Leave-Out Leverage

An Updating Formula for the Projection Matrix

Nikolas Kuschnig*

November 4, 2023

This report presents a tractable updating formula for elements of the projection matrix after one or more observations are removed. The formula is computationally cheap, and can be applied recursively to efficiently compute the leave-K-out *leverage, and generalize various leave-one-out influence measures to higher orders. Implementations of the formula are made available.*

Consider a linear model $\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{e}$, where $\mathbf{y}, \mathbf{e} \in \mathbb{R}^N$ and $\mathbf{X} \in \mathbb{R}^{N \times M}$. The *projection matrix* $\mathbf{H} \in \mathbb{R}^{N \times N}$ maps the responses $\mathbf{y}$ onto the column space of $\mathbf{X}$, 'putting the hat' on the fitted values $\hat{\mathbf{y}}$. It is given by

$$\mathbf{H} = \mathbf{X}\,(\mathbf{X}'\mathbf{X})^{-1}\,\mathbf{X}', \tag{1}$$

with elements given by $h_{ij} = \mathbf{x}_i\,(\mathbf{X}'\mathbf{X})^{-1}\,\mathbf{x}_j'$.

This matrix is a fundamental building block for numerous quantities of interest; the most prominent quantity is the *leverage* of an observation i, which is directly given by the diagonal element h_{ii}. The leverage features prominently in measures such as Cook's distance [2], but also in convenient rank-one updating formulae for the influence of single observations on $\hat{\boldsymbol{\beta}}$ and $\hat{\mathbf{y}}$ [1], which facilitate, e.g., Jackknife estimation or checks for the sensitivity to certain subsamples of the data.

Computation and storage of the $N \times N$ projection matrix can be prohibitive, and there is currently no convenient way to update the leverage or other elements of the projection matrix after one or more observations are removed. One consequence of this is that

*Correspondence to nikolas.kuschnig@wu.ac.at, at the Vienna University of Economics and Business. Part of the research presented here was conducted at the University of California, Berkeley.

most sensitivity checks are constrained to rank-one updates, making them susceptible to Type II errors [3]. Approaches that go beyond rank-one updates are limited by the need to re-compute the projection matrix.

An updating formula for the projection matrix

Here, I present a tractable formula to update elements of the projection matrix after one or more observations are removed. This formula holds for all of its elements, meaning that it can be applied recursively.

Proposition 1. *Let* $\mathbf{H}$ *be the projection matrix in Equation 1, let* $\mathbf{S}_x = \mathbf{X}'\mathbf{X}$ *for notational convenience, let the superscript* $!r$ *denote the removal of observation* r*, and assume that* $\mathbf{S}_x^{!r}$ *if of full rank. Then*

$$h_{ij}^{!r} = \mathbf{x}_i \left(\mathbf{S}_x^{!r}\right)^{-1} \mathbf{x}_j' = h_{ij} + \frac{h_{ir}h_{rj}}{1 - h_{rr}}. \tag{2}$$

Before I prove this result directly, it is useful to recall two results. First, the removal of observation r from $\mathbf{S}_x$ is equivalent to a rank-one update, i.e., $\mathbf{S}_x^{!r} = \mathbf{S}_x - \mathbf{x}_r'\mathbf{x}_r$. Second, the Sherman-Morrison formula [4], given by

$$(\mathbf{A} - \mathbf{u}\mathbf{v}')^{-1} = \mathbf{A}^{-1} + \frac{\mathbf{A}^{-1}\mathbf{u}\mathbf{v}'\mathbf{A}^{-1}}{1 - \mathbf{v}'\mathbf{A}^{-1}\mathbf{u}}, \tag{3}$$

provides a closed-form expression for a rank-one update. On their own, these two results provide a way to avoid the re-computation of $\mathbf{S}_x^{!r}$ and its inverse, but provide no remedy for expensive matrix multiplications and memory requirements. However, as I will show next, the expression of interest can be simplified further.

Proof. Using aforementioned results, we can express Equation 2 as

$$\begin{aligned} h_{ij}^{!r} &= \mathbf{x}_i \left[\mathbf{S}_x^{-1} + \frac{\mathbf{S}_x^{-1}\mathbf{x}_r'\mathbf{x}_r\mathbf{S}_x^{-1}}{1 - \mathbf{x}_r\mathbf{S}_x^{-1}\mathbf{x}_r'}\right]\mathbf{x}_j', \\ &= \mathbf{x}_i\mathbf{S}_x^{-1}\mathbf{x}_j' + \mathbf{x}_i\frac{\mathbf{S}_x^{-1}\mathbf{x}_r'\,\mathbf{x}_r\mathbf{S}_x^{-1}}{1 - \mathbf{x}_r\mathbf{S}_x^{-1}\mathbf{x}_r'}\mathbf{x}_j', \\ &= h_{ij} + \frac{h_{ir}h_{rj}}{1 - h_{rr}}. \end{aligned}$$

The first step follows from Equation 3, and the second step removes the brackets. The final step follows from the definition of the projection matrix. □

Discussion

Equation 2 is useful to analyze theoretical properties of higher order leave-out statistics, and allows for the efficient computation of many quantities of interest. Here, I introduce the notion of a *leave-one-out leverage* matrix, briefly discuss the extension of existing statistics to higher orders, and highlight the computational benefits of the formula.

Leave-one-out leverage Consider a matrix $\mathbf{\Pi}$ whose elements are given by $\pi_{ij} = h_{ii}^{!j}$. The diagonal elements π_{ii} can be understood as the *counterfactual leverage* of observations if they were not included in the sample. This gives rise to a notion of *extreme* leverage as one that results in a pathological counterfactual, i.e., $h_{ii}^{!i} \geqslant 1$. This cutoff is exceeded for $h_{ii} \geqslant 0.5$, and is exceeded by the correction term alone for $h_{ii} \geqslant \frac{\sqrt{5}-1}{2}$.

Meanwhile, the off-diagonal elements of $\mathbf{\Pi}$ can be seen as links between observations, and $\mathbf{\Pi}$ can be interpreted as a network between observations. This opens up new tools, such as measures of centrality, for sensitivity analyses. Moreover, Equation 2 shows that the leverage of an observation — and thus various types of influence — is fundamentally determined and limited by its full-sample leverage, and the similarity to other (high-leverage) observations. This insight allows practitioners to limit or tweak the search space when assessing the influence of (sets of) observations.

Leave-K-out coefficients The leverage and leave-K-out updates for it are holistic measures of influence. In practice, other measures that are more targeted towards quantities of interest can prove to be more insightful. One example for such a measure is the influence of an observation on coefficient estimates. For rank-one updates, this measure exists in a computationally convenient form, and is known as DFBETA [1].

With the help of Equation 2, we can easily generalize this result to higher orders. Consider the formula for a rank-two update

$$\hat{\boldsymbol{\beta}}^{!r} - \hat{\boldsymbol{\beta}}^{!r,i} = \frac{\left(\mathbf{S}_x^{!r}\right)^{-1} \mathbf{x}_i' e_i^{!r}}{1 - h_{ii}^{!r}}. \tag{4}$$

Of the three terms to compute, $h_{ii}^{!r}$ has been the prohibitive one. Computing or updating $\mathbf{S}_x^{!r}$ or its inverse is comparatively inexpensive, while $e_i^{!r}$ is readily available from a well-known result for the influence on fitted values [1]. With Equation 2, the last piece

of the puzzle becomes trivial to compute, allowing for new adaptive algorithms that search the sample space more efficiently.

Computation A major appeal of Equation 2 concerns the ease of computation. All ingredients for a rank-one update of the projection matrix are directly available, or can be computed recursively for updates of higher order. The necessary computations are simple, and can easily be subset to elements of interest.

To illustrate, consider the case of computing the leave-one-out leverage of all observations after removing an observation r. In terms of memory, two vectors of length N are required. The first one holds the diagonal elements of $\mathbf{H}$, and the second one, which can be overwritten on-the-fly, its r'th row (or column). In terms of operations, the fact that elements of the correction term, $\frac{h_{ir}^2}{1-h_{rr}}$, are strictly positive even allows for the use of logarithms and scalar addition and subtraction.[1]

An implementation for the full leave-out-leverage matrix, $\mathbf{\Pi}$, is given here.

```
# Leave-one-out leverage in row i for removals of column j
LOO_verage <- diag(H) + t(exp(log(H^2) - log(1 - diag(H))))
```

The leverage of a single observation can easily be computed as below.

```
loo_verage <- function(i, r) # Looverage of i after removing r
    H[i, i] + exp(log(H[i, r]^2) - log(1 - H[r, r]))
```

Implementations for arbitrary entries of the projection matrix have to rely on scalar multiplication and division for the correction term, since its elements may be negative. Higher-order updates, such as the leave-K-out leverage, can be implemented in a similar way, for instance, using a recursive approach.

```
lko_H <- function(i, j, R) { # H[i, j] after removing set R
    if(length(R) == 0) return(H[i, j])
    Recall(i, j, R[-1]) +
        Recall(i, R[1], R[-1]) * Recall(j, R[1], R[-1]) /
        (1 - Recall(R[1], R[1], R[-1]))
}
```

Notably, the details of the implementation are important for higher-order updates. Naive recursive implementations, such as the one presented here, scale particularly poorly for larger sets. Additional code is provided in the Appendix.

[1] A total of $\mathcal{O}(N)$ operations is necessary. A conventional approach to computing this quantity relies on the $N \times M$ matrix $\mathbf{X}$, and requires $\mathcal{O}(NM^2)$ multiplications.

Conclusion

In this report, I presented a tractable formula to update elements of the projection matrix after one or more observations are removed. I proposed analysis of the leave-one-out leverage as an alternative way to assess the influence of observations, and showed how to generalize rank-one update formulae, such as the influence of observations on regression coefficients, to higher orders. Lastly, I provided implementations of the proposed updating formulae.

References

[1] Belsley, D. A., E. Kuh, and R. E. Welsch (1980): *Regression diagnostics: Identifying influential data and sources of collinearity*: John Wiley & Sons, 10.1002/0471725153.

[2] Cook, R. D. (1979): "Influential observations in linear regression," *Journal of the American Statistical Association*, 74, 169–174, 10.2307/2286747.

[3] Kuschnig, N., G. Zens, and J. Crespo Cuaresma (2021): "Hidden in plain sight: Influential sets in linear regression," *CESifo Working Paper*, https://kuschnig.eu/files/wp_influential-sets_wip.pdf.

[4] Sherman, J., and W. J. Morrison (1950): "Adjustment of an inverse matrix corresponding to a change in one element of a given matrix," *The Annals of Mathematical Statistics*, 21, 124–127, 10.1214/aoms/1177729893.

Appendix

A brief illustration with code, based on the example by Kuschnig et al. [3], is presented below. A script version of the code is available here.

```
# Projection matrix updates ---
set.seed(753)
prec <- .Machine$double.eps

N <- 54 # Simulate data
x <- as.matrix(c(rnorm(N), rnorm(3, 6, 0.25), rnorm(3, 8, 0.25)))
y <- c(
  x[seq(N)] * -0.5 + rnorm(N, 0, 1),
  x[seq(N + 1, N + 3)] * 0.1 + rnorm(3, 0, 0.1),
  x[seq(N + 4, N + 6)] * 0.4 + rnorm(3, 0, 0.1)
)
m <- lm(y ~ x - 1)

get_H <- \(x) x %*% solve(crossprod(x), t(x))

loo_verage <- function(i, r) # Looverage of after removing r
  H[i, i] + exp(log(H[i, r]^2) - log(1 - H[r, r]))

loo_H <- function(i, j = i, r) # H[i, j] after removing r
  H[i, j] + H[i, r] * H[j, r] / (1 - H[r, r])

lko_H <- function(i, j = i, R) { # H[i, j] after removing set R
  if(length(R) == 0) return(H[i, j])
  Recall(i, j, R[-1]) +
    Recall(i, R[1], R[-1]) * Recall(j, R[1], R[-1]) /
    (1 - Recall(R[1], R[1], R[-1]))
}

H <- get_H(x) # Projection matrix

# Leave-one-out leverage in row i for removals of column j
```

```r
LOO_verage <- diag(H) + t(exp(log(H^2) - log(1 - diag(H))))

# Updating formula for the inverse to get the coefficients
update_inv <- function(XX_inv, X_rm) {
  XX_inv + (XX_inv %*% crossprod(X_rm) %*% XX_inv) /
    as.numeric(1 - X_rm %*% tcrossprod(XX_inv, X_rm))
}

# Leave-one-out DFBETA
Sx_inv <- chol2inv(chol(crossprod(x)))
dfe <- resid(m) + t(H * resid(m) / (1 - diag(H))) # Updated error
LOO_dfbeta <- matrix(NA_real_, NROW(x), NROW(x))
for(j in seq_len(NROW(x))) {
  LOO_dfbeta[, j] <- (t(update_inv(Sx_inv, x[j, , drop = FALSE]) %*%
    t(x * dfe[, j])) / (1 - LOO_verage[, j]))[, 1L]
}

# Check equivalence
abs(loo_verage(1, r = 2) -
  get_H(x[-2, ])[1, 1]) < prec
abs(loo_H(1, 2, r = N) -
  get_H(x[-N, ])[1, 2]) < prec
abs(lko_H(1, 2, R = seq(3, 7)) -
  get_H(x[-seq(3, 7), ])[1, 2]) < prec
all(abs(LOO_verage[-N, N] -
  diag(get_H(x[-N, ]))) < prec)
all(abs(LOO_dfbeta[-1, 1] -
  dfbeta(lm(y[-1] ~ x[-1, ] - 1))[, 1]) < prec)
abs(coef(m)[1] - dfbeta(m)[1, 1] - LOO_dfbeta[2, 1] -
  coef(lm(y[-1:-2] ~ x[-1:-2, ] - 1))[1]) < prec
```