spesso utilizzati per effettuare selezione di variabili. Tuttavia, tali metodi possono essere estremamente instabili: anche una minima variazione del dataset (ad esempio rimuovendo qualche osservazione) può determinare un insieme di variabili selezionate completamente diverso.

La *stability selection* nasce proprio per affrontare questa instabilità. L'idea di base è semplice ma estremamente efficace:

1. si seleziona casualmente metà delle osservazioni;
2. si applica il lasso su questo sotto-campione;
3. si registra l'insieme delle variabili selezionate;
4. si ripete il processo molte volte;
5. si calcola, per ogni variabile, la frequenza con cui viene selezionata.

Una variabile realmente informativa verrà scelta spesso; una variabile irrilevante verrà scelta solo occasionalmente. Questa procedura consente di ottenere una selezione più stabile, più robusta e soprattutto più interpretabile.

10.1 Percorso di regolarizzazione e percorso di stabilità

Regularisation path del lasso

Il lasso dipende dal parametro di penalizzazione λ. Variando λ su una griglia Λ, si ottiene l'insieme

$$\{\hat{\beta}_j(\lambda),\ j = 1, \dots, p,\ \lambda \in \Lambda\},$$

denominato *regularisation path*. Per valori molto grandi di λ, i coefficienti vengono spinti a zero; diminuendo λ, nuove variabili entrano gradualmente nel modello.

Stability path

La stability selection non si concentra sui coefficienti stimati, ma sulla **frequenza di selezione** delle variabili in molteplici sotto-campionamenti.

Su ciascun sotto-campione di dimensione $n/2$ il lasso produce un sottoinsieme selezionato. Ripetendo l'operazione per B sotto-campionamenti, otteniamo la stima:

$$\hat{\pi}_j(\lambda) = \frac{1}{B} \sum_{b=1}^{B} \mathbf{1}\{j \in \hat{S}^{(b)}_{n/2}(\lambda)\}.$$

Il corrispondente *stability path* è costituito dalle curve $\hat{\pi}_j(\lambda)$ per $j = 1, \ldots, p$.

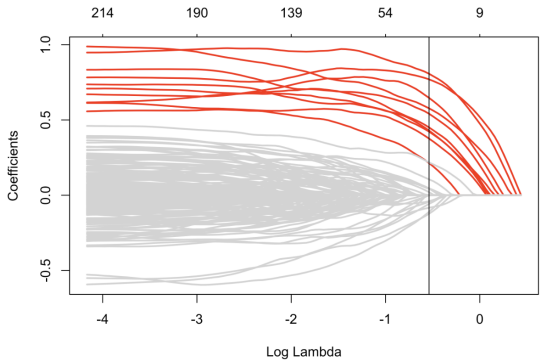
10.2 Codice R: simulazione e percorsi

Generazione del dataset simulato

```
library(glmnet)

n <- 200
p <- 1000
s <- 10

beta <- c(rep(1,s), rep(0,p-s))
S <- which(beta != 0)

rho <- 0
Sigma <- toeplitz(rhoˆ(0:(p-1)))

SNR <- 1
sigma2 <- (t(beta) %*% Sigma %*% beta) / SNR

set.seed(123)
X <- as.matrix(matrix(rnorm(n * p), n, p) %*% chol(Sigma))
y <- X %*% beta + rnorm(n, mean=0, sd=sqrt(sigma2))
```

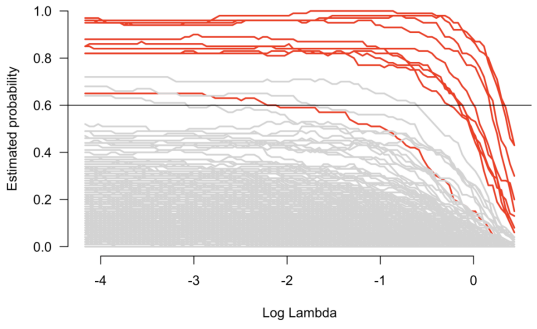
Regularisation path

```
fit <- glmnet(X, y)
Lambda <- fit$lambda
l <- cv.glmnet(X, y)$lambda.1se

S_hat <- which(coef(fit, s=l)[-1] != 0)

col <- rep("lightgray", p)
col[S] <- "red"

plot(fit, xvar="lambda", col=col, lwd=2)
abline(v=log(l))
```



Stability path

```
B = 100
S_hat_half <- array(NA, dim = c(B, p, length(Lambda)))

for (b in 1:B) {
  ind <- as.logical(sample(rep(0:1, each = n/2)))
  fit_b <- glmnet(X[ind,], y[ind], lambda = Lambda)
  S_hat_half[b,,] <- as.matrix(coef(fit_b)[-1,] != 0)
}

pi_hat <- apply(S_hat_half, 2:3, mean)

matplot(log(Lambda), t(pi_hat),
        type = "l", lty = 1, col = col,
        xlab = "Log Lambda", ylab = "Estimated probability")

tau = 0.6
abline(h = tau)
```



10.3 Interpretazione delle frequenze di selezione

Il valore $\hat{\pi}_j(\lambda)$ rappresenta la proporzione di sotto-campionamenti in cui la variabile j è stata selezionata.

- Variabili informative $\Rightarrow$ $\hat{\pi}_j(\lambda)$ elevata.
- Variabili irrilevanti $\Rightarrow$ $\hat{\pi}_j(\lambda)$ prossima a zero.

Questo rende la selezione molto più robusta rispetto all'utilizzo di un singolo sottoinsieme di dati, attenuando la sensibilità del lasso a piccole variazioni campionarie.

10.4 Controllo dell'errore

La stability selection permette anche di controllare il numero atteso di falsi positivi:

$$E\big(|\hat{S}_{\text{stab}} \cap N|\big) \leq \frac{1}{2\tau - 1} \frac{q^2}{p},$$

dove:

- N è l'insieme delle variabili irrilevanti,
- $q = E(|\hat{S}_{n/2}|)$ è il numero medio di variabili selezionate dal lasso,
- p è il numero totale di predittori.

In pratica:

- se il metodo seleziona molte variabili (q grande), il rischio di falsi positivi aumenta;
- se q è troppo piccolo, si rischia di perdere variabili realmente informative.

10.5 Complementary Pairs Stability Selection

Un'estensione particolarmente efficace è la *Complementary Pairs Stability Selection* (CPSS). L'idea è di suddividere ogni volta il dataset in due metà complementari e applicare la selezione ad entrambe. Questa procedura riduce ulteriormente la variabilità e migliora il controllo dell'errore.

Algoritmo

[H] Complementary Pairs Stability Selection (CPSS) [1]
Una procedura di selezione variabile $\hat{S}_n$, un intero $B \in N$, una soglia $\tau \in (0.5, 1)$.
$b = 1, \ldots, B$ Suddividere l'insieme $\{1, \ldots, n\}$ in due sottoinsiemi disgiunti

$$(I^{2b-1},\, I^{2b}), \qquad |I^{2b-1}| = |I^{2b}| = n/2.$$

Applicare la procedura di selezione su ciascun sottoinsieme ottenendo

$$\hat{S}^{2b-1}_{n/2} \subseteq \{1, \ldots, p\}, \qquad \hat{S}^{2b}_{n/2} \subseteq \{1, \ldots, p\}.$$

Calcolare le frequenze di selezione:

$$\hat{\pi}_j = \frac{1}{2B} \sum_{b=1}^{B} \left(\mathbf{1}\{j \in \hat{S}_{n/2}^{2b-1}\} + \mathbf{1}\{j \in \hat{S}_{n/2}^{2b}\} \right), \qquad j = 1, \ldots, p.$$

Determinare l'insieme dei predittori stabili:

$$\hat{S}_{\text{stab}} = \{j : \hat{\pi}_j \geq \tau\}.$$

Proprietà dello stimatore di frequenza

La frequenza relativa di selezione

$$\hat{\pi}_j$$

è uno stimatore non distorto della probabilità che la variabile j venga selezionata utilizzando un campione casuale di dimensione $n/2$, ossia

$$\hat{\pi}_j \quad \text{è unbiased per} \quad \pi_j^{n/2} = P\Big(j \in \hat{S}_{n/2}\Big).$$

Tuttavia, in generale $\hat{\pi}_j$ *non è unbiased* per

$$\pi_j^n = P\Big(j \in \hat{S}_n\Big) = E\Big[\, \mathbf{1}\{j \in \hat{S}_n\}\Big],$$

ossia per la probabilità di selezione che si otterrebbe con l'intero campione.

L'idea chiave della stability selection è di migliorare lo stimatore ingenuo

$$\mathbf{1}\{j \in \hat{S}_n\}$$

di π_j^n tramite il **subsampling**. Invece di basarsi su un'unica realizzazione della procedura di selezione, si considera una media su molti sotto-campionamenti:

$$\hat{\pi}_j \quad \text{ha varianza molto più piccola di} \quad \mathbf{1}\{j \in \hat{S}_n\},$$

e questa riduzione di varianza compensa ampiamente l'eventuale bias dovuto all'uso di campioni di dimensione $n/2$.

In altre parole:

- la stability selection riduce la variabilità inerente ai metodi di selezione basati su un unico dataset;
- la media su molti sotto-campioni stabilizza le stime e rende più affidabile l'identificazione dei predittori rilevanti;
- il trade-off tra bias e varianza è favorevole: meno varianza $\Rightarrow$ selezione più robusta.

Il risultato finale è che l'insieme

$$\hat{S}_{\text{stab}} = \{j : \hat{\pi}_j \geq \tau\}$$

risulta tipicamente più interpretabile, più stabile e con un minor numero di falsi positivi rispetto alla selezione diretta basata sul lasso o su altre procedure.

Teorema [Meinshausen & Bühlmann, 2010]

Assumiamo che:

1. le variabili irrilevanti siano **scambiabili**, cioè

$$\{\mathbf{1}\{j \in \hat{S}_{n/2}\},\ j \in N\} \quad \text{siano identicamente distribuite;}$$

2. la procedura di selezione non sia peggiore di una scelta casuale, ovvero:

$$\frac{E\big(|\hat{S}_{n/2} \cap S|\big)}{E\big(|\hat{S}_{n/2} \cap N|\big)} \ \geq\ \frac{|S|}{|N|}.$$

Allora, per ogni $\tau \in (1/2, 1]$, il numero atteso di falsi positivi dopo stability selection soddisfa:

$$E\big(|\hat{S}_{\text{stab}} \cap N|\big) \ \leq\ \frac{1}{2\tau - 1}\,\frac{q^2}{p},$$

dove

$$q = E\big(|\hat{S}_{n/2}|\big).$$

Il teorema fornisce un limite superiore sul numero atteso di predittori irrilevanti selezionati dalla stability selection. Alcune conseguenze importanti:

- **La scelta del numero di sotto-campionamenti B ha importanza minore**, poiché il bound non dipende da B.
- **Si può fissare direttamente** q come valore atteso delle variabili selezionate su un mezzo campione. Tuttavia, se q è troppo piccolo, la procedura seleziona solo una parte delle variabili rilevanti:

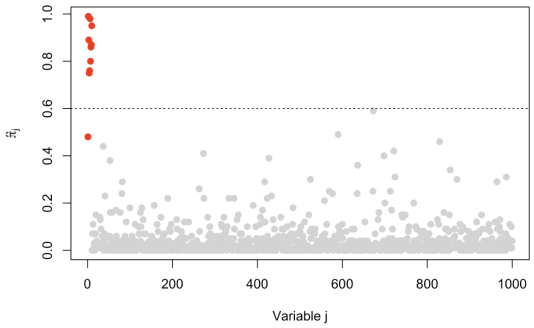
$$|\hat{S}_{\text{stab}}| \ \leq\ |\hat{S}_{n/2}| = q.$$

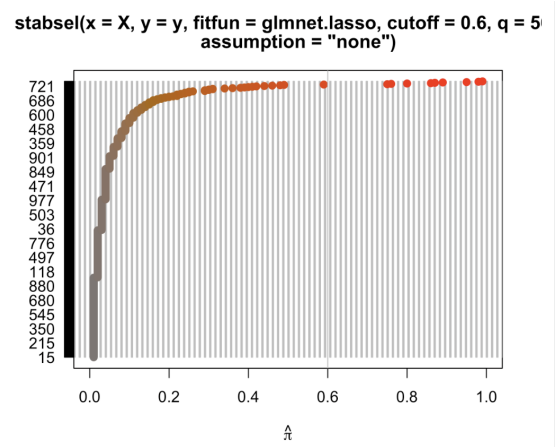
- **Esempio**. Con $p = 1000$, $q = 50$ e $\tau = 0.6$:

$$E\big(|\hat{S}_{\text{stab}} \cap N|\big) \ \leq\ 12.5.$$

Codice R

```
library(stabs)

fit <- stabsel(x = X, y = y, fitfun = glmnet.lasso,
               q = 50, cutoff = 0.6, assumption = "none")

plot(1:p, fit$max, col=col,
     xlab="Variable j", ylab=expression(hat(pi)[j]), pch=19)
abline(h = 0.6, lty = 3)


    plot(fit)
```





10.6 Conclusioni

La stability selection:

- rende la selezione delle variabili più stabile e riproducibile;
- riduce drasticamente l'inclusione di variabili irrilevanti;
- migliora l'interpretabilità dei modelli;
- può essere applicata sopra qualsiasi metodo di selezione (lasso, elastic net, forward selection, ecc.).

L'idea fondamentale è intuitiva: **una variabile realmente importante tende ad essere selezionata indipendentemente dal sotto-campione utilizzato; una variabile non importante no**. Misurare la frequenza di selezione permette di distinguere in modo naturale tra queste due categorie.

Chapter 11

Knockoff filter

In questo capitolo introduciamo il *knockoff filter*, una procedura per la selezione di variabili con controllo esplicito della False Discovery Rate (FDR) in contesti ad alta dimensionalità.

L'idea chiave è costruire, per ogni covariata originale X_j, una **copia knockoff** $\tilde{X}_j$ che:

- imita la struttura di dipendenza delle covariate originali;
- è, per costruzione, "nulla" rispetto alla risposta y.

Confrontando l'importanza di X_j e di $\tilde{X}_j$ lungo un percorso di stima (ad esempio Lasso) o tramite statistiche di importanza (ad esempio Random Forest), si ottengono statistiche W_j che godono di una simmetria fondamentale quando X_j è nullo. Tale simmetria permette di stimare la FDP in modo data–driven e di scegliere una soglia che garantisce controllo della FDR.

Esistono due varianti principali:

- **Fixed-X knockoffs**: assumono $\mathbf{X}$ fissata e a rango pieno, tipicamente con $n \geq 2p$;
- **Model-X knockoffs**: assumono un modello (o una buona approssimazione) per la distribuzione di $\mathbf{X}$ e funzionano anche quando $p > n$.

In quanto segue descriviamo prima il caso fixed-X, quindi il caso model-X con particolare attenzione al caso gaussiano.

11.1 Fixed-X knockoffs

Nel caso fixed-X, la matrice delle covariate $\mathbf{X}$ è trattata come deterministica (ad esempio risultati di un esperimento) e si suppone che:

- le colonne di $\mathbf{X}$ siano centrate e scalate, in modo che $\|X_j\|_2^2 = 1$ per ogni j;
- $\mathbf{X}$ sia a rango pieno e che $n \geq 2p$, in modo da poter costruire knockoff che rispettano vincoli di ortogonalità.

11.1.1 Costruzione delle knockoff fixed-X

Sia $\mathbf{X}$ una matrice $n \times p$ e sia

$$\Sigma = \mathbf{X}^\top \mathbf{X}$$

la matrice di correlazione (o di Gram) tra le covariate. L'obiettivo è costruire una matrice $\tilde{\mathbf{X}} \in R^{n\times p}$ tale che la matrice "combinata"

$$\mathbf{G} = \begin{bmatrix}\mathbf{X} & \tilde{\mathbf{X}}\end{bmatrix}^\top \begin{bmatrix}\mathbf{X} & \tilde{\mathbf{X}}\end{bmatrix} = \begin{bmatrix}\Sigma & \Sigma - \mathbf{D} \\ \Sigma - \mathbf{D} & \Sigma\end{bmatrix}$$

sia definita positiva per qualche matrice diagonale

$$\mathbf{D} = \mathrm{diag}(d_1, \ldots, d_p).$$

Queste condizioni implicano:

- $\tilde{\mathbf{X}}^\top \tilde{\mathbf{X}} = \mathbf{X}^\top \mathbf{X} = \Sigma$, cioè le knockoff hanno la stessa struttura di correlazione delle variabili originali;
- per $j \neq k$,
$$\tilde{X}_j^\top X_k = X_j^\top X_k,$$
quindi la correlazione incrociata con le altre variabili è preservata;
- per ogni j,
$$\tilde{X}_j^\top X_j = 1 - d_j,$$
cioè il parametro d_j controlla quanto la knockoff è simile alla variabile originale: d_j vicino a 1 rende $\tilde{X}_j$ quasi indipendente da X_j, aumentando la potenza ma rendendo più difficile la soddisfazione della positività definita.

Una costruzione classica di *knockoff equi-correlate* assume $d_j = d$ per tutti gli indici e definisce

$$\tilde{\mathbf{X}} = \mathbf{X}\left(\mathbf{I}_p - d\Sigma^{-1}\right) + \mathbf{U}\mathbf{C},$$

dove:

- $\mathbf{U} \in R^{n\times p}$ è una matrice con colonne ortonormali tale che $\mathbf{U}^\top \mathbf{X} = \mathbf{0}$;
- $\mathbf{C} \in R^{p\times p}$ è ottenuta dalla decomposizione di Cholesky di
$$\mathbf{C}^\top \mathbf{C} = 4\left(\frac{d}{2}\mathbf{I}_p - \left(\frac{d}{2}\right)^2 \Sigma^{-1}\right).$$

In pratica, pacchetti software come `knockoff` implementano due scelte principali per $\mathbf{D}$:

- `method="equi"`: $d_j = d$ uguale per tutti;
- `method="sdp"`: scelta di $\mathbf{D}$ tramite programmazione semidefinita per massimizzare $\sum_j (1 - d_j)$ sotto il vincolo $\mathbf{G} \succ 0$.

Esempio di simulazione fixed-X

Consideriamo una simulazione con:

- $n = 1000$, $p = 200$, sparsità $s = 40$;
- metà dei coefficienti non nulli uguali a 2 e metà uguali a -2;
- predittori indipendenti con $\rho = 0$ e varianza unitaria.

In R:

```
n <- 1000
p <- 200
s <- 40
set.seed(123)

beta <- sample(c(rep(2, s/2), rep(-2, s/2), rep(0, p-s)))
S <- which(beta != 0)
N <- setdiff(1:p, S)

rho   <- 0
Sigma <- toeplitz(rho^(0:(p-1)))
sigma2 <- 1

normc <- function(X, center = TRUE) {
  X.centered <- scale(X, center = center, scale = FALSE)
  X.scaled   <- scale(X.centered, center = FALSE,
                      scale = sqrt(colSums(X.centered^2)))
  X.scaled[, ]
}

X_raw <- as.matrix(matrix(rnorm(n * p), n, p) %*% chol(Sigma))
X     <- normc(X_raw, center = TRUE)
y_raw <- X %*% beta + rnorm(n, mean = 0, sd = sqrt(sigma2))
y     <- as.numeric(normc(matrix(y_raw, ncol = 1)))
```

Si costruiscono poi le knockoff $\tilde{\mathbf{X}}$ con la formula equi-correlata o usando una funzione dedicata del pacchetto `knockoff`.

11.1.2 Statistiche knockoff via Lasso

Una volta costruita la matrice aumentata

$$\mathbf{X}^{\text{aug}} = \begin{bmatrix} \mathbf{X} & \tilde{\mathbf{X}} \end{bmatrix},$$

si adatta il Lasso su $(\mathbf{X}^{\text{aug}}, \mathbf{y})$ lungo un percorso di penalizzazione $\lambda \in \Lambda$.

Per ogni variabile originale X_j e la corrispondente knockoff $\tilde{X}_j$, si definiscono:

$$Z_j = \sup\{\lambda \in \Lambda : \hat{\beta}_j(\lambda) \neq 0\}, \qquad \tilde{Z}_j = \sup\{\lambda \in \Lambda : \tilde{\beta}_j(\lambda) \neq 0\}.$$

Questi rappresentano, per ciascuna variabile, il "tempo di ingresso" nel cammino del Lasso (più grande è Z_j, più importante è la variabile).

La statistica knockoff per la variabile j è

$$W_j = \max\{Z_j, \tilde{Z}_j\} \cdot \operatorname{sign}(Z_j - \tilde{Z}_j) = \begin{cases} Z_j & \text{se } X_j \text{ entra prima di } \tilde{X}_j, \\ 0 & \text{se } Z_j = \tilde{Z}_j, \\ -\tilde{Z}_j & \text{se } \tilde{X}_j \text{ entra prima di } X_j. \end{cases}$$

Intuitivamente:

- W_j grande e positivo indica che X_j "vince" nettamente sulla sua knockoff, suggerendo un segnale reale;
- W_j grande e negativo indica che la knockoff è più importante della variabile originale, comportamento tipico di una variabile nulla;
- per variabili realmente nulle, la distribuzione di W_j è (approssimativamente) simmetrica rispetto a zero.

11.1.3 Stima della FDP e scelta della soglia

Dato un livello di soglia $\tau \geq 0$, si seleziona l'insieme

$$\hat{S}^\tau = \{j \in \{1, \dots, p\} : W_j \geq \tau\}.$$

La FDP vera è

$$\mathrm{FDP}(\hat{S}^\tau) = \frac{\#\{j \in N : W_j \geq \tau\}}{\#\{j : W_j \geq \tau\}},$$

ma il numeratore dipende dall'insieme ignoto N.

La proprietà di simmetria delle statistiche knockoff suggerisce che, per variabili nulle, i grandi valori positivi di W_j hanno la stessa probabilità dei corrispondenti grandi valori negativi. Possiamo quindi approssimare

$$\#\{j \in N : W_j \geq \tau\} \approx \#\{j \in N : W_j \leq -\tau\} \leq \#\{j : W_j \leq -\tau\},$$

che è osservabile. Una stima conservativa della FDP è quindi

$$\widehat{\mathrm{FDP}}(\hat{S}^\tau) = \frac{1 + \#\{j : W_j \leq -\tau\}}{\max\{1, \#\{j : W_j \geq \tau\}\}}.$$

L'algoritmo knockoff sceglie una soglia adattativa

$$\hat{\tau} = \min\big\{\tau > 0 : \widehat{\mathrm{FDP}}(\hat{S}^\tau) \leq \alpha\big\},$$

con $\hat{\tau} = +\infty$ se la condizione non è soddisfatta per nessun τ. L'insieme finale selezionato è $\hat{S} = \hat{S}^{\hat{\tau}}$, che gode di controllo della FDR in senso finito-campione sotto opportune condizioni.

Interpretazione della stima di FDP

L'idea chiave è che le knockoff, per costruzione, sono **variabili nulle artificiali**. Quando una knockoff "batte" la variabile originale (cioè W_j è molto negativo), stiamo osservando un evento che non può corrispondere a un vero segnale. Il numero di W_j molto negativi fornisce quindi un conteggio empirico di quanti falsi positivi ci aspetteremmo nella coda positiva. L'aggiunta di 1 al numeratore rende la stima ancora più conservativa, garantendo il controllo della FDR.

11.1.4 Variable importance statistics

L'approccio precedente si basa sul Lasso, ma la costruzione delle statistiche W_j è molto più generale: *basta disporre di una misura di importanza per ogni variabile.*

Una strategia comune è usare la **variable importance** di una Random Forest adattata al design aumentato $[\mathbf{X}, \tilde{\mathbf{X}}]$:

1. si adatta una Random Forest usando come covariate tutte le colonne di $[\mathbf{X}, \tilde{\mathbf{X}}]$;
2. si calcola l'importanza $Z_j = \mathrm{VI}(X_j)$ e $\tilde{Z}_j = \mathrm{VI}(\tilde{X}_j)$, ad esempio come somma delle riduzioni di impurità dovute agli split sulla variabile, mediata sui vari alberi;
3. si definisce la statistica knockoff
$$W_j = |Z_j| - |\tilde{Z}_j|.$$

Anche in questo caso, per variabili nulle X_j e knockoff $\tilde{X}_j$ sono intercambiabili, e quindi W_j è simmetrico rispetto a zero. Si può quindi riutilizzare lo stesso schema:

- selezione $\hat{S}^\tau = \{j : W_j \geq \tau\}$;
- stima della FDP tramite
$$\widehat{\mathrm{FDP}}(\hat{S}^\tau) = \frac{1 + \#\{j : W_j \leq -\tau\}}{\max\{1, \#\{j : W_j \geq \tau\}\}};$$
- scelta adattativa di $\hat{\tau}$ come minimo valore tale che $\widehat{\mathrm{FDP}}(\hat{S}^{\hat{\tau}}) \leq \alpha$.

Questo permette di combinare la flessibilità di metodi non lineari (come Random Forest) con la robustezza del controllo FDR offerto dal knockoff filter.

11.2 Model-X knockoffs

La versione **Model-X** del knockoff filter è pensata per scenari in cui la matrice $\mathbf{X}$ è considerata casuale e si dispone (o si assume di disporre) di un buon modello per la sua distribuzione marginale. Il punto cruciale è che:

- non è necessario specificare un modello per $y \mid \mathbf{X}$;
- è sufficiente modellare la distribuzione di $\mathbf{X}$ per costruire knockoff che mimino la dipendenza tra covariate ma siano indipendenti dalla risposta.

Null in termini di indipendenza condizionale

Si supponga di avere osservazioni i.i.d. (x_i, y_i), $i = 1, \ldots, n$, da una distribuzione ignota. Si assume però di conoscere (o di poter approssimare bene) la distribuzione marginale di x_i.

Le **variabili nulle** sono definite come

$$N = \{j \in \{1, \ldots, p\} : y \perp\!\!\!\perp x_j \mid x_{-j}\},$$

dove x_{-j} è il vettore delle covariate escluse la j-esima.

Questa definizione è puramente **non parametrica**: una variabile è nulla se, una volta condizionato sulle altre covariate, non aggiunge alcuna informazione su y. È indipendente dal fatto che la relazione vera sia lineare, non lineare o altamente complessa.

Principio delle Model-X knockoffs

Il framework Model-X separa:

- la difficoltà di modellare $y \mid \mathbf{X}$ (che può essere affrontata con qualsiasi metodo predittivo: Lasso, Random Forest, reti neurali, ...);
- il problema di costruire knockoff $\tilde{\mathbf{X}}$ usando solo la distribuzione di $\mathbf{X}$.

Se riusciamo a costruire $(\mathbf{X}, \tilde{\mathbf{X}})$ tali che:

- la distribuzione congiunta di $(\mathbf{X}, \tilde{\mathbf{X}})$ è invariata rispetto allo **scambio** di X_j con $\tilde{X}_j$ per ogni $j \in N$;
- $\tilde{\mathbf{X}}$ è condizionatamente indipendente da y dato $\mathbf{X}$,

allora le statistiche W_j costruite su $[\mathbf{X}, \tilde{\mathbf{X}}]$ godono di una proprietà di *flip-sign*: per variabili nulle, W_j ha distribuzione simmetrica e i grandi valori negativi possono essere usati per stimare la FDP tra i grandi valori positivi, come nel caso fixed-X.

11.2.1 Knockoffs in the Gaussian case

Un caso particolarmente importante è quello in cui le covariate sono gaussiane:

$$x_i \sim N_p(\mu, \Sigma), \qquad i = 1, \dots, n.$$

In questo contesto è possibile costruire esplicitamente knockoff gaussiane. Si considera la distribuzione congiunta

$$\begin{bmatrix} X \\ \tilde{X} \end{bmatrix} \sim N\left(\begin{bmatrix} \mu \\ \mu \end{bmatrix}, \begin{bmatrix} \Sigma & \Sigma - \mathbf{D} \\ \Sigma - \mathbf{D} & \Sigma \end{bmatrix} \right),$$

dove $\mathbf{D} = \mathrm{diag}(d_1, \dots, d_p)$ è tale che la matrice di covarianza complessiva sia definita positiva.

Condizionando su $X = x_i$ si ha

$$\tilde{x}_i \mid x_i \sim N\left(\mu + (\Sigma - \mathbf{D})\Sigma^{-1}(x_i - \mu),\ \Sigma - (\Sigma - \mathbf{D})\Sigma^{-1}(\Sigma - \mathbf{D})\right).$$

In pratica:

- si sceglie $\mathbf{D}$ in modo che la covarianza congiunta sia definita positiva (ad esempio con il metodo equi-correlato o via SDP);
- si generano i campioni $\tilde{x}_i$ dalla distribuzione gaussiana condizionata;
- se μ e Σ sono sconosciute, si sostituiscono con stime $\hat{\mu}$ e $\hat{\Sigma}$ (ad esempio la media campionaria e la covarianza campionaria).

In R, il pacchetto `knockoff` implementa questa costruzione:

```
library(knockoff)

set.seed(123)

mu <- rep(0, p)

gaussian_knockoffs <- function(X) create.gaussian(X, mu, Sigma)
```

```
result <- knockoff.filter(X_raw, y_raw,
                          knockoffs = gaussian_knockoffs)

print(result$selected)

fdp <- function(selected) {
  sum(beta[selected] == 0) / max(1, length(selected))
}

fdp(result$selected)
```

In questo esempio:

- le knockoff vengono costruite assumendo un modello gaussiano per le covariate;
- la funzione `knockoff.filter` calcola automaticamente statistiche W_j, stima la FDP e sceglie una soglia adattativa per selezionare le variabili;
- la FDP empirica sulle variabili selezionate è tipicamente prossima al livello nominale α scelto dall'utente.

Interpretazione del caso gaussiano

Nel caso gaussiano, la costruzione delle knockoff è particolarmente trasparente:

- la matrice di covarianza congiunta

$$\mathbf{V} = \begin{bmatrix} \Sigma & \Sigma - \mathbf{D} \\ \Sigma - \mathbf{D} & \Sigma \end{bmatrix}$$

  caratterizza completamente la distribuzione di $(X, \tilde{X})$;
- il diagonale $\mathbf{D}$ controlla quanta informazione viene "condivisa" tra X_j e $\tilde{X}_j$: d_j grandi riducono la correlazione X_j–$\tilde{X}_j$, incrementando la capacità di discriminare i segnali veri, ma rendono più restrittivo il vincolo $\mathbf{V} \succ 0$;
- una volta fissata $\mathbf{D}$, il campionamento da una distribuzione gaussiana condizionata consente di generare knockoff che preservano l'intera struttura di dipendenza tra le covariate, pur essendo indipendenti da y dato X.

Queste proprietà garantiscono la simmetria delle statistiche knockoff per gli indici nulli e, di conseguenza, il controllo rigoroso della FDR. In pratica, quando μ e Σ sono sconosciute, l'uso di stime plug-in mantiene buone proprietà inferenziali sotto ipotesi regolari e rende l'approccio Model-X knockoffs applicabile a un'ampia gamma di problemi reali.

Appendix A

Significato dei simboli

In questa appendice sono raccolte le notazioni usate nel testo, organizzate per tema.

A.1 Notazione generale

X	Vettore/matrice dei predittori (input).
Y	Variabile risposta (target).
$f(x)$	Funzione vera (sconosciuta) che lega X a Y.
$\hat{f}(x)$	Funzione stimata a partire dai dati.
ε	Errore casuale con $E[\varepsilon] = 0$, $\mathrm{Var}(\varepsilon) = \sigma^2$.
$E[\cdot]$	Aspettativa.
$\mathrm{Var}(\cdot)$	Varianza.
$\mathrm{Cov}(\cdot, \cdot)$	Covarianza.
n	Numero di osservazioni (train).
m	Numero di osservazioni (test).
p	Numero di parametri/feature (intercetta inclusa se presente).
σ^2	Varianza del rumore.
$\hat{\sigma}^2$	Stima di σ^2 (es. $RSS/(n-p)$ in OLS).

A.2 Modello di regressione lineare (OLS)

$\mathbf{X} \in R^{n \times p}$	Design matrix; $\mathbf{x}_i^\top$ sua i-esima riga.
$\mathbf{y} = (y_1, \ldots, y_n)^\top$	Vettore risposta.
$\hat{\boldsymbol{\beta}}$	$(\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$.
$\hat{\mathbf{y}}$	Predetti: $\hat{\mathbf{y}} = \mathbf{H}\mathbf{y}$, $\mathbf{H} = \mathbf{X}(\mathbf{X}^\top \mathbf{X})^{-1}\mathbf{X}^\top$.
h_{ii}	Leverage: diagonale di $\mathbf{H}$.
RSS	$\sum_{i=1}^{n} (y_i - \hat{y}_i)^2$.
TSS	$\sum_{i=1}^{n} (y_i - \bar{y})^2$, $ESS = TSS - RSS$.
$R^2,\ \bar{R}^2$	$1 - RSS/TSS,\ \ 1 - \dfrac{RSS/(n-p)}{TSS/(n-1)}$.
df	Gradi di libertà effettivi: $\mathrm{tr}(\mathbf{H}) = p$.

A.3 Metodi non parametrici e k-NN

Notazione k-NN (regressione)

k	Numero di vicini usati per la media locale.
$\mathcal{N}_k(x)$	Insieme degli indici dei k vicini più prossimi a x.
$d(\cdot,\cdot)$	Distanza su R^p (default: euclidea $\|\cdot\|_2$).
$\|\cdot\|_2$	Norma euclidea; $\|\cdot\|_{\Sigma^{-1}}$ distanza di Mahalanobis.
$\mathbf{S}$	Matrice di covarianza dei predittori (per Mahalanobis).
z	Predittore standardizzato (centering+scaling).

Stima k-NN (regressione, Fixed-X)

$\hat{f}(x)$	$\frac{1}{k}\sum_{i\in\mathcal{N}_k(x)} y_i.$
$\mathrm{Bias}(x)$	$\frac{1}{k}\sum_{i\in\mathcal{N}_k(x)} f(x_i) - f(x).$
$\mathrm{Var}\big(\hat{f}(x)\big)$	$\frac{\sigma^2}{k}$ (con errori i.i.d.).

A.4 Errori e misure di accuratezza

MSE_{Tr}	$\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}(\mathbf{x}_i))^2.$
MSE_{Te}	$\frac{1}{m}\sum_{i=1}^{m}(y_i^* - \hat{f}(\mathbf{x}_i^*))^2.$
Err_F	Errore di previsione atteso (Fixed-X).
Err_R	Errore di previsione atteso (Random-X).
$\mathrm{Opt}_F, \mathrm{Opt}_R$	Ottimismo medio (gap test–train) in Fixed-/Random-X.

A.5 Criteri di selezione del modello

C_p	$\frac{RSS}{\hat{\sigma}^2} - n + 2p = n\frac{MSE_{Tr}}{\hat{\sigma}^2} - n + 2p.$
AIC	$n\log(MSE_{Tr}) + 2p$ (gaussiano).
AIC_c	$AIC + \frac{2p(p+1)}{n-p-1}.$
BIC	$n\log(MSE_{Tr}) + (\log n)p.$

A.6 Notazione Random-X (gaussiano bivariato)

μ_x, μ_y	Medie marginali; σ_x, σ_y deviazioni standard.
ρ	Correlazione tra X e Y.
α, β	$\alpha = \mu_y - \rho(\sigma_y/\sigma_x)\mu_x,\ \ \beta = \rho(\sigma_y/\sigma_x).$
σ^2	Varianza residua: $\sigma_y^2(1-\rho^2)$.

A.7 Altri simboli

$\mathcal{N}(\mu, \sigma^2)$	Normale con media μ e varianza σ^2.
$\mathbf{I}_n$	Identità $n \times n$; $\mathrm{tr}(\cdot)$ traccia.
$\arg\min$	Argomento che minimizza una funzione.

A.8 Interpretazioni chiave

- **Bias–Varianza (Fixed-X):** $\mathrm{Err}_F(x) = \sigma^2 + \mathrm{Bias}^2(x) + \mathrm{Var}(\hat{f}(x))$.
- **k-NN:** $k\downarrow \Rightarrow$ bias$\downarrow$, var$\uparrow$; $k\uparrow \Rightarrow$ bias$\uparrow$, var$\downarrow$.
- **Curse of dimensionality:** in alta p i "vicini" sono lontani $\Rightarrow$ peggiora sia bias che var.

A.9 Smoothers lineari

$\hat{\mathbf{y}} = \mathbf{S}\mathbf{y}$	Smoother lineare con matrice di lisciamento $\mathbf{S} \in R^{n \times n}$.
df	Gradi di libertà effettivi: $\mathrm{tr}(\mathbf{S})$ (per OLS: $\mathbf{S} = \mathbf{H}$, $\mathrm{df} = p$).
S_{ii}	Leverage generalizzato (diagonale di $\mathbf{S}$).
$\mathrm{Var}(\hat{y}_i)$	$\sigma^2 \, \lVert \mathbf{S}_{i\cdot} \rVert_2^2$.

A.10 Metodi non parametrici e k-NN

Classificazione k-NN

$\hat{\pi}(x)$	$\frac{1}{k} \sum_{i \in \mathcal{N}_k(x)} 1\{y_i = 1\}$ (stima locale di $P(Y{=}1 \mid x)$).
$\hat{y}(x)$	Regola di decisione: $1\{\hat{\pi}(x) \geq \tau\}$ (tipico $\tau = 0.5$).
$\mathrm{TP}, \mathrm{FP}, \mathrm{TN}, \mathrm{FN}$	Conteggi confusione binaria.
$\mathrm{TPR}, \mathrm{FPR}$	$\mathrm{TPR} = \dfrac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$, $\mathrm{FPR} = \dfrac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}}$.
AUC	Area sotto la curva ROC (TPR vs FPR al variare di τ).

Kernel smoother (Nadaraya–Watson)

$K(\cdot)$	Kernel non negativo (es. gaussiano, Epanechnikov).
h	Banda (ampiezza finestra): controlla bias–varianza.
$w_i(x)$	$\dfrac{K\left(\frac{d(x_i, x)}{h}\right)}{\sum_j K\left(\frac{d(x_j, x)}{h}\right)}$ (pesi normalizzati).
$\hat{f}(x)$	$\sum_i w_i(x) \, y_i$.

A.11 Altri simboli

PRESS	Predicted Residual Sum of Squares per LOOCV.
$\widehat{\mathrm{Err}}_{\mathrm{LOO}}$	Stima LOOCV dell'errore medio.

A.12 Best subset, stepwise e criteri

$X \in R^{n\times p}$	Matrice dei predittori; $y \in R^n$ risposta.
β_0	Vettore coefficienti veri.
$S = \{j : \beta_{0j} \neq 0\}$, $s = \lvert S\rvert$	Indici e numerosità delle variabili attive (vere).
$M \subset \{1, \dots, p\}$	Sottoinsieme di predittori; X_M sue colonne.
B_k	Miglior modello (min RSS) di dimensione k nel best subset.
S_k	Modello alla k-esima iterazione in stepwise (forward/backward).
$\binom{p}{k}$	Numero di modelli di dimensione k.
$\mathrm{RSS}(M)$	Residual Sum of Squares del modello M.
$\mathrm{AIC}, \mathrm{BIC}, C_p$	Criteri informativi per la scelta del modello.

A.13 Lasso, ridge, elastic net e varianti

$\lVert\beta\rVert_0,\ \lVert\beta\rVert_1,\ \lVert\beta\rVert_2$	(Quasi-)norme: conteggio, L_1, L_2.
$\lambda \geq 0$	Parametro di penalizzazione.
s, k	Parametri di raggio/complessità per formulazioni vincolate.
$\hat\beta^R_\lambda$	Stima ridge.
$\hat\beta^L_\lambda$, $\hat S_\lambda = \{j : \hat\beta^L_{\lambda,j} \neq 0\}$	Stima e supporto lasso.
$\alpha \in [0,1]$	Mixing elastic net (α=1 lasso, α=0 ridge).
$\widehat\beta^{EN}_{\lambda,\alpha}$	Stima elastic net.
$\widehat A_\lambda$	Supporto selezionato dal lasso alla penalizzazione λ.
$\gamma \in [0,1]$	Peso del **Relaxed Lasso**.
$G_1, \dots, G_q$	Partizione in gruppi per Group Lasso; $m_k = \sqrt{\lvert G_k\rvert}$.

A.14 Linear smoothers e spline

$h > 0$	Bandwidth (parametro di smoothing).
$w_h(t) = \frac{1}{h} w_0(t/h)$	Kernel con scala h; w_0 densità simmetrica.
$\hat f(x)$	Stima locale di $f(x)$.
$a_r(x; h)$	Momenti pesati: $a_r = \frac{1}{n}\sum_i (x_i - x)^r w_h(x_i - x)$.
$B_{ij} = B_j(x_i)$	Matrice base per spline/regression splines.
$\Omega_{jk} = \int B_j''(x) B_k''(x)\, dx$	Matrice di roughness.
$H_\lambda = B(B^\top B + n\lambda\Omega)^{-1} B^\top$	Smoothing matrix.
GCV, LOOCV	Criteri per la scelta di λ.
$(x)_+ = \max\{x, 0\}$	Funzione parte positiva (basi troncate).
$k_1 < \cdots < k_P$	Nodi interni (knots).

A.15 Data splitting for variable selection

$L,\ I$	Indici dello split: blocco di *Learning/Selection* e blocco di *Inference*, $\lvert L\rvert = \lvert I\rvert = n/2$.
$\hat{M}_L$	Insieme di variabili selezionate sul blocco L (es. con Lasso/CV).
p_j^I	p-value su blocco I per il test $H_j : \beta_j = 0$ nel modello ristretto a $\hat{M}_L$.
$\tilde{p}_j$	p-value corretti per multiplicità su I (es. Bonferroni/Holm).
$\tilde{S}$	Insieme finale single-split: $\tilde{S} = \{j \in \hat{M}_L : \tilde{p}_j \leq \alpha\}$.
B	Numero di ripetizioni in *multi-split*.
$\tilde{p}_j^{(b)}$	p-value corretto ottenuto allo split $b = 1, \ldots, B$.
$\bar{p}_j$	Aggregato multi-split (es.) $\bar{p}_j = 2 \cdot \text{median}(\tilde{p}_j^{(1)}, \ldots, \tilde{p}_j^{(B)})$.
$\hat{S}$	Insieme finale multi-split: $\hat{S} = \{j : \bar{p}_j \leq \alpha\}$.
δ	Tolleranza di *screening*: $P(\hat{M}_L \supseteq S) \geq 1 - \delta$.
$\text{rank}(\mathbf{X}^I_{\hat{M}_L})$	Condizione di rango pieno sul blocco d'inferenza.

A.16 Stability selection

λ	Iper–parametro di penalizzazione del lasso; Λ griglia dei valori.
$\hat{\beta}_j(\lambda)$	Coefficiente lasso per la variabile j al valore λ.
$\hat{S}_n(\lambda)$	Insieme delle variabili selezionate dal lasso sul campione completo (n).
$\hat{S}^{(b)}_{n/2}(\lambda)$	Variabili selezionate al b-esimo sotto–campionamento di taglia $n/2$.
B	Numero di sotto–campionamenti (o di coppie complementari in CPSS).
$\hat{\pi}_j(\lambda)$	Frequenza di selezione stimata per la variabile j al valore λ.
τ	Soglia di selezione stabile ($\tau \in (0.5, 1]$).
$\hat{S}_{\text{stab}}(\lambda, \tau)$	Insieme dei predittori stabili: $\{j : \hat{\pi}_j(\lambda) \geq \tau\}$.
q	$q = E\Big(\lvert\hat{S}_{n/2}(\lambda)\rvert\Big)$, numerosità media selezionata a $n/2$.
p	Numero totale di predittori; S rilevanti (sconosciuti), N irrilevanti.
CPSS	Complementary Pairs Stability Selection (sotto–campioni disgiunti e complementari).

Percorsi

Regularisation path	$\{\hat{\beta}_j(\lambda) : j = 1, \ldots, p,\ \lambda \in \Lambda\}$.
Stability path	$\{\hat{\pi}_j(\lambda) : j = 1, \ldots, p,\ \lambda \in \Lambda\}$.

A.17 Knockoff filter

$\tilde{\mathbf{X}}$	Matrice delle variabili *knockoff* (copie artificiali di $\mathbf{X}$).
Σ	Matrice di Gram/covarianza delle covariate (fixed-X / model-X).
$\mathbf{D} = \text{diag}(d_1, \ldots, d_p)$	Diagonale che regola la correlazione X_j–$\tilde{X}_j$.
$Z_j,\ \tilde{Z}_j$	Misure di importanza (es. "tempo di ingresso" Lasso, o VI RF).
W_j	Statistica knockoff con proprietà di flip-sign per indici nulli.
$\hat{S}^\tau$	Insieme a soglia τ: $\hat{S}^\tau = \{j : W_j \geq \tau\}$.
$\widehat{\text{FDP}}(\hat{S}^\tau)$	Stima conservativa della FDP tramite la coda negativa dei W_j.
$\hat{\tau}$	Soglia adattativa: minimo τ con $\widehat{\text{FDP}}(\hat{S}^\tau) \leq \alpha$.
Fixed-X	Imposta $\mathbf{X}$ come fissata; richiede $n \geq 2p$ per costruzione esatta.
Model-X	Modella la distribuzione di $\mathbf{X}$; definisce null come $y \perp x_j \mid x_{-j}$.
N	Indici nulli (irrilevanti) nel senso d'indipendenza condizionata.

Appendix B

Formulario

B.1 OLS, proiezioni, varianze

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^\top \mathbf{X})^{-1}\mathbf{X}^\top \mathbf{y}, \qquad \hat{\mathbf{y}} = \mathbf{H}\mathbf{y},\ \mathbf{H} = \mathbf{X}(\mathbf{X}^\top \mathbf{X})^{-1}\mathbf{X}^\top,$$
$$RSS = \|\mathbf{y} - \hat{\mathbf{y}}\|_2^2, \quad \hat{\sigma}^2 = \frac{RSS}{n-p}, \quad \mathrm{Var}(\hat{\boldsymbol{\beta}}) = \sigma^2(\mathbf{X}^\top \mathbf{X})^{-1},$$
$$\mathrm{Var}(\hat{y}_i) = \sigma^2 h_{ii}.$$

Intervalli (nuovo $\mathbf{x}_0$).

$$\hat{y}_0 = \mathbf{x}_0^\top \hat{\boldsymbol{\beta}}, \quad \mathrm{Var}(\hat{y}_0) = \sigma^2\, \mathbf{x}_0^\top (\mathbf{X}^\top \mathbf{X})^{-1}\mathbf{x}_0,$$
$$\mathrm{PI} : \hat{y}_0 \pm t_{1-\alpha/2,\, n-p}\ \hat{\sigma}\sqrt{1 + \mathbf{x}_0^\top (\mathbf{X}^\top \mathbf{X})^{-1}\mathbf{x}_0}.$$

B.2 Decomposizioni e metriche

$$TSS = \sum_{i=1}^{n}(y_i - \bar{y})^2, \qquad R^2 = 1 - \frac{RSS}{TSS}, \qquad \bar{R}^2 = 1 - \frac{RSS/(n-p)}{TSS/(n-1)}.$$
$$\mathrm{Err}_F(x) = \sigma^2 + \mathrm{Bias}^2(x) + \mathrm{Var}(\hat{f}(x)).$$

B.3 Ottimismo ed errore atteso (Fixed-/Random-X)

$$\mathrm{Opt}_F = \frac{2\sigma^2\,\mathrm{df}}{n} \quad (\text{OLS: } \mathrm{df} = p), \qquad \widehat{\mathrm{Err}}_F = MSE_{Tr} + \frac{2\hat{\sigma}^2\,\mathrm{df}}{n},$$
$$\mathrm{Opt}_R = \mathrm{Opt}_F + \frac{\sigma^2 p}{n}\left(\frac{p+1}{n-p-1}\right), \qquad \widehat{\mathrm{Err}}_R = MSE_{Tr} + \frac{\hat{\sigma}^2 p}{n}\left(2 + \frac{p+1}{n-p-1}\right).$$

B.4 Criteri d'informazione e C_p

$$C_p = \frac{RSS}{\hat{\sigma}^2} - n + 2p = n\frac{MSE_{Tr}}{\hat{\sigma}^2} - n + 2p,$$
$$AIC = n\log(MSE_{Tr}) + 2p, \qquad AIC_c = AIC + \frac{2p(p+1)}{n-p-1},$$
$$BIC = n\log(MSE_{Tr}) + (\log n)\, p.$$

B.5 k-Nearest Neighbours (regressione)

Definizione (Fixed-X).

$$\hat{f}(x) = \frac{1}{k}\sum_{i\in\mathcal{N}_k(x)} y_i, \qquad \mathcal{N}_k(x) = k \text{ indici con distanza minima } d(x_i, x).$$

Distanze e standardizzazione.

$$d_{\ell_2}(u,v) = \|u-v\|_2, \quad d_{\text{Mah}}(u,v) = \sqrt{(u-v)^\top \mathbf{S}^{-1}(u-v)}, \quad d_{\cos}(u,v) = 1 - \frac{u^\top v}{\|u\|_2\|v\|_2}.$$

Pratica: centrare e standardizzare le feature prima di usare d_{ℓ_2}.

Bias–Varianza locali (Fixed-X, errori i.i.d.).

$$\text{Bias}(x) = \frac{1}{k}\sum_{i\in\mathcal{N}_k(x)} f(x_i) - f(x), \qquad \text{Var}\big(\hat{f}(x)\big) = \frac{\sigma^2}{k}.$$

$$\boxed{\text{Err}_F(x) = \sigma^2 + \text{Bias}^2(x) + \frac{\sigma^2}{k}}.$$

Scelta di k (CV).

$$k^\star \in \arg\min_{k\in\mathcal{K}} \widehat{\text{Err}}_{K\text{-fold}}(k), \quad \widehat{\text{Err}}_{K\text{-fold}} = \frac{1}{K}\sum_{r=1}^{K}\frac{1}{n_r}\sum_{i\in\mathcal{I}_r}\big(y_i - \hat{f}^{(-r)}(\mathbf{x}_i)\big)^2.$$

B.6 Curse of Dimensionality (implicazioni per k-NN)

- In alta dimensione, anche il "più vicino" è spesso lontano ⇒ la media locale non è più davvero locale (bias↑).
- Per ridurre la varianza servono k grandi, ma ciò allarga il vicinato (bias↑); con k piccoli la media è instabile (var↑).
- Dati richiesti crescono **esplosivamente** con p.

Mitigazioni pratiche.

- Riduzione della dimensionalità (PCA, autoencoder) o selezione di feature.
- Distanze informate (Mahalanobis) e standardizzazione sistematica.
- Considerare modelli alternativi più adatti ad alta p (ridge/lasso, RF/GBM, NN).

B.7 Diagnostica rapida per k-NN

Learning curve : valuta varianza residua vs n a k fissato;
Validation curve : valuta $MSE_{val}(k)$ per selezionare k;
Scaling check : verifica impatto della standardizzazione sulle distanze.

B.8 Equivalenze pratiche

k piccolo $\Rightarrow$ fit flessibile (bias $\downarrow$, var $\uparrow$);
k grande $\Rightarrow$ fit liscio (bias $\uparrow$, var $\downarrow$);
CV su $k \Rightarrow$ compromesso operativo ottimo in media.

B.9 Smoothers lineari: df, ottimismo e LOOCV

$$\hat{\mathbf{y}} = \mathbf{S}\mathbf{y}, \qquad \mathrm{df} = \mathrm{tr}(\mathbf{S}), \qquad \widehat{\mathrm{Err}}_F = MSE_{Tr} + \frac{2\,\hat{\sigma}^2\,\mathrm{df}}{n}.$$

$$\hat{y}_i^{(-i)} = \hat{y}_i - \frac{y_i - \hat{y}_i}{1 - S_{ii}}, \qquad \mathrm{PRESS} = \sum_{i=1}^{n} \left(\frac{y_i - \hat{y}_i}{1 - S_{ii}} \right)^2, \qquad \widehat{\mathrm{Err}}_{\mathrm{LOO}} = \frac{\mathrm{PRESS}}{n}.$$

B.10 Kernel/Nadaraya–Watson

$$\hat{f}(x) = \sum_{i=1}^{n} \frac{K\left(\frac{d(x_i,x)}{h}\right)}{\sum_{j=1}^{n} K\left(\frac{d(x_j,x)}{h}\right)}\, y_i, \quad h\downarrow\text{: bias}\downarrow,\ \text{var}\uparrow; \quad h\uparrow\text{: bias}\uparrow,\ \text{var}\downarrow.$$

B.11 k-NN come smoother lineare

$$\hat{f}(x) = \frac{1}{k} \sum_{i \in \mathcal{N}_k(x)} y_i = \sum_{i=1}^{n} w_i(x)\, y_i, \quad w_i(x) = \begin{cases} 1/k, & i \in \mathcal{N}_k(x) \\ 0, & \text{altrimenti.} \end{cases}$$

$$E[\hat{f}(x)] = \sum_i w_i(x)\, f(x_i), \qquad \mathrm{Var}(\hat{f}(x)) = \sigma^2 \sum_i w_i(x)^2 = \frac{\sigma^2}{k}.$$

B.12 k-NN in classificazione: soglia, ROC e AUC

$$\hat{\pi}(x) = \frac{1}{k} \sum_{i \in \mathcal{N}_k(x)} 1\{y_i = 1\}, \qquad \hat{y}(x) = 1\{\hat{\pi}(x) \geq \tau\}.$$

$$\mathrm{TPR}(\tau) = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}, \qquad \mathrm{FPR}(\tau) = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}}, \qquad \mathrm{AUC} = \int_0^1 \mathrm{TPR}\, d(\mathrm{FPR}).$$

B.13 LOOCV e PRESS per OLS (caso particolare)

$$\hat{y}_i^{(-i)} = \hat{y}_i - \frac{e_i}{1 - h_{ii}}, \qquad \mathrm{PRESS} = \sum_{i=1}^{n} \left(\frac{e_i}{1 - h_{ii}} \right)^2, \qquad \widehat{\mathrm{Err}}_{\mathrm{LOO}} = \frac{\mathrm{PRESS}}{n}.$$

B.14 Stima di $E[\hat{f}(x)]$ con dati denoised

$$\hat{f}_{\mathrm{bar}}(x) := \sum_i w_i(x)\, f(x_i) \;\approx\; E[\hat{f}(x)], \qquad \widehat{\mathrm{Bias}}(x) = \hat{f}_{\mathrm{bar}}(x) - f(x).$$

B.15 Best subset e stepwise

Guadagno potenziale della sparsità.

$$\frac{1}{n} E\|X\beta_0 - X\hat{\beta}_{\mathrm{OLS}}\|_2^2 = \frac{p}{n}\sigma^2, \qquad \frac{1}{n} E\|X_S\beta_{0,S} - X_S\hat{\beta}_{\mathrm{OLS},S}\|_2^2 = \frac{s}{n}\sigma^2.$$

Best subset selection (problema).

$$\min_{\beta} \; \|y - X\beta\|_2^2 \quad \text{s.t.} \quad \|\beta\|_0 \leq s \qquad \text{(equivalente a ricerca su } 2^p \text{ modelli).}$$

Algoritmo per dimensione k.

$$B_k = \arg\min_{|M|=k} \mathrm{RSS}(M), \quad k = 0, \ldots, p; \text{ poi seleziona } k \text{ via AIC/BIC/CV.}$$

Forward/backward stepwise (greedy).

$$\text{Forward: } S_{k+1} = \arg\min_{j \notin S_k} \mathrm{crit}\big(S_k \cup \{j\}\big), \quad \text{Backward: } S_{k-1} = \arg\min_{j \in S_k} \mathrm{crit}\big(S_k \setminus \{j\}\big).$$

CV corretta con selezione. La selezione va rifatta in ogni fold: per ciascun fold, esegui selezione su training e valida su validation.

B.16 Penalizzazioni: forme penalizzate e vincolate

$$\text{Ridge:} \quad \min_{\beta} \; \|y - X\beta\|_2^2 + \lambda\|\beta\|_2^2 \;\equiv\; \min_{\beta} \; \|y - X\beta\|_2^2 \text{ s.t. } \|\beta\|_2^2 \leq s.$$

$$\text{Lasso:} \quad \min_{\beta} \; \|y - X\beta\|_2^2 + \lambda\|\beta\|_1 \;\equiv\; \min_{\beta} \; \|y - X\beta\|_2^2 \text{ s.t. } \|\beta\|_1 \leq s.$$

$$\ell_0 \text{ (best subset):} \quad \min_{\beta} \; \|y - X\beta\|_2^2 + \lambda\|\beta\|_0 \;\not\equiv\; \text{vincolato in generale.}$$

B.17 Lasso ed effetti

Problema (centrando X e y).

$$\hat{\beta}^L_\lambda = \arg\min_\beta \left\{ \frac{1}{2n}\|y - X\beta\|_2^2 + \lambda\|\beta\|_1 \right\}, \quad \hat{S}_\lambda = \{j : \hat{\beta}^L_{\lambda,j} \neq 0\}.$$

Bias–varianza qualitativo.

$$\lambda \uparrow \Rightarrow \text{bias} \uparrow,\ \text{var} \downarrow; \qquad \lambda = 0 :\ \hat{\beta} = \hat{\beta}_{\text{OLS}}.$$

B.18 One–SE rule (selezione di λ)

$$\lambda_{1\text{SE}} = \arg\min_{\lambda \in \mathcal{C}} \ \text{df}(\lambda) \quad \text{con} \quad \mathcal{C} = \{\lambda :\ \text{CVErr}(\lambda) \leq \text{CVErr}(\lambda_{\min}) + SE(\text{CVErr})\}.$$

B.19 Elastic Net

$$\widehat{\beta}^{EN}_{\lambda,\alpha} = \arg\min_\beta \left\{ \frac{1}{2n}\|y - X\beta\|_2^2 + \lambda\Big((1-\alpha)\tfrac{1}{2}\|\beta\|_2^2 + \alpha\|\beta\|_1\Big) \right\}, \quad \alpha \in [0,1].$$

$$(1-\alpha)\lambda > 0 \ \Rightarrow\ \text{unicità della soluzione (stretta convessità).}$$

B.20 Relaxed e Adaptive Lasso

Relaxed Lasso.

$$\widehat{A}_\lambda = \{j : \hat{\beta}^L_{\lambda,j} \neq 0\}, \quad \hat{\beta}^{RELAX}_\lambda = \gamma\, \hat{\beta}^L_\lambda + (1-\gamma)\, \hat{\beta}^{OLS}_{\widehat{A}_\lambda},\ \gamma \in [0,1].$$

Adaptive Lasso (idea di base).

$$\hat{\beta}^{adapt}_\lambda = \arg \min_{\beta:\ \beta_{(\widehat{S}_{init})^c}=0} \left\{ \frac{1}{2n}\|y - X\beta\|_2^2 + \lambda \sum_{k \in \widehat{S}_{init}} \frac{|\beta_k|}{|\hat{\beta}^{init}_k|} \right\}.$$

B.21 Group Lasso

$$\min_\beta \ \frac{1}{2n}\|y - X\beta\|_2^2 + \lambda \sum_{k=1}^{q} m_k \|\beta_{G_k}\|_2, \qquad m_k = \sqrt{|G_k|}.$$

Selezione a gruppi: $\hat{\beta}_{G_k} = 0$ oppure $\hat{\beta}_{G_k} \neq 0$ per tutti $j \in G_k$.

B.22 Local regression (linear/quadratic)

Local linear (stima).

$$(\hat{\alpha}, \hat{\beta}) = \arg\min_{\alpha,\beta} \sum_{i=1}^{n} \big(y_i - \alpha - \beta(x_i - x)\big)^2 w_h(x_i - x), \quad \hat{f}(x) = \hat{\alpha}.$$

Forma filtro lineare:

$$\hat{f}(x) = \frac{\frac{1}{n}\sum_i \{a_2 - a_1(x_i - x)\}\, w_h(x_i - x)\, y_i}{a_2 a_0 - a_1^2}, \quad a_r = \frac{1}{n}\sum_i (x_i - x)^r w_h(x_i - x).$$

Local quadratic.

$$\min_{\alpha,\beta,\gamma} \sum_i \big(y_i - \alpha - \beta(x_i - x) - \gamma(x_i - x)^2\big)^2 w_h(x_i - x), \quad \hat{f}(x) = \hat{\alpha}.$$

B.23 Bias–varianza (ordine)

$$E[\hat{f}(x)] \approx f(x) + \tfrac{h^2}{2}\sigma_w^2 f''(x), \qquad \mathrm{Var}[\hat{f}(x)] \approx \frac{\sigma_w^2}{nh}\,\frac{g(x)}{\alpha(w)}.$$

Ottimo asintotico:

$$h_{\mathrm{opt}}(x) = \left(\frac{1}{n}\,\frac{\sigma^2\,\alpha(w)}{\sigma_w^4\, f''(x)^2\, g(x)}\right)^{1/5}.$$

B.24 Smoothing splines

Problema di ottimizzazione.

$$\min_{f \in C^2} \; \frac{1}{n}\sum_{i=1}^{n}(y_i - f(x_i))^2 + \lambda \int (f''(x))^2\, dx \quad \Rightarrow \quad f = \text{natural cubic spline (nodi su } x_i).$$

Forma matriciale.

$$\hat{\beta}_\lambda = (B^\top B + n\lambda\Omega)^{-1} B^\top y, \qquad \hat{y} = H_\lambda y, \;\; H_\lambda = B(B^\top B + n\lambda\Omega)^{-1}B^\top.$$

LOOCV rapida.

$$\mathrm{CV}(\lambda) = \sum_{i=1}^{n}\left(\frac{y_i - \hat{f}^{\,\lambda}(x_i)}{1 - \{H_\lambda\}_{ii}}\right)^2.$$

GCV.

$$\mathrm{GCV}(\lambda) = \sum_{i=1}^{n}\left(\frac{y_i - \hat{f}^{\,\lambda}(x_i)}{1 - \mathrm{tr}(H_\lambda)/n}\right)^2.$$

B.25 Basi troncate e natural cubic splines

Truncated power basis (ordine M con nodi k_p).

$$\{1, x, \ldots, x^M, (x - k_1)_+^M, \ldots, (x - k_P)_+^M\}, \quad \mathrm{df} = 1 + M + P.$$

Natural cubic (lineare fuori dai boundary knots). Base equivalente con combinazioni di $(x - k_j)_+^3$ che impongono linearità oltre k_1, k_P.

B.26 Data splitting for variable selection

Single-split (Wasserman–Roeder).

$$\tilde{p}_j = \begin{cases} \min\{|\hat{M}_L| \cdot p_j^I,\, 1\}, & j \in \hat{M}_L, \\ 1, & j \notin \hat{M}_L, \end{cases} \qquad \tilde{S} = \{j \in \hat{M}_L :\ \tilde{p}_j \le \alpha\}.$$

Controllo FWER (screening + rango). Se $P(\hat{M}_L \supseteq S) \ge 1 - \delta$ e $\operatorname{rank}(\mathbf{X}^I_{\hat{M}_L}) = |\hat{M}_L|$, allora

$$P(\tilde{S} \cap N \neq \emptyset) \ \le\ \alpha + \delta.$$

Multi-split (aggregazione robusta).

$$\bar{p}_j \ =\ 2 \cdot \operatorname{median}\big(\tilde{p}_j^{(1)}, \ldots, \tilde{p}_j^{(B)}\big), \qquad \hat{S} \ =\ \{j : \bar{p}_j \le \alpha\}.$$

B.27 Stability selection: definizioni operative

Frequenze di selezione (subsampling).

$$\hat{\pi}_j(\lambda) = \frac{1}{B} \sum_{b=1}^{B} \mathbf{1}\Big\{ j \in \hat{S}^{(b)}_{n/2}(\lambda) \Big\}, \qquad \hat{S}_{\text{stab}}(\lambda, \tau) = \{ j : \hat{\pi}_j(\lambda) \ge \tau \}.$$

Proprietà di base.

$E\big[\hat{\pi}_j(\lambda)\big] = \pi_j^{n/2}(\lambda) = P\Big(j \in \hat{S}_{n/2}(\lambda)\Big)$, (unbiased per $n/2$; in generale non unbiased per n).

B.28 Percorsi: regularisation e stability

Regularisation path: $\{\hat{\beta}_j(\lambda)\}_{\lambda \in \Lambda}$; Stability path: $\{\hat{\pi}_j(\lambda)\}_{\lambda \in \Lambda}$.

$\lambda\uparrow$: modelli più sparsi; $\lambda\downarrow$: entrano più variabili.

B.29 CPSS (Complementary Pairs Stability Selection)

Definizione delle coppie. Per $b = 1, \ldots, B$,

$$(I^{2b-1}, I^{2b}) \text{ disgiunti}, \quad |I^{2b-1}| = |I^{2b}| = n/2,$$

applico la selezione su ciascun mezzo–campione e definisco

$$\hat{\pi}_j \ =\ \frac{1}{2B} \sum_{b=1}^{B} \big(\mathbf{1}\{ j \in \hat{S}^{2b-1}_{n/2} \} + \mathbf{1}\{ j \in \hat{S}^{2b}_{n/2} \} \big).$$

B.30 Controllo dell'errore (bound di falsi positivi)

Sotto scambiabilità delle irrilevanti e "non–peggio–di–casuale", per ogni $\tau \in (1/2, 1]$:

$$\boxed{E\Big(|\hat{S}_{\text{stab}} \cap N|\Big) \;\le\; \frac{1}{2\tau - 1}\,\frac{q^2}{p}} \qquad \text{con} \quad q = E\Big(|\hat{S}_{n/2}|\Big)\,.$$

Conseguenze pratiche.

$$q \text{ grande} \Rightarrow \text{maggiore rischio FP}; \qquad q \text{ piccolo} \Rightarrow \text{maggiore rischio FN}.$$

$$\text{Regola operativa:} \quad \hat{S}_{\text{stab}} = \{j : \hat{\pi}_j \ge \tau\}, \quad \tau \in [0.6, 0.9] \text{ (tipico)}.$$

B.31 Selezione stabile sopra il lasso

Dato $\lambda \in \Lambda$,

$$\hat{S}_{\text{lasso}}(\lambda) = \{j : \hat{\beta}_j(\lambda) \neq 0\}, \qquad \hat{S}_{\text{stab}}(\lambda, \tau) = \{j : \hat{\pi}_j(\lambda) \ge \tau\}.$$

La stability selection agisce come "meta–procedura" che trasforma una selezione instabile $\mathbf{1}\{j \in \hat{S}_n(\lambda)\}$ in una decisione più robusta via media su sotto–campioni.

B.32 Scelte tipiche dei parametri

B (decine–centinaia, p.es. 50–200), $\tau \in (0.6, 0.9)$, q prefissato per controllare il bound, Λ da glmnet/CV.

B.33 Knockoff filter

Costruzione (fixed-X): vincolo semidefinito.

$$\mathbf{G} = \begin{bmatrix} \Sigma & \Sigma - \mathbf{D} \\ \Sigma - \mathbf{D} & \Sigma \end{bmatrix} \succ 0, \quad \tilde{\mathbf{X}}^\top \tilde{\mathbf{X}} = \Sigma,\; \tilde{X}_j^\top X_k = X_j^\top X_k \;(j \neq k),\; \tilde{X}_j^\top X_j = 1 - d_j.$$

Statistiche knockoff (es. Lasso).

$$Z_j = \sup\{\lambda :\; \hat{\beta}_j(\lambda) \neq 0\}, \quad \tilde{Z}_j = \sup\{\lambda :\; \tilde{\beta}_j(\lambda) \neq 0\}, \qquad W_j = \max\{Z_j, \tilde{Z}_j\}\,\text{sign}(Z_j - \tilde{Z}_j).$$

Stima FDP e soglia adattativa.

$$\widehat{\text{FDP}}(\hat{S}^\tau) \;=\; \frac{1 + \#\{j :\; W_j \le -\tau\}}{\max\{1,\; \#\{j :\; W_j \ge \tau\}\}}, \qquad \hat{\tau} \;=\; \min\{\tau > 0 :\; \widehat{\text{FDP}}(\hat{S}^\tau) \le \alpha\}.$$

$$\boxed{\hat{S} = \hat{S}^{\hat{\tau}} = \{j :\; W_j \ge \hat{\tau}\} \;\;\Rightarrow\;\; \text{FDR controllata (condizioni standard)}}.$$

Model-X (gaussiano, formula condizionata). Se $x \sim \mathcal{N}_p(\mu, \Sigma)$ e $\begin{bmatrix} X \\ \tilde{X} \end{bmatrix} \sim \mathcal{N}\Big(\begin{bmatrix} \mu \\ \mu \end{bmatrix}, \begin{bmatrix} \Sigma & \Sigma - \mathbf{D} \\ \Sigma - \mathbf{D} & \Sigma \end{bmatrix}\Big)$, allora

$$\tilde{x} \mid X = x \;\sim\; \mathcal{N}\Big(\mu + (\Sigma - \mathbf{D})\Sigma^{-1}(x - \mu),\; \Sigma - (\Sigma - \mathbf{D})\Sigma^{-1}(\Sigma - \mathbf{D})\Big),$$

con $\mathbf{D}$ scelto per garantire la positività definita della covarianza congiunta.