How bad is Cholesky decomposition for OLS?

Asked 4 years, 7 months ago Modified 2 years, 5 months ago Viewed 997 times

5

I've heard that I should never use Cholesky decomposition for OLS as it's super unstable, so I thought I'd try a very simple example to see how bad it could be.

First let's set up the problem in R:

```
X <- rbind(c(1, 1), c(1.00001, 1))
kappa(t(X) %*% X)
```

The condition number is 160002098109! Oooh boy, coefficients will be off by at least a billion for sure!

```
y <- rnorm(2)
L <- chol(t(X) %*% X)
z <- solve(t(L), t(X) %*% y)

result <- data.frame(chol = solve(L, z), qr = coefficients(lm.fit(X, y, method = 'qr')))
result

        chol        qr
x1 -148230.9 -148231.0
x2  148231.9  148232.1
```

...almost no difference. What happened? If the condition number is so big that no method can handle it well, what is the range of condition numbers at which the Cholesky solution is in fact appreciably less stable than with QR decomposition?

regression machine-learning least-squares

Share Cite Edit Follow Flag

asked Jan 27, 2019 at 5:10

badmax
2,161 13 25

1 Answer

Sorted by: Highest score (default)

What you are seeing is exactly what one would expect. The condition number of your matrix X is about 10^6. Double precision floating point calculations give about 18 figures of accuracy

number of significant figures because forming X^TX squares the condition number.) Hence the coefficients from the two algorithms should start to differ in the 6th significant figure, which is exactly what you see. Your results show that for `x2` the relative difference between the two algorithms is about

$$0.2/148232 = 1.35 \times 10^{-6},$$

which matches the condition number you started out with surprisingly well.

The condition number of X is a reasonable guide to precision when the residuals are small. If we make sure that the exact coefficients are 1 and 1.00001 by

```
> y <- X %*% c(1,1.00001)
```

and rerun your calculation we get

```
> result
                 chol               qr
x1 0.9999833544961814 0.999999999968327
x2 1.0000266455870466 1.000010000031673
```

confirming that QR is in fact correct to 12 significant figures and Cholesky is correct to 6 significant figures.

The fitted values from Cholesky are generally more precise than the estimated coefficients. If the fitted values are more important to you than the coefficients, and that will often be so in highly collinear cases, then Cholesky is usually fine. Cholesky is also about as good as QR when the residuals are very large, but this is the hardest case for both algorithms. My PhD supervisor was Australia's foremost numerical analyst (Mike Osborne) and he used to say that Cholesky was not as bad as it's made out to be.

The full sensitivity analysis for Cholesky vs QR is given in Section 5.3.8 of Golub and Van Loan (1996), and is very much more complex than the simple calculation I used above. The sensitivity of Cholesky is roughly proportion to $\kappa + \rho\kappa^2$ where κ is the condition number of X and ρ is a theoretical quantity that is almost impossible to compute in practice. ρ depends on the size of the residuals-- it is roughly proportional to but much smaller than the average squared residual. The sensitivity of QR is somewhat better than Cholesky although not always as good as I have suggested above. Golub and Van Loan remark:

> At the very minimum, this discussion should convince you how difficult it can be to choose the "right" algorithm!

Reference

Share Cite Edit Follow Flag

edited Apr 11, 2021 at 1:14

answered Jan 27, 2019 at 6:51

Gordon Smyth
11.9k 1 34 53

1 @Ben That's a good point. I had followed the spelling used in the R function help page. I was also influence by the fact that "-ski" is the prefered Anglicization of the original slavic surname that Cholesky derives from. But you are absolutely right that Cholesky himself spelt his name with a "y" including on the original manuscript (bibnum.education.fr/sites/default/files/...) so I've changed the spelling in my answer. Thanks. – Gordon Smyth Apr 10, 2021 at 7:27