Journal of Data Science 771-780 , DOI: 10.6339/JDS.201810_16(4).00006

FORWARD REGRESSION IN R: FROM THE EXTREME SLOW TO THE EXTREME FAST

Michail Tsagris[*] and Manos Papadakis[*]

Department of Computer Science, University of Crete, Heraklion, Creece,

ABSTRACT

Forward regression has been criticised heavily and one of the many reasons is regarding its speed and its stopping criteria. The main focus of this paper is on demonstrating how to make it efficient, using R. Our method works for continuous predictor variables only, as the use of the partial correlation plays the most important role.

KEYWORDS: forward regression, partial correlation coefficient, computational efficiency

[*] CONTACT: Michail Tsagris. Email: mtsagris@yahoo.gr
[*] CONTACT: Manos Papadakis. Email: papadakm95@gmail.com

1. Introduction

It is common truth that efficient code is not a matter of computer, operating system, or programming language, but a matter of implementation. Despite this ordinal common reasoning argument, not many programmers, scientific developers and mainly researchers put too much emphasis on this aspect. Researchers looking for speed in their calculation tend to use Matlab, Python, C++, etc. In general, the aforementioned languages or software are faster than R. What most people are not aware of though is that R has many ”powerful” functions whose efficiency is comparable to Matlab and Python (see Ozgur et al. (2017) for a comparison).

Computational efficiency is a main drawback of forward regression as well. Neverthe- less it is taught in the undergraduate departments of statistics across the globe. This paper deals exactly with this issue, right at the heart of the problem, how to make for- ward regression efficient. To this end, we will make use of R’s built-in command *cor* ; a very efficiently written function whose abilities we will take into advantage here.

The paper focuses on the case where both the response and the predictor variables are continuous (i.e. not categorical nor survival, etc.). In the next section we present the forward selection algorithm using the correlation coefficient. Section 3 shows, in R, the key algorithmic point responsible for reducing execution time. Examples demonstrate the great computational savings and Section 4 concludes the paper.

2. Forward selection method for linear models

Despite the title of this Section, the general idea of the forward selection method is applicable to all kinds of models, linear or not, regardless of normally distributed error or not. Suppose we have a univariate response variable $y \in \mathbb{R}^n$ and a matrix of p predictor variables $\mathrm{X} \in \mathbb{R}^{n \times p}$. The algorithm below summarizes the method.

Forward selection method

1. Standardise all predictor variables, (make them have zero mean and unit variance) and center the values of the response variable.
2. Examine all univariate associations between y and x$_i$, for $i = 1, \cdots, p$. The j-th most significant among the significant ones enters the model.
3. Examine all conditional associations (y, x$_i$) $|CS$. The most significant among the significant ones enters the model. CS stands for conditioning set.
4. Repeat Step 3, increasing the CS every time by 1 (one variable is added) until no significant association exists.

In the next 2 subsections we will see how to perform Steps 2-4. That is, how to assess the significance of the (conditional) associations.

2.1 Signicance of a univariate association

In order to assess the univariate associations, Pearson's correlation coefficient is calculated. The significance of the sample correlation r will be assessed via Fisher's z-transform (Fisher, 1915)

$$z = \frac{1}{2} log \frac{1+r}{1-r} = tanh^{-1}(r)$$

Under H_0

$$z \sim N\left(\frac{1}{2} log \frac{1+\rho}{1-\rho}, \frac{1}{n-3}\right),$$

where ρ is the true correlation coefficient. Alternatively, a t_{n-3} distribution can be used. Note that this is different from assessing the significance of the slope coefficient of a simple linear regression in which case we would use the following test statistic

$$T = \frac{r\sqrt{n-2}}{\sqrt{1-r^2}}$$

Under H_0, $T \sim t_{n-2}$. Note that in the simple linear regression

$$y_i = \hat{\alpha} + \hat{\beta} z_i + e_i \qquad (1)$$

T^2 coincides with the Wald test statistic for the $H_0 : \beta = 0$ which is given by

$$T^2 = W = \frac{\hat{\beta}^2}{var(\hat{\beta})} \tag{2}$$

and the usual F-test

$$T^2 = F = \frac{SSE_0 - SSE_1}{SSE_1/(n-2)},$$

where SSE_1 is the sum of squares of the errors of the regression model (1) and SSE_0 is simply$(n-1)\sigma_y^2$, where σ_y^2 is the variance of y. The proof of straightforward and hence omitted. The z or t test based upon Fisher's z-transformation is asymptotically equivalent to the Wald (and hence the T^2 and the F tests) (Anderson, 2003).

The convenience of the z-transformation lies in the fact that its standard error has a known formula and requires no further calculation and this is taken into advantage when the computational cost is examined. The use of the correlation and partial correlation instead of many linear regression models has also been advised by Weisberg (2005) for the purpose of computational savings.

2.2 Significance of a conditional association

In the second and latter steps of the forward regression, the significance of the conditional correlations is to be examined. The two models are

$$\begin{aligned} H_0 &: y_i = \hat{\alpha}_1 + z_i\hat{\beta}_1 \qquad + e_i \\ H_1 &: y_i = \hat{\alpha}_2 + z_i\hat{\beta}_2 + \gamma x_i + e_i \end{aligned} \tag{3}$$

Hence, under H_0 $\gamma = 0$. The partial F-test, or in general the F-test for comparing two models, one nested within the other, is

$$F = \frac{SSE_0 - SSE_1}{SSE_1/(n-q-1-1)}$$

The conditional correlation must be used with the cardinality of the conditional set being, $|z| = q$, i.e there are q variables. When there is only one conditioning variable, $|z| = 1$ the conditional correlation has a closed form which we also take into account in order to decrease the computational cost

$$r(y,x|z) = \frac{r(y,x) - r(x,z)r(y,z)}{\sqrt{1-r^2(x,z)}\sqrt{1-r^2(y,z)}},$$

where *r(y; x)* denotes the correlation between *y & x*, and *r(y; z), r(x; z)* are the correlation coefficients between *y & z* and between *x & z* respectively. The general method of calculating the partial correlation, for $|z| = q \geq 1$ is given by (Fisher et al., 1924)

$$r(y,x|z) = r(e_{1i}, e_{2i}), \tag{4}$$

where e_{1i} and e_{2i} are the residuals of the two regression models

$$e_{1i} = y_i - \hat{\beta}_0 + z_i\hat{\beta}_1 \tag{5}$$

$$e_{2i} = x_i - \hat{\delta}_0 + z_i\hat{\delta}_1 \tag{6}$$

The variance of *r(y; x|z)* is equal to *1 = (n - 3 - q)*.

The γ coefficient of the second model (3) is related to the partial correlation (4) via the following relationship

$$\mathrm{r(y,x|z)} = \frac{W}{\sqrt{W^2 + n - q - 1}}$$

where W is the Wald test statistic (2) for the γ coefficient.

Note that, both for the partial correlation and the partial F test, two linear regression models are fitted. Hence, one can argue that the computational cost is the same. We will see however later on that this is not true when using R. Alternatively, the relationship equation can be used with one fitted model.

3. Forward selection with linear models in R

The R code for the forward selection method using the partial correlation can be found in the package Rfast (Papadakis et al., 2018). The first two steps of the method are rather straightforward. We remind the reader that all predictor variables have been centred and scaled to have unit variance and the response variable is only centred. In the first step, the command cor(y, x) is used, where y is a vector and x is a matrix. In the second step, the partial correlation, with one conditioning variable, is applied. The next (if necessary) steps are the most important ones regarding the speed gains. For this reason we take into advantage a very powerful base command in R, the .lm.fit, which is 10 times faster than lm.fit.

In the next two lines of R code, z is the selected variables at step k (conditioning set), x is the whole set of variables, including z and y is the response variable. The vector of partial correlations (4) between the response and each of the predictor variables conditioning on the selected variables is then calculated in the third line.

```
z <- x[, sela]
e1 <- .lm.fit(z, y)$residuals
e2 <- .lm.fit(z, x)$residuals
yx.z <- cor(e2, e1)
```

By taking a close look in the above R code segment, you will see that the calculation of the partial correlation coefficient via the residuals (5) is what makes the forward regression fast. According the algorithm of forward search, one must "scan" all non selected predictor variables at each step. So, with 1000 predictor variables, if 10 variables were to be selected, one would have to create roughly 10; 000 regression models. Using a *for* loop in R, this has already become slow.

We will now bridge the mathematics with the R code. The estimates of the linear regression coeffcients are given by

$$\hat{B} = (X^T X)^{-1} X^T Y. \tag{7}$$

Let us translate the two residual objects of the above R code into mathematics using (7)

$$e_1 = Y - (Z^T Z)^{-1} Z^T Y \ \text{ (vector of residuals)}$$
$$E_2 = X - (Z^T Z)^{-1} Z^T X \ \text{ (matrix of residuals).}$$

Hence,instead of having to "scan" the design matrix of predictor variables at every step, all we need to do is two multiplications, one by the left and one by the right. The using the cor command, we get the vector of partial correlation coefficients at almost no time.

When a new candidate variable is tried, it is possible that the crossproduct of the design matrix may not be invertible, because the candidate variable is highly correlated with a previously selected variable, and thus a check should be made at every step. However, R's implementation of the forward regression will work just fine, because the QR decomposition implementation behind handles these cases.

3.1 Examples comparing execution time

We compared our implementation against the more general implementation of forward regression available in MXM (Lagani et al., 2017). MXM is a variable selection oriented package, offering many different methods for many types of data. The comparison might not look fair, mainly because the MXM's command lm.fsreg for linear models is generic, it is designed to accept any type of predictor variables, continuous and/or categorical and uses command lm for this reason. In addition, the algorithm has not been optimized, using .lm.fit and incrementally updating the design matrix. That implementation is based on the algorithm of the forward regression. At each step, all the remaining variables are "scanned" and their significance is examined.

We did two single examples, with only 500 predictor variables. In the first case no variable is associated with the response variable. In the second case, the response is linearly related with three predictor variables. The relevant code to produce the data and the output using the package microbenchmark are given below.

```
x <- matrix(runif(500 * 500, 1, 100), ncol = 500)
## first case scenario
y <- rnorm(500)
mb <- microbenchmark(cor.fsreg(y, x), lm.fsreg(y, x) )
## second case scenario
y <- 3 * x[, 10] + 2 * x[, 100] + 3 * x[, 20] + rnorm(500, 0, 5)
mb2 <- microbenchmark(cor.fsreg(y, x), lm.fsreg(y, x) )
```

In the first example, Rfast required 22 miliseconds (0.02 seconds), whereas MXM required 2,204 miliseconds (2.2 seconds). In the second example Rfast required 64 miliseconds (0.064 seconds), whereas MXM required 6,590 miliseconds (6.59 seconds).

We can see that our implementation is 100 times faster than MXM's generic implementation when no predictor variable is truly significant. It is 274 times faster when there are only three predictor variables truly significant. In the first case, two variables were selected, whereas in the second case, the three significant variables were selected.

In order to get better insights into the time savings we expanded our simulation studies with larger sample sizes and higher number of predictor variables using the same case scenarios. The sample sizes we tried were n = (500; 1; 000; 5; 000; 10; 000; 20; 000; 50; 000) and the number of variables spanned from 100 to 1000, with an increasing step of 100. Figure 2 presents the speed-up factors of cor.fsreg relative lm.fsreg which indicate how slower the latter is relative to the first one. With small sample sizes for example (left column of plots) lm.fsreg can be more than 250 times slower than cor.fsreg. This number decreases with larger sample sizes (right column of plots), but is above 10 nonetheless. To give an example though, if lm.freg required on average 3; 400 seconds, nearly an hour, cor.fsreg required 280 seconds, less than 5 minutes.

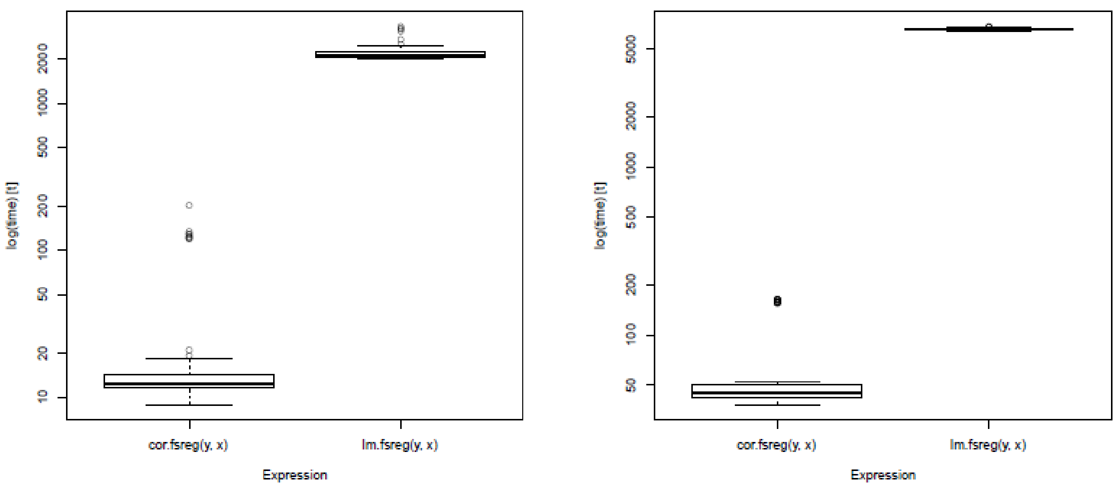
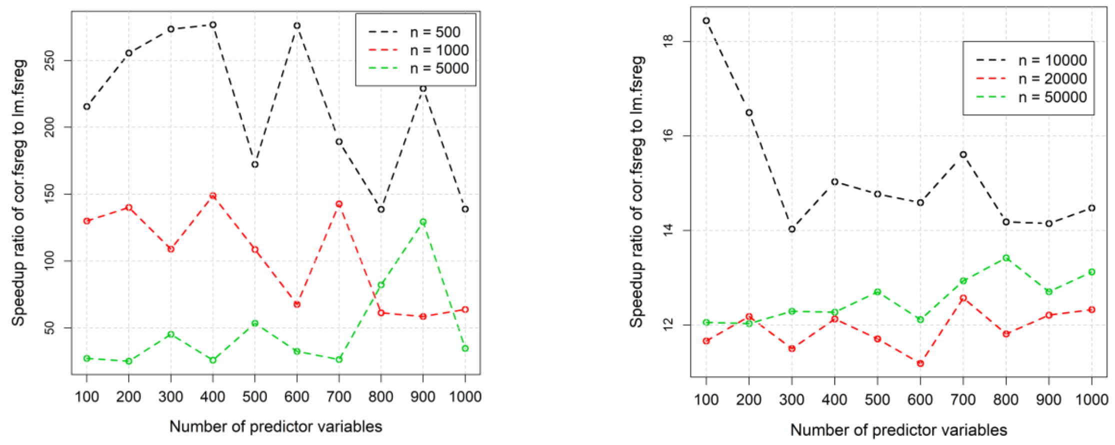
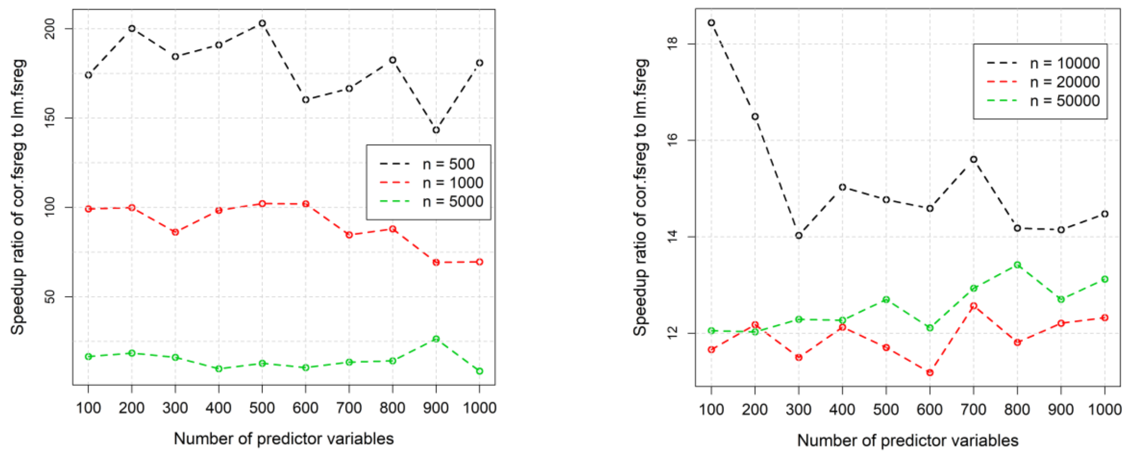


Figure 1: Box plots of execution time for both functions. The left graph refers to the first example, whereas the right graph to the second graph.



First case scenario



Second case scenario

Figure 2: Speed-up factors of cor.fsreg to lm.fsreg. All numbers are greater than 1,indicating the number of times lm.fsreg' is slower than cor.fsreg.

4. Conclusions

We have showed how to speed up dramatically the forward regression in the linear models case with continuous predictor variables only. A comparison of Rfast's optimised forward regression implementation, for the continuous response and continuous predictor variables case, with the generic (not fully optimised) forward regression implementation in MXM showed dramatic speed-up factors. The relative differences decrease with larger sample sizes, yet, they are significant.

We demonstrated the fact that it is not a matter of computer, the computational power, or even of the algorithm many times, but a matter of the implementation. This sentence is very common, yet not everybody writes functions and packages without taking this into account. The second message here is that in order to have really fast functions, one must take cases and optimise each case separately, rather than have a generic function. It becomes clear though, that the optimisation of the generic function is the optimal solution.

References

[1] Anderson, T. W. (2003). *An introduction to multivariate statistical analysis (3rd Ed.).*Wiley: New York.

[2] Fisher, R. A. (1915). Frequency distribution of the values of the correlation coefficient in samples from an indefinitely large population. *Biometrika*, 10(4):507-521.

[3] Fisher, R. A. et al. (1924). The distribution of the partial correlation coefficient. *Metron*,3(3-4):329-332.

[4] Lagani, V., Athineou, G., Farcomeni, A., Tsagris, M., and Tsamardinos, I. (2017). Feature Selection with the R Package MXM: Discovering Statistically-Equivalent Feature Subsets. *Journal of Statistical Software*, 80(7).

[5] Ozgur, C., Hall, U., Colliau, T., Rogers, G., and Hughes, Z. (2017). Matlab vs. python vs. r. *Journal of Data Science*, 15(3):355-372.

[6] Papadakis, M., Tsagris, M., Dimitriadis, M., Fafalios, S., Tsamardinos, I., Fasiolo, M.,Borboudakis, G., Burkardt, J., Zou, C., and Lakiotaki, K. (2018). *Rfast: Fast R Functions*. R package version 1.9.0.

[7] Weisberg, S. (2005). *Applied linear regression (3rd Ed.)*. John Wiley & Sons: New Jersey.