STAT 341: Tutorial 4 – Practice with Implicit Attributes

Friday January 31, 2020

Part I: Not all Estimating Equations Arise from Derivatives

With this example I want to make clear that althought many of the estimating equations we encounter arise as derivatives from some objective function, the *don't have to*.

Consider the implicitly defined attribute $\boldsymbol{\theta} = (\mu, \sigma)$ where μ and σ are respectively measures of center and spread in the population $\mathcal{P} = \{y_1, y_2, \ldots, y_N\}$. Find $\widehat{\boldsymbol{\theta}} = \left(\widehat{\alpha}, \widehat{\beta}\right)$, the solution to the following system of equations

$$\boldsymbol{\psi}(\boldsymbol{\theta}; \mathcal{P}) = \mathbf{0}$$

given by

$$\begin{bmatrix} \left(\frac{1}{N}\sum_{u\in\mathcal{P}} y_u\right) - \mu \\ \\ \left(\frac{1}{N}\sum_{u\in\mathcal{P}} y_u^2\right) - \mu^2 - \sigma^2 \end{bmatrix} = \begin{bmatrix} 0 \\ \\ 0 \end{bmatrix}$$

Note that this system of equations arose by equating sample moments with population moments. This method of estimation is referred to as the Method of Moments.

Part II: Least Absolute Deviations Regression

In class we have talked a lot about estimating $\boldsymbol{\theta} = (\alpha, \beta)$ the intercept and slope associated with the simple linear regression

$$y_u = \alpha + \beta(x_u - \overline{x}) + r_u$$

And we have done this in a variety of different ways by altering the objective function. One such possible objective function is the *absolute error loss* function which gives rise to **least absoluate deviations (LAD) regression**:

$$\widehat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta} \in \boldsymbol{\Theta}}{\operatorname{argmin}} \sum_{u \in \mathcal{P}} |y_u - \alpha - \beta(x_u - \overline{x})|$$

In this part we will use gradient descent to fit this model to the `Animals` data. But first we need to define `rho` and `gradient` functions in `R` which requires that we first compute the gradients by hand.

(a) Let

$$\rho(\boldsymbol{\theta}; \mathcal{P}) = \sum_{u \in \mathcal{P}} |y_u - \alpha - \beta(x_u - \overline{x})|$$

Calculate the gradient vector $\boldsymbol{g} = \nabla \rho(\boldsymbol{\theta}; \mathcal{P})$.

(b) Write *factory functions* `createLADRho(x,y)` and `createLADGradient(x,y)` which take in as input only the data and which return as output the least absolute deviations objective function and the corresponding gradient function, respectively

```
createLADRho <- function(x, y) {
    ## local variable
    xbar <- mean(x)
    ## Return this function
    function(theta) {
        alpha <- theta[1]
        beta <- theta[2]
        sum(abs(y - alpha - beta * (x - xbar)))
    }
}

createLADGradient <- function(x, y) {
    ## local variables
    xbar <- mean(x)
    function(theta) {
        alpha <- theta[1]
        beta <- theta[2]
        ru = y - alpha - beta * (x - xbar)
        -1 * c(sum(sign(ru)), sum(sign(ru) * (x - xbar)))
    }
}
```

(c) Using the `gradientDescent` function (from class) together with the `gridLineSearch` and `testConvergence` functions (from class) as well as `rho` and `gradient` functions created by your factory functions from part (b), find the LAD estimates $\widehat{\boldsymbol{\theta}} = \left(\widehat{\alpha}, \widehat{\beta}\right)$ for the `Animals` data.

```
library(robustbase)
rho <- createLADRho(x = log(Animals2$body), y = log(Animals2$brain))

g <- createLADGradient(x = log(Animals2$body), y = log(Animals2$brain))

res.manual <- gradientDescent(theta = c(0, 0), rhoFn = rho, gradientFn = g,
    lineSearchFn = gridLineSearch, testConvergenceFn = testConvergence,
    maxIterations = 5000, tolerance = 1e-20, relative = TRUE)

print(res.manual)
```

```
## $theta
## [1] 3.3589387 0.7368431
##
## $converged
## [1] TRUE
##
## $iteration
## [1] 26
##
## $fnValue
## [1] 48.01975
```

(d) In class we performed LAD regression using the `lad` function from the `L1pack` package. Let's confirm that what we found above agrees with the output of this other function.

```
library(L1pack)
res.L1pack <- lad(log(Animals2$brain) ~ I(log(Animals2$body) - mean(log(Animals2$body))))
print(res.L1pack)
```

```
## Call:
## lad(formula = log(Animals2$brain) ~ I(log(Animals2$body) - mean(log(Animals2$body))))
## Converged in 9 iterations
##
## Coefficients:
##                                        (Intercept)
##                                             3.3541
## I(log(Animals2$body) - mean(log(Animals2$body)))
##                                             0.7373
##
## Degrees of freedom: 65 total; 63 residual
## Scale estimate: 1.04454
```

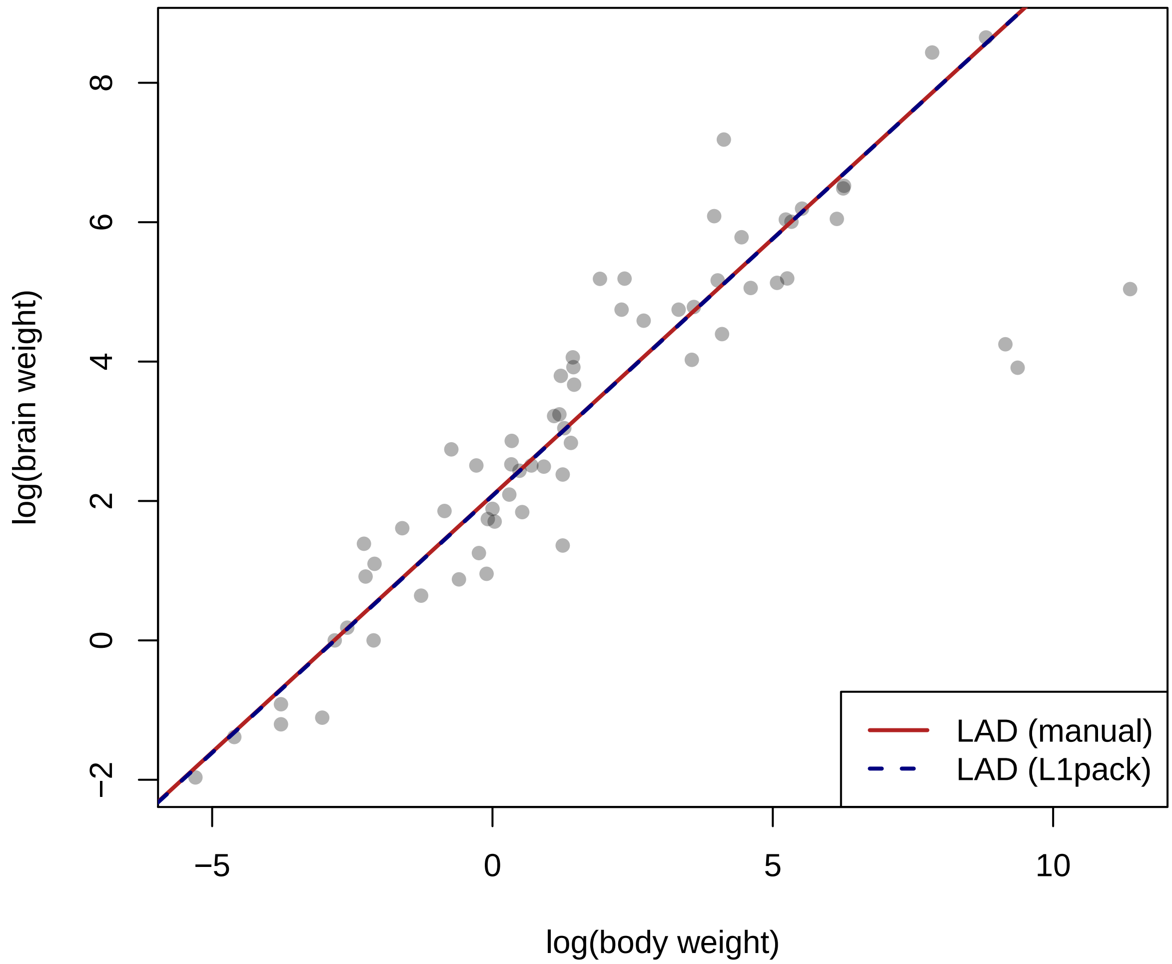
(e) Construct a scatter plot of `log(brain weight)` versus `log(body weight)` for the `Animals` data and plot both lines of best fit – the one we determined manually and the one calculated using `lad` – to see if the difference is material. Use a legend to distinguish among the lines.

```r
plot(x = log(Animals2$body), y = log(Animals2$brain), main = "", xlab = "log(body weight)",
    ylab = "log(brain weight)", pch = 16, col = adjustcolor("black", alpha.f = 0.3))

abline(a = res.manual$theta[1] - res.manual$theta[2] * mean(log(Animals2$body)),
    b = res.manual$theta[2], col = "firebrick", lwd = 2, lty = 1)

abline(a = res.L1pack$coef[1] - res.L1pack$coef[2] * mean(log(Animals2$body)),
    b = res.L1pack$coef[2], col = "navyblue", lwd = 2, lty = 2)

legend("bottomright", legend = c("LAD (manual)", "LAD (L1pack)"), col = c("firebrick",
    "navyblue"), lty = 1:2, lwd = 2)
```



(f) Can we apply the Newton-Raphson Method to this problem? In other words, can we determine the LAD estimate $\widehat{\boldsymbol{\theta}} = \left(\widehat{\alpha}, \widehat{\beta}\right)$ via the Newton-Raphson Method?