covMcd() – Considerations about Generalizing the FastMCD

Martin Mächler

January 4, 2021

1 Introduction

The context is robust multivariate "location and scatter" estimation, which corresponds to estimating the first two moments in cases they exist. We assume data and a model

$$x_i \in \mathbb{R}^p, \quad i = 1, 2, \ldots, n \tag{1}$$

$$x_i \sim \mathcal{F}(\mu, \Sigma), \quad \text{i.i.d.}; \quad \mu \in \mathbb{R}^p, \quad \Sigma \in \mathbb{R}^{p \times p}, \text{ positive definite}, \tag{2}$$

where a conceptual null model is the p-dimensional normal distribution. One typical assumption is that $\mathcal{F}$ is a mixture with the majority component ("good data") being $\mathcal{N}_p(\mu, \Sigma)$ and other components modeling "the outliers".

In other words, we want estimates $\left(\hat{\mu}, \hat{\Sigma}\right)$ which should be close to the true "good data" (μ, Σ) — and do not say more here.

2 MCD and "the Fast" MCD (= fastmcd) Algorithm

The `robustbase` R package has featured a function `covMcd()` since early on (Feb. 2006) and that has been an interface to the Fortran routine provided by the original authors and (partly) described in Rousseeuw and van Driessen (1999). We describe shortly how the algorithm works, partly building on the documentation provided in the source (R, S, and Fortran) codes:

The minimum covariance determinant estimator of location and scatter (MCD) implemented in `covMcd()` is similar to R function `cov.mcd()` in `MASS`. The ("theoretical") MCD looks for the $h = h_\alpha (> 1/2)$ out of n observations whose classical covariance matrix has the lowest possible determinant. In more detail, we will use $h = h_\alpha = h(\alpha, n, p) \approx \alpha \cdot (n+p+1)$, where as Rousseeuw and van Driessen (1999) mainly use (the default) $\alpha = \frac{1}{2}$, where $h = h(1/2, n, p) = \left\lfloor \frac{n+p+1}{2} \right\rfloor$. For general $\alpha \geq \frac{1}{2}$, the R implementation (derived from their original S code) uses $h = h(\alpha, n, p) =$ `h.alpha.n(alpha,n,p)` (function in `robustbase`), which is

$$h = h_\alpha = h(\alpha, n, p) := \lfloor 2n_2 - n + 2\alpha(n - n_2) \rfloor, \text{ where } n_2 := \left\lfloor \frac{n+p+1}{2} \right\rfloor. \tag{3}$$

The fraction $\alpha \geq \frac{1}{2}$ can be chosen by the user, where $\alpha = \frac{1}{2}$ is the most robust, and indeed, $h_{1/2} = n_2 = \left\lfloor \frac{n+p+1}{2} \right\rfloor$. Even in general, as long as $n \gg p$, α is approximately the *proportion* of the subsample size h in the full sample (size n):

$$h \approx \alpha \cdot n \iff \alpha \approx \frac{h}{n}, \tag{4}$$

```
> require(robustbase)
> n <- c(5, 10, 20, 30, 50, 100, 200, 500)
> hmat <- function(alpha, p) cbind(n, h.alpha = h.alpha.n (alpha, n,p),
+         h. = floor(alpha * (n + p + 1)), alpha.n = round(alpha * n))
> hmat(alpha = 1/2, p = 3)
```

```
       n h.alpha  h. alpha.n
[1,]   5       4   4       2
[2,]  10       7   7       5
[3,]  20      12  12      10
[4,]  30      17  17      15
[5,]  50      27  27      25
[6,] 100      52  52      50
[7,] 200     102 102     100
[8,] 500     252 252     250
```

```
> hmat(alpha = 3/4, p = 4)
```

```
       n h.alpha  h. alpha.n
[1,]   5       5   7       4
[2,]  10       8  11       8
[3,]  20      16  18      15
[4,]  30      23  26      22
[5,]  50      38  41      38
[6,] 100      76  78      75
[7,] 200     151 153     150
[8,] 500     376 378     375
```

The breakdown point (for $h > \frac{n}{2}$) then is

$$\epsilon_* = \frac{n-h+1}{n}, \tag{5}$$

which is less than but close to $\frac{1}{2}$ for $\alpha = \frac{1}{2}$, and in general, $h/n \approx \alpha$, the breakdown point is approximately,

$$\epsilon_* = \frac{n-h+1}{n} \approx \frac{n-h}{n} = 1 - \frac{h}{n} \approx 1 - \alpha. \tag{6}$$

The raw MCD estimate of location, say $\hat{\mu}_0$, is then the average of these h points, whereas the raw MCD estimate of scatter, $\hat{\Sigma}_0$, is their covariance matrix, multiplied by a consistency factor `.MCDcons(p, h/n)`) and (by default) a finite sample correction factor `.MCDcnp2(p, n, alpha)`, to make it consistent at the normal model and unbiased at small samples.

In practice, for reasonably sized n, p and hence h, it is not feasible to search the full space of all $\binom{n}{h}$ h-subsets of n observations. Rather, the implementation of `covMcd` uses the Fast MCD algorithm of Rousseeuw and van Driessen (1999) to approximate the minimum covariance determinant estimator, see Section 3.

Based on these raw MCD estimates, $\left(\hat{\mu}_0, \hat{\Sigma}_0\right)$, a reweighting step is performed, i.e., `V <- cov.wt(x,w)`, where `w` are weights determined by "outlyingness" with respect to the scaled raw MCD, using the "Mahalanobis"-like, robust distances $d_i\left(\hat{\mu}_0, \hat{\Sigma}_0\right)$, see (7). Again, a consistency factor and a finite sample correction factor are applied. The reweighted covariance is typically considerably more efficient than the raw one, see Pison et al. (2002).

The two rescaling factors for the reweighted estimates are returned in `cnp2`. Details for the computation of the finite sample correction factors can be found in Pison et al. (2002).

3 Fast MCD Algorithm – General notation

Note: In the following, apart from the mathematical notation, we also use variable names, e.g., `kmini`, used in the Fortran and sometimes R function code, in R package `robustbase`.

Instead of directly searching for h-subsets (among $\binom{n}{h} \approx \binom{n}{n/2}$) the basic idea is to start with small subsets of size $p+1$, their center μ and covariance matrix Σ, and a corresponding h-subset of the h observations with smallest (squared) ("Mahalanobis"-like) distances

$$d_i = d_i(\mu, \Sigma) := (x_i - \mu)'\Sigma^{-1}(x_i - \mu), \quad i = 1, 2, \ldots, n, \tag{7}$$

and then use concentration steps ("C steps") to (locally) improve the chosen set by iteratively computing μ, Σ, new distances d_i and a new set of size h with smallest distances $d_i(\mu, \Sigma)$. Each C step is proven to decrease the determinant $\det(\Sigma)$ if μ and Σ did change at all. Consequently, convergence to a local minimum is sure, as the number of h-subsets is finite.

To make the algorithm *fast* for non small sample size n the data set is split into "groups" or "sub-datasets" as soon as

$$n \geq 2n_0, \text{ where } n_0 := \texttt{nmini } (= 300, \text{by default}). \tag{8}$$

i.e., the default cutoff for "non small" is at $n = 600$. The *number* of such subsets in the original algorithm is maximally 5, and we now use

$$k_M = \texttt{kmini } (= 5, \text{by default}), \tag{9}$$

as upper limit. As above, we assume from now on that $n \geq 2n_0$, and let

$$k := \left\lfloor \frac{n}{n_0} \right\rfloor \geq 2 \tag{10}$$

and now distinguish the two cases,

$$\begin{cases} A. & k < k_M \iff n < k_M \cdot n_0 \\ B. & k \geq k_M \iff n \geq k_M \cdot n_0 \end{cases} \tag{11}$$

In case A k (= `ngroup`) subsets aka "groups" or "sub datasets" are used, $k \in \{2, 3, \ldots, k_M - 1\}$, of group sizes n_j, $j = 1, \ldots, k$ (see below). Note that case A may be empty because of $2 \leq k < k_M$, namely if $k_M = 2$. Hence, in case A, we have $k_M \geq 3$.

in case B k_M (= `ngroup`) groups each of size n_0 are built and in the first stage, only a *subset* of $k_M \cdot n_0 \leq n$ observations is used.

In both cases, the disjoint groups ("sub datasets") are chosen at random from the n observations. For the group sizes for case A, n_j, $j = 1, \ldots, k$, we have

$$n_1 = \left\lfloor \frac{n}{k} \right\rfloor = \left\lfloor \frac{n}{\lfloor \frac{n}{n_0} \rfloor} \right\rfloor \quad (\geq n_0) \tag{12}$$

$$n_j = n_1, \qquad j = 2, \ldots, j_* \tag{13}$$

$$n_j = n_1 + 1, \quad j = j_* + 1, \ldots, k, \tag{14}$$

$$\text{where } j_* := k - r \ \in \{1, \ldots, k\}, \tag{15}$$

$$\text{and } r := n - kn_1 = n - k\left\lfloor \frac{n}{k} \right\rfloor \in \{0, 1, \ldots, k-1\}, \tag{16}$$

where the range of j_*, $1, \ldots, k$ in (15) is a consequence of the range of the integer division remainder $r \in \{0, 1, \ldots, k-1\}$ in (16). Consequently, (14) maybe empty, namely iff $r = 0$ ($\iff$ $n = k \cdot n_1$ is a multiple of k): $j_* = k$, and all $n_j \equiv n_1$.

Considering the range of n_j in case A, the minimum $n_1 \geq n_0$ in (12) is easy to verify. What is the maximal value of n_j , i.e., an upper bound for $n_{\max} := n_1 + 1 \geq \max_j n_j$? Consider $n_{1,\max}(k) = \max_{n,\text{given } k} n_1 = \max_{n,\text{given } k} \lfloor \frac{n}{k} \rfloor$. Given k, the maximal n still fulfilling $\lfloor \frac{n}{n_0} \rfloor = k$ is $n = (k+1)n_0 - 1$ where $\lfloor \frac{n}{n_0} \rfloor = k + \lfloor 1 - \frac{1}{n_0} \rfloor = k$. Hence, $n_{1,\max}(k) = \lfloor \frac{(k+1)n_0 - 1}{k} \rfloor = n_0 + \lfloor \frac{n_0 - 1}{k} \rfloor$,

and as $k \geq 2$, the maximum is at $k = 2$, $\max n_1 = \max_k n_{1,\max}(k) = n_0 + \lfloor \frac{n_0-1}{2} \rfloor = \lfloor \frac{3n_0-1}{2} \rfloor$. Taken together, as $n_j = n_1 + 1$ is possible, we have

$$\begin{aligned} n_0 \leq & n_1 \leq \left\lfloor \frac{3n_0 - 1}{2} \right\rfloor \\ n_0 \leq & n_j \leq \left\lfloor \frac{3n_0 + 1}{2} \right\rfloor, \quad j \geq 2. \end{aligned} \tag{17}$$

Note that indeed, $\lfloor \frac{3n_0+1}{2} \rfloor$ is the length of the auxiliary vector `subndex` in the Fortran code.

References

Pison, G., S. Van Aelst, and G. Willems (2002). Small sample corrections for lts and mcd. *Metrika 55*(1-2), 111–123.

Rousseeuw, P. J. and K. van Driessen (1999, August). A fast algorithm for the minimum covariance determinant estimator. *Technometrics 41*(3), 212–223.