

Aspetti computazionali

Data Mining

CLAMSES - University of Milano-Bicocca

Aldo Solari

Riferimenti bibliografici

- AS §2.2.1
- HTF §3.2.3
- LKA §2.3, §2.5, §2.6

Table of Contents

Le equazioni normali

Soluzione via scomposizione di Cholesky

Ortogonalizzazione di Gram-Schmidt

Soluzione via scomposizione QR

Nei libri di testo viene spesso presentata la soluzione ad un certo problema con una formula matematica. Ad esempio, si consideri la soluzione

$$\hat{\beta} = (X^T X)^{-1} X y$$

Tuttavia la traduzione diretta di questo tipo di formule in codice non è sempre consigliabile perché ci sono molti aspetti problematici dei computer che semplicemente non sono rilevanti quando si scrivono le cose su carta

I potenziali problemi computazionali che possono emergere sono

- *Overflow* :

Quando i numeri diventano troppo grandi, non possono essere rappresentati su un computer e quindi spesso vengono prodotte NA

- *Underflow* :

Simile all'overflow, i numeri possono diventare troppo piccoli per essere rappresentati dai computer, provocando errori o avvisi o calcoli imprecisi

- *Dipendenza quasi lineare* :

Il computer (che ha una precisione finita) può confondere una dipendenza quasi lineare per una dipendenza lineare

Sia y il vettore delle risposte, X la matrice del disegno e β il vettore dei parametri in un modello lineare. Lo stimatore OLS (Ordinary Least Squares)

$$\hat{\beta} = (X^T X)^{-1} X y$$

Questa soluzione può essere tradotta in codice R come

```
betahat <- solve(t(X) %*% X) %*% t(X) %*% y
```

Tuttavia non è consigliabile calcolare il valore di $\hat{\beta}$ in questo modo.

La ragione principale è che il calcolo dell'inversa di $X^T X$ è molto costoso dal punto di vista computazionale ed è un'operazione potenzialmente instabile su un computer quando c'è un'elevata multicollinearità tra i predittori.

Inoltre, per calcolare $\hat{\beta}$ non abbiamo bisogno dell'inversa di $X^T X$, quindi perché calcolarla?

Le equazioni normali

Basta infatti considerare le equazioni normali

$$X^T X \beta = X^T y$$

che rappresentano un caso particolare di un generico sistema di equazioni

$$Ax = b$$

La funzione `solve(A, b, ...)` risolve questo sistema di equazioni, dove A è una matrice quadrata e b può essere un vettore o una matrice.

Se b non viene specificato, allora diventa la matrice identità I , quindi il problema si traduce in $Ax = I$, ovvero trovare l'inversa di A .

Quindi date le equazioni normali

$$X^T X \beta = X^T y$$

basta risolvere

```
solve(crossprod(X), crossprod(X, y))
```

Questo approccio ha il vantaggio di essere più stabile numericamente e di essere molto più veloce


```

set.seed(123)
n <- 500
p <- 100
X <- matrix(rnorm(n*p), ncol=p)
y <- rnorm(n)
library(microbenchmark)
microbenchmark(solve(t(X) %*% X) %*% t(X) %*% y,
               solve(crossprod(X), crossprod(X, y)))

```

Unit: milliseconds

			expr	min	lq
			solve(t(X) %*% X) %*% t(X) %*% y	7.791536	8.095084
			solve(crossprod(X), crossprod(X, y))	3.053421	3.196977
mean	median	uq	max	neval	cld
9.817109	8.362006	10.211855	37.96365	100	b
3.869815	3.297905	3.560238	23.04237	100	a

Problemi di multicollinearità

Per ottenere una situazione di multicollinearità, possiamo aggiungere una colonna a X che è molto simile (ma non identica) alla prima colonna di X

```
W <- cbind(X, X[, 1] + rnorm(n, sd = 1e-10))
```

L'approccio "diretto" fallisce quando c'è elevata multicollinearità

```
solve(crossprod(W), crossprod(W, y))
```

La difficoltà nel risolvere un sistema di equazioni lineari può essere descritto dal numero di condizionamento κ (condition number), definito come il rapporto tra il più grande e più piccolo valore singolare di $X^T X$. Se κ è molto grande, siamo in presenza di un problema mal condizionato (ill-conditioned, un problema dove le soluzioni sono molto sensibili a piccole perturbazioni dei dati iniziali)

Table of Contents

Le equazioni normali

Soluzione via scomposizione di Cholesky

Ortogonalizzazione di Gram-Schmidt

Soluzione via scomposizione QR

La scomposizione di Cholesky

Poichè nel nostro caso la matrice $A = X^T X$ è simmetrica, e se X è a rango pieno, è anche definita positiva, allora possiamo considerare la decomposizione di Cholesky

$$A = LL^T$$

dove L è una matrice triangolare inferiore.

Con questa decomposizione possiamo scrivere

$$LL^T x = b$$

$$Lz = b$$

dove $z = L^T x$ è la nuova incognita.

Quando nel generico sistema di equazioni $Ax = b$ la matrice A è triangolare inferiore (superiore), si può applicare l'algoritmo di sostituzione in avanti, denominato *forwardsolve* (l'algoritmo di sostituzione in indietro, denominato *backsolve*).

Algoritmo di backsolve

Si consideri la seguente matrice triangolare superiore

$$\begin{bmatrix} l_{1,1} & l_{1,2} & l_{1,3} \\ 0 & l_{2,2} & l_{2,3} \\ 0 & 0 & l_{3,3} \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

Dall'ultima riga (equazione) risulta

$$z_3 = \frac{b_3}{l_{3,3}}$$

La seconda riga (equazione) coinvolge solamente z_2 e z_3 , quindi

$$z_2 = \frac{b_2}{l_{2,2}} - \frac{l_{2,3}z_3}{l_{2,2}}$$

Infine

$$z_1 = \frac{b_1}{l_{1,1}} - \frac{l_{1,2}z_2}{l_{1,1}} - \frac{l_{1,3}z_3}{l_{1,1}}$$

Forwardsolve e backsolve

L'algoritmo di forwardsolve utilizza sostanzialmente la stessa tecnica per matrici triangolari inferiori. Per il nostro problema, possiamo scrivere

$$X^T X x = X^T y$$

$$L L^T x = b$$

$$L z = b$$

dove L è una matrice triangolare inferiore
 $p \times p$

Prima si risolve $L z = b$ con

`forwardsolve(L, b)`

Poi si risolve $L^T x = z$ con

`backsolve(t(L), forwardsolve(L, b))`

```
# Calcolare la stima OSL con la decomposizione di Cholesky
#
# Argomenti:
# X: la matrice del disegno
# y: il vettore risposta
#
# Ritorna:
# Il vettore hatbeta di lunghezza ncol(X).
ols_chol <- function(X, y)
{
  XtX <- crossprod(X)
  Xty <- crossprod(X, y)
  L <- t(chol(XtX))
  betahat <- backsolve(t(L), forwardsolve(L, Xty))
  betahat
}
```

Table of Contents

Le equazioni normali

Soluzione via scomposizione di Cholesky

Ortogonalizzazione di Gram-Schmidt

Soluzione via scomposizione QR

Matrice di proiezione

Sia $L \subseteq \mathbb{R}^n$ un sottospazio vettoriale, i.e. $L = \text{span}\{v_1, \dots, v_k\}$ per dei vettori $v_1, \dots, v_k \in \mathbb{R}^n$. Sia $V \in \mathbb{R}^{n \times k}$ la matrice che contiene v_1, \dots, v_k come sue colonne, allora

$$\text{span}\{v_1, \dots, v_k\} = \{a_1 v_1 + \dots + a_k v_k : a_1, \dots, a_k \in \mathbb{R}\}$$

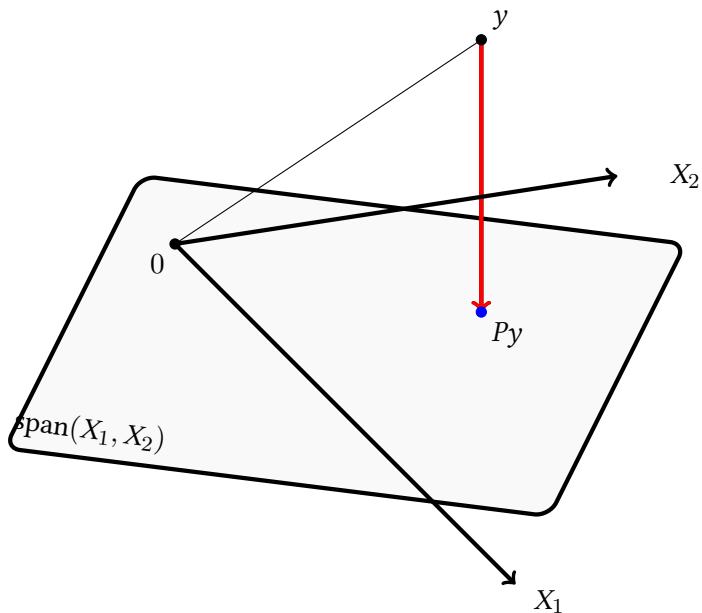
Sia $F(y) = Py$ la funzione (lineare) che proietta $y \in \mathbb{R}^n$ su L , dove il test $P \in \mathbb{R}^{n \times n}$ è la matrice di proiezione su L . La matrice P è simmetrica, i.e. $P^t = P$ e idempotente, i.e. $P^2 = P$. Inoltre abbiamo

$$Py = y \quad \forall y \in L, \quad Py = 0 \quad \forall y \perp L$$

Per ogni sottospazio L , il suo complemento ortogonale è

$$L^\perp = \{y \in \mathbb{R}^n : y \perp L\} = \{y \in \mathbb{R}^n : y \perp v \text{ per qualsiasi } v \in L\}$$

Geometria del modello lineare



Geometria del modello lineare

La stima

$$\hat{y} = X(X^tX)^{-1}X^ty = Py$$

del modello lineare è esattamente la proiezione di y nel sottospazio $\text{span}\{X_1, \dots, X_p\}$

Il vettore dei residui

$$y - \hat{y} = (I - P)y = P^\perp y$$

è la proiezione di y nel sottospazio $\text{span}\{X_1, \dots, X_p\}^\perp$ quindi $y - \hat{y}$ è ortogonale a ciascun X_1, \dots, X_p

Regressione univariata

Sia $\langle a, b \rangle = a^t b = \sum_{i=1}^n a_i b_i$ il prodotto scalare (Euclidean inner product) di $a, b \in \mathbb{R}^n$

Sia $\|a\|_2^2 = a^t a = \sum_{i=1}^n a_i^2$ la norma Euclidea al quadrato

Data la variabile X_j (j -sima colonna di X), il coefficiente di regressione di y su X_j è

$$\hat{\beta}_j = \frac{\langle X_j, y \rangle}{\|X_j\|_2^2}$$

Se X_1, \dots, X_p sono ortogonali, allora è anche il coefficiente di X_j della regressione multivariata di y su X_1, \dots, X_p

Regressione univariata con intercetta

Per la regressione di y su $\mathbf{1}$, $x \in \mathbb{R}^n$, possiamo scrivere

$$\hat{\beta}_1 = \frac{\langle x - \bar{x}\mathbf{1}, y \rangle}{\|x - \bar{x}\mathbf{1}\|_2^2}$$

Questo risultato si può ottenere in due passi

1. Regressione di x su $\mathbf{1}$, ottenendo il coefficiente

$$\frac{\langle \mathbf{1}, y \rangle}{\|\mathbf{1}\|_2^2} = \frac{\langle \mathbf{1}, y \rangle}{n} = \bar{x}$$

e il vettore dei residui

$$z = x - \bar{x}\mathbf{1} \in \mathbb{R}^n$$

2. Regressione di y su z , ottenendo il coefficiente

$$\hat{\beta}_1 = \frac{\langle z, y \rangle}{\|z\|_2^2} = \frac{\langle x - \bar{x}\mathbf{1}, y \rangle}{\|x - \bar{x}\mathbf{1}\|_2^2}$$

Regressione multivariata via ortogonalizzazioni successive

1. Sia $Z_1 = X_1$
2. Per $j = 2, \dots, p$:
Regressione di X_j su Z_1, \dots, Z_{j-1} per ottenere i coefficienti

$$\hat{\gamma}_{jk} = \frac{\langle Z_k, X_j \rangle}{\|Z_k\|_2^2}, \quad k = 1, \dots, j-1$$

e il vettore dei residui

$$Z_j = X_j - \sum_{k=1}^{j-1} \hat{\gamma}_{jk} Z_k$$

3. Regressione di y su Z_p per ottenere il coefficiente $\hat{\beta}_p$, ovvero il coefficiente di X_p nella regressione di y su X_1, \dots, X_p

Per ogni j , dalla definizione $Z_j = X_j - \sum_{k=1}^{j-1} \hat{\gamma}_{jk} Z_k$ segue che ciascun Z_j è una combinazione lineare di X_1, \dots, X_j , quindi

$$\text{span}(Z_1, \dots, Z_j) \subseteq \text{span}(X_1, \dots, X_j)$$

Riarrangiando i termini, si può mostrare che ciascun X_j è una combinazione lineare di Z_1, \dots, Z_j , quindi

$$\text{span}(X_1, \dots, X_j) \subseteq \text{span}(Z_1, \dots, Z_j)$$

Poiché $\text{span}(X_1, \dots, X_j) = \text{span}(Z_1, \dots, Z_j)$, la regressione di y su X_1, \dots, X_p è la stessa di la regressione di y su Z_1, \dots, Z_p . Sia

$$\hat{y} = c_1 Z_1 + \dots + c_p Z_p$$

Poichè Z_1, \dots, Z_p sono ortogonali, i coefficienti c_1, \dots, c_p corrispondono a quelli della regressione univariata, in particolare

$$c_p = \frac{\langle Z_p, y \rangle}{\|Z_p\|_2^2} = \hat{\beta}_p$$

Abbiamo

$$\hat{y} = c_1 Z_1 + \dots + c_{p-1} Z_{p-1} + \hat{\beta}_p Z_p$$

Si noti che la variabile X_p compare solo attraverso Z_p :

$$Z_p = X_p - \sum_{k=1}^{p-1} \hat{\gamma}_{jk} Z_k$$

Quindi possiamo scrivere, per certe costanti a_1, \dots, a_{p-1}

$$\hat{y} = a_1 X_1 + \dots + a_{p-1} X_{p-1} + \hat{\beta}_p X_p$$

quindi $\hat{\beta}$ è il coefficiente di X_p nella regressione di y su X_1, \dots, X_p

(Non-normalized) Classical Gram-Schmidt

Require: $X \in \mathbb{R}^{n \times p}$

for $j \leftarrow 1$ to p **do**

$Z_j = X_j$

for $k \leftarrow 1$ to $j - 1$ **do**

$Z_j \leftarrow Z_j - \left(\frac{\langle Z_k, X_j \rangle}{\|Z_k\|_2^2} \right) Z_k$

end for

end for

return orthogonal set Z_1, \dots, Z_p

```
classicGS = function(X){  
  X <- as.matrix(X)  
  p <- ncol(X)  
  n <- nrow(X)  
  Z <- matrix(0, n, p)  
  for (j in 1:p){  
    Zj = X[,j]  
    if (j > 1) {  
      for (k in 1:(j-1)){  
        coef = crossprod(Z[,k], X[,j]) / crossprod(Z[,k])  
        Zj = Zj - coef * Z[,k]  
      }  
    }  
    Z[,j] <- Zj  
  }  
  return(Z)  
}
```

Table of Contents

Le equazioni normali

Soluzione via scomposizione di Cholesky

Ortogonalizzazione di Gram-Schmidt

Soluzione via scomposizione QR

La decomposizione QR

Require: $X \in \mathbb{R}^{n \times p}$

Initialize $R \in \mathbb{R}^{p \times p}$: $R \leftarrow 0$

for $j \leftarrow 1$ to p **do**

$Z_j = X_j$

for $k \leftarrow 1$ to $j - 1$ **do**

$R_{jk} \leftarrow \langle Q_k, X_j \rangle$

$Z_j \leftarrow Z_j - R_{jk}Q_k$

end for

$R_{jj} \leftarrow \|Z_j\|_2$

$Q_j \leftarrow Z_j / R_{jj}$

end for

return orthonormal set Q_1, \dots, Q_p , upper triangular matrix R

```

factorizationQR = function(X){
  p <- ncol(X)
  n <- nrow(X)
  Q <- matrix(0, n, p)
  R <- matrix(0, p, p)
  for (j in 1:p){
    Zj = X[,j]
    if (j > 1) {
      for (k in 1:(j-1)){
        R[k,j] = crossprod(Q[,k], X[,j])
        Zj = Zj - R[k,j] * Z[,k]
      }
    }
    R[j,j] <- sqrt( crossprod(Zj) )
    Q[,j] <- Zj / R[j,j]
  }
  return(list(Q=Q, R=R))
}

```

La decomposizione QR (ridotta) prevede

$$\underset{n \times p}{X} = \underset{n \times p}{Q} \underset{p \times p}{R}$$

dove Q è una matrice ortogonale tale che $Q^T Q = I$, ed R è una matrice triangolare superiore, quindi

$$\begin{aligned} X^T X \beta &= X^T y \\ R^T Q^T Q R \beta &= R^T Q^T y \\ R^T R \beta &= R^T Q^T y \\ R \beta &= Q^T y \end{aligned}$$

Possiamo quindi risolvere il sistema con l'algoritmo backsolve senza dover calcolare $X^T X$, che potrebbe risultare numericamente instabile.

```
# Calcolare le stime OLS con la decomposizione QR
#
# Argomenti:
# X: la matrice del disegno
# y: il vettore risposta
#
# Ritorna:
# Il vettore hatbeta di lunghezza ncol(X).
ols_qr <-
function(X, y)
{
  qr_obj <- qr(X)
  Q <- qr.Q(qr_obj)
  R <- qr.R(qr_obj)
  Qty <- crossprod(Q, y)
  betahat <- backsolve(R, Qty)
  betahat
}
```

Costo computazionale

La funzione `lm` utilizza la decomposizione QR (programmata in C e richiamata dalla funzione). Questa soluzione ha un costo computazionale di

$$2np^2$$

operazioni, rispetto al costo

$$p^3 + np^2/2$$

della scomposizione di Cholesky.