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

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

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

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 precisone finita) può confondere una dipendenza quasi lineare per una dipendenza lineare

Sia $\underset{n\times 1}{y}$ il vettore delle risposte, $\underset{n\times p}{X}$ la matrice del disegno e $\underset{p\times 1}{\beta}$ il vettore dei parametri in un modello lineare. Lo stimatore OLS (Ordinary Least Squares)

$$\hat{\beta} = (X^{\top}X)^{-1}Xy$$

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 $X^{\top}X$ 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 $X^{\top}X$, quindi perché calcolarla?

Le equazioni normali

Basta infatti considerare le equazioni normali

$$X^\mathsf{T} X\beta = X^\mathsf{T} y$$

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

$$X^{\mathsf{T}} X\beta = X^{\mathsf{T}} y$$

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 $X^\top X$. Se κ è molto grande, siamo in presenza di un un problema mal condizionato (ill-conditioned, un problema dove le soluzioni sono molto sensibili a piccole perturbazioni dei dati iniziali)

Table of Contents

La scomposizione di Cholesky

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

$$A = LL^\top$$

dove L è una matrice triangolare inferiore.

Con questa decomposizione possiamo scrivere

$$\begin{aligned} LL^\top x &= b \\ Lz &= b \end{aligned}$$

dove $z = L^\top x$ 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

$$\begin{aligned} X^\top X x &= X^\top y \\ LL^\top x &= b \\ Lz &= b \end{aligned}$$

dove $\underset{p \times p}{L}$ è una matrice triangolare inferiore

Prima si risolve $Lz = b$ con

```
forwardsolve(L, b)
```

Poi si risolve $L^\top x = z$ 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

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

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

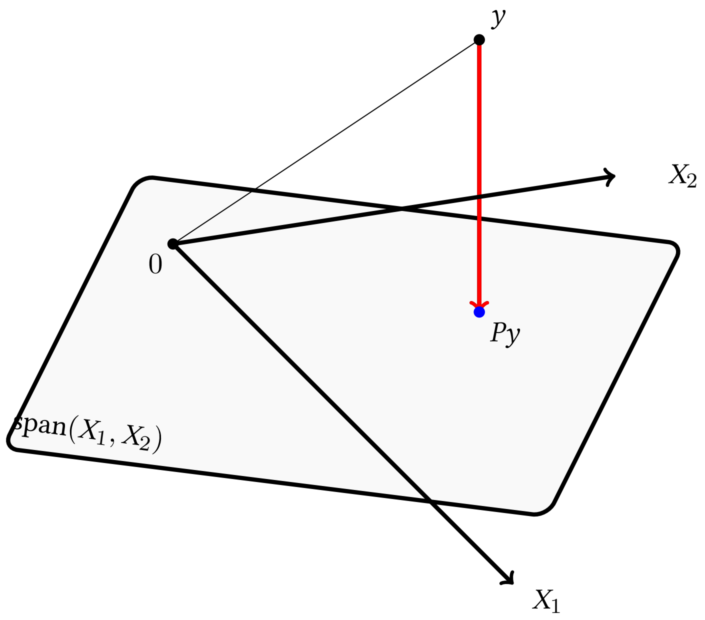
Sia $F(y) = Py$ la funzione (lineare) che proietta $y \in \mathrm{R}^n$ su L , dove il test $P \in \mathbb{R}^{n \times n}$ è la matrice di proiezione su L. La matrice P è simmetrica, i.e. $P^t = P$ e idempotente, i.e. $P^2 = P$. Inoltre abbiamo

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

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

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

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

Il vettore dei residui

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

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

Regressione univariata

Sia $\langle a, b \rangle = a^t b = \sum_{i=1}^n a_i b_i$ il 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 $X_1, \ldots, X_p$ sono ortogonali, allora è anche il coefficiente di X_j della regressione multivariata di y su $X_1, \ldots, X_p$

Regressione univariata con intercetta

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

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

Questo risultato si può ottenere in due passi

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

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

   e il vettore dei residui

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

2. Regressione di y su z, ottenendo il coefficiente

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

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 $X_1, \ldots, X_j$, quindi

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

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

$$\text{spam}(X_1, \ldots, X_j) \subseteq \text{spam}(Z_1, \ldots, Z_j)$$

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

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

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

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

Abbiamo

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

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 $a_1, \ldots, a_{p-1}$

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

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

(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**
      $Z_j \leftarrow Z_j - (\frac{\langle Z_k, X_j \rangle}{\|Z_k\|_2^2}) Z_k$
    end for
  end for
  return orthogonal set $Z_1, \ldots, Z_p$

```
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

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 $Q^\top Q = I$, ed R è una matrice triangolare superiore, quindi

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

Possiamo quindi risolvere il sistema con l'algoritmo backsolve senza dover calcolare $X^\top X$, 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.

Quando n è grande

Si consideri il caso $n \gg p$. Sia

$$W = X^t X, \quad u = X^t y$$

di dimensione $p \times p$ e $p \times 1$, quindi

$$\hat{\beta} = W^{-1} u$$

Otteniamo

$$W = \sum_{i=1}^{n} x_i x_i^t, \quad u = \sum_{i=1}^{n} x_i y_i$$

dove x_i^t è l'i-sima riga di X

Possiamo scrivere

$$W_{(j)} = W_{(j-1)} + x_j x_j^t, \quad u_{(j)} = u_{(j-1)} + x_j y_j \quad j = 2, \ldots, n$$

dove $W_{(j)}$ è la matrice formata dalla somma dei primi j elementi di W, e analogamente per $u_{(j)}$, partendo da

$$W_{(1)} = x_1 x_1^t, \quad u_{(1)} = x_1 y_1$$

Ora è chiaro che W e u possono essere calcolati leggendo i dati di un singolo record alla volta e aumentando gradualmente le somme man mano che i dati vengono letti

Stime ricorsive

Si ipotizzi di avere a disposizione un flusso di dati continuo nel tempo.

Per ogni nuova osservazione ricevuta, bisognerà aggiornare $\hat{\beta}$.

Quindi, ogni volta si dovrà invertire la matrice $W = (X^t X)$, dove X è, ad ogni passaggio, aumentata della nuova unità statistica. La procedura finora considerata perviene alla stima del nuovo $\hat{\beta}$ ma, specialmente quando il numero p di covariate è elevato e l'acquisizione del dato è molto rapida, potrebbe risultare troppo onerosa.

L'algoritmo dei minimi quadrati ricorsivi può aiutare nella soluzione di questo problema introducendo una tecnica di stima più efficiente in termini computazionali.

Si ipotizzi di aver calcolato $\hat{\beta}_{(n)}$, cioè la stima ai minimi quadrati di β sulle prime n osservazioni. Per semplicità di notazione, si consideri:

$$V_{(n)} = W_{(n)}^{-1} = \left[X_{(n)}^t \, X_{(n)}\right]^{-1}$$

al giungere della $(n+1)$-esima osservazione, formata da $\left[y_{(n+1)}, x_{(n+1)}^t\right]$, dove $x_{(n+1)}$ è un vettore di dimensioni $p \times 1$, si otterrà:

$$X_{(n+1)} = \begin{bmatrix} X_{(n)} \\ x_{(n+1)}^t \end{bmatrix} \longrightarrow W_{(n+1)} = X_{(n+1)}^t X_{(n+1)} = X_{(n)}^t X_{(n)} + x_{(n+1)} x_{(n+1)}^t$$

Inoltre, ricordando la formula di Sherman-Morrison:

$$V_{(n+1)} = W_{(n+1)}^{-1} = V_{(n)} - h \, V_{(n)} x_{(n+1)} x_{(n+1)}^t V_{(n)},$$

con $h = \left[1 + x_{(n+1)}^t V_{(n)} x_{(n+1)}\right]^{-1} \in \mathbb{R}$. Sapendo inoltre che:

$$\hat{\beta}_{(n+1)} = V_{(n+1)} \, X_{(n+1)}^t \, y_{(n+1)},$$

Si procede

$$X^t_{(n+1)}\, y_{(n+1)} \;=\; X^t_{(n)}\, y_{(n)} + x_{(n+1)}\, y_{(n+1)},$$

quindi:

$$\begin{aligned}
\hat{\beta}_{(n+1)} \;&=\; \left[V_{(n)} \;-\; h\, V_{(n)}\, x_{(n+1)}\;\; x^t_{(n+1)} V_{(n)}\right]\left[X^t_{(n)}\, y_{(n)} + x_{(n+1)}\, y_{(n+1)}\right] \\
&=\; \hat{\beta}_{(n)} + h\, V_{(n)}\, x_{(n+1)}\, \Big[y_{(n+1)}\, h^{-1} - x^t_{(n+1)}\hat{\beta}_{(n)} - \tilde{x}^t_{(n+1)} V_{(n)}\, x_{(n+1)}
\end{aligned}$$

Raccogliendo $y_{(n+1)}$, ricordando come è stato definito h e semplificando, finalmente si trova l'espressione:

$$\hat{\beta}_{(n+1)} \;=\; \hat{\beta}_{(n)} + h\, V_{(n)}\, x_{(n+1)}\, \left[y_{(n+1)} - x^t_{(n+1)}\hat{\beta}_{(n)}\right].$$

Si può osservare che $e_{(n+1)} = \left[y_{(n+1)} - x^t_{(n+1)}\hat{\beta}_{(n)}\right]$ rappresenta l'errore di previsione commesso utilizzando $\hat{\beta}_{(n)}$ per stimare $y_{(n+1)}$. Per questo motivo, si dice che lo stimatore ai minimi quadrati ricorsivi "apprende dagli errori che commette".

Per stimare ricorsivamente il parametro β, dunque, la dispendiosa inversione di W viene eseguita solo una volta durante l'inizializzazione.

Quest'ultima avviene stimando $\hat{\beta}_{(p)}$ e quindi $V_{(p)}$ sulle prime p osservazioni o addirittura, in caso di n sufficientemente elevato, assegnando a $\hat{\beta}_{(0)}$ un vettore nullo ed a $V_{(0)}$ la matrice identità di ordine p, evitando totalmente l'inversione di W.

Si può, inoltre, dimostrare una formula ricorsiva anche per il calcolo della devianza residua:

$$Q_{(n+1)}\left(\hat{\beta}_{(n+1)}\right) \;=\; Q_{(n)}\left(\hat{\beta}_{(n)}\right) + h\, e^2_{(n+1)}.$$

Initialize $\underset{p\times p}{W} \leftarrow 0$, $\underset{p\times 1}{u} \leftarrow 0$, $Q \leftarrow 0$

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

  leggi osservazione n-sima: $x = x_n$, $y = y_n$

  $W \leftarrow W + xx^t$

  $u \leftarrow u + xy$

end for

$V \leftarrow W^{-1}$

$\hat{\beta} = Vu$

for $n \leftarrow p+1, p+2, \ldots$ **do**

  leggi osservazione n-sima: $x = x_n$, $y = y_n$

  $h \leftarrow 1/(1 + x^t V x)$

  $e \leftarrow y - x^t \hat{\beta}$

  $\hat{\beta} \leftarrow \hat{\beta} + hVxe$

  $V \leftarrow V - hVxx^t V$

  $Q \leftarrow Q + he^2$

  $s^2 \leftarrow Q/(n-p)$

  $se(\hat{\beta}) \leftarrow s\,\mathrm{diag}(V)^{1/2}$

end for