EDHEC BUSINESS SCHOOL

ALGORITHMS

CHRISTOPHE CROUX

Make an impact

FROM MY OWN RESEARCH

ELSEVIER

European Journal of Operational Research

Volume 297, Issue 2, 1 March 2022, Pages 782-794

Interfaces with Other Disciplines

Sparse regression for large data sets with outliers

Example: from a data set of 12800 hotels, we estimate:

Hotel Price = -15.7 – 14.3 Distance +22.8 Stars + 176 Rating

1. IDEA

Combine the shooting algorithm used for sparse regression with an additional weighting step to give less weight to outliers.

2. OUTLINE OF ALGORITHM

paper.pdf

3. Algorithmic details

We subsequently discuss the computation of the regression weights w_{ij} (Section 3.1), the robustified version of the response $\tilde{y}_i^{(j)}$ (Section 3.2), the choice of starting value (Section 3.3), the selection of the sparsity parameter λ (Section 3.4), the standardization of the predictors (Section 3.5), and the choice and interplay of robustness constants (Section 3.6). Pseudo-code of the algorithm and the full computer code for the sparse shooting S is available as Supplementary Material.[1]

The algorithm requires calculation of a robust measure of *scale* of the residuals. We use the M-estimator of residual scale (e.g., Maronna et al., 2018, page 34), combining robustness to outliers with a high statistical efficiency. The latter means that in absence of outliers the mean squared estimation error is not much higher than when using the standard deviation (assuming normality of the errors).

3.1. Regression weights

The cell-specific weights in Eq. (3) ensure that one obtains cellwise robust regression estimates. These regression weights are a function of the residuals $r_i^{(j)} = \tilde{y}_i^{(j)} - \hat{\beta}_j x_{ij}$, where $\hat{\beta}_j$ results from the previous iteration of the shooting loop. A popular choice of weight function is Tukey's biweight, visualized in Fig. 2. The weight function is applied to the residuals standardized by s_j, a robust measure of scale of these residuals. One has

$$w_{ij} = \begin{cases} (1 - (\frac{r_i^{(j)}}{c \cdot s_j})^2)^2 & \text{if } \frac{|r_i^{(j)}|}{s_j} \leq c \\ 0 & \text{otherwise.} \end{cases} \quad (4)$$

Regression outliers give rise to large standardized residual values and are, in turn, downweighted. We take $c = 3.420$, yielding a good trade-off between statistical efficiency and robustness to outliers of the corresponding regression estimator. More specifically, this choice of c yields an efficiency of 85% (for normally distributed error terms) and a breakdown point[2] of 20%, see Rousseeuw and Yohai (1984). Note that the M-estimator of scale uses the same weight function and tuning constant.

3.2. Robustified response

The new response $\tilde{y}_i^{(j)}$ in the simple regression Eq. (2) filters out the effect of the predictor variables with index different from j. To prevent outliers in the cells x_{ik} to propagate, we need to use a robustified response

$$\tilde{y}_i^{(j)} = y_i - \sum_{k \neq j} \tilde{x}_{ik} \hat{\beta}_k,$$

where a "cleaned" predictor value $\tilde{x}_{ik}$ is used instead of the observed cell value x_{ik}. The "cleaned" cells are

$$\tilde{x}_{ik} = \begin{cases} x_{ik} & \text{if } \frac{|r_i^{(k)}|}{s_k} \leq 3 \\ \hat{x}_{ik} & \text{otherwise,} \end{cases} \quad (5)$$

where $\hat{x}_{ik}$ is an expected cell value. Hence, if the standardized residual is not flagged as outlying, the cleaned cell equals the observed cell value. Otherwise, the cleaned cell value equals the expected cell value. The expected cell values are obtained using a simplified version of the robust data imputation method of Rousseeuw and Van Den Bossche (2018), outlined below. This cutoff value 3 in (5) ensures that, given a regression model with normally distributed errors, less than 0.3% of the observations are expected to be mislabelled as outliers.

Robust data imputation We use a simplified version of the robust data imputation method of Rousseeuw and Van Den Bossche (2018) which contains the core of their procedure and works well for our purpose. The idea is to consider outlying values in the data matrix as missing values, that are to be replaced by so called expected cell values. Consider predictor variable k, with values in column k of the data matrix. First, we obtain its most robustly correlated variable, say variable j. Then we regress predictor variable k

3. IMPLEMENTATION

sparseshootS.R

```
sparseshootingS <- function(x, y, k = 3.420, maxIteration = 100, tol = 10^-2,
                            betaEst = NULL, intercept = NULL, scaleVar = NULL, xhat = NULL, xtilde = NULL,
                            maxituniv = 1, maxitscale = 100, wvalue = 3, shoot_order = "default",
                            nlambda = 100, post = TRUE, lambda_grid = NULL, kpred = NULL, predset = NULL){
  #### Function to compute sparse shooting S ####
```

Example:

sparseshootingS(x=features,y=price,data=hotels)

The idea of the algorithm may be implemented in an almost infinite number of ways.

Implementations differ in

- Speed of execution
- Stability
- User friendliness / documentation
- …

TOY EXAMPLE

- We need an algorithm for:

divide a number by 2 if it is even and double the number if it is odd.

Idea:

Divide the number by two.
If the remainder of this division is zero, then the number is even.

Implementation in Python:

```
def myfunction(x):
    test=x/2-round(x/2)
    if (test == 0):
     result= x/2
    else:
     result= 2*x
    return(result)
```

Implementation in Python:

```
def myfunction(x):
    test=x/2-round(x/2)
    if (test == 0):
     result= x/2
    else:
     result= 2*x
    return(result)
```

Other Implementation:

```
def myfunction2(x):
    if (x%2 == 0):
     result= x/2
    else:
     result= 2*x
    return(result)
```

Implementation in R:

```
myfunction <-function(x)
{
  test=x/2-round(x/2)
  if (test==0) result=x/2
  else result=2*x
  result
}
```