EM for Linear Regression LAD

AUTHOR
Zehaan Naik

Problem Statement

In the classic linear regression setup, i.e.:

$$y = X\beta + \epsilon \mid \epsilon \sim N(0,1),$$

we wish to find the LAD estimator for β. For the proof of concept, we consider a simplified version of this problem. Particularly, we seek to minimize:

$$\sum_{i=1}^{n} \mid y_i - a - bx_i \mid$$

Proposed Solution

We know that the following problems have well defined solutions:

$$\sum_{i=1}^{n} \mid y_i - a \mid, \quad \sum_{i=1}^{n} \mid y_i - ax_i \mid$$

Then, one attempt to solve the problem can be looked at in terms breaking the current problem statement into 2 parts:

1. Fix some initial value for a say a^0 and absorb this constant into y_is and solve the equation:

$$\sum_{i=1}^{n} \mid z_i - bx_i \mid \quad \text{where } z_i = y_i - a^0$$

2. Use the computed value for b say b^1 and absorb this constant ($b^1 x_i$) into y_is and solve the equation:

$$\sum_{i=1}^{n} \mid w_i - a \mid \quad \text{where } z_i = y_i - b^1 x_i$$

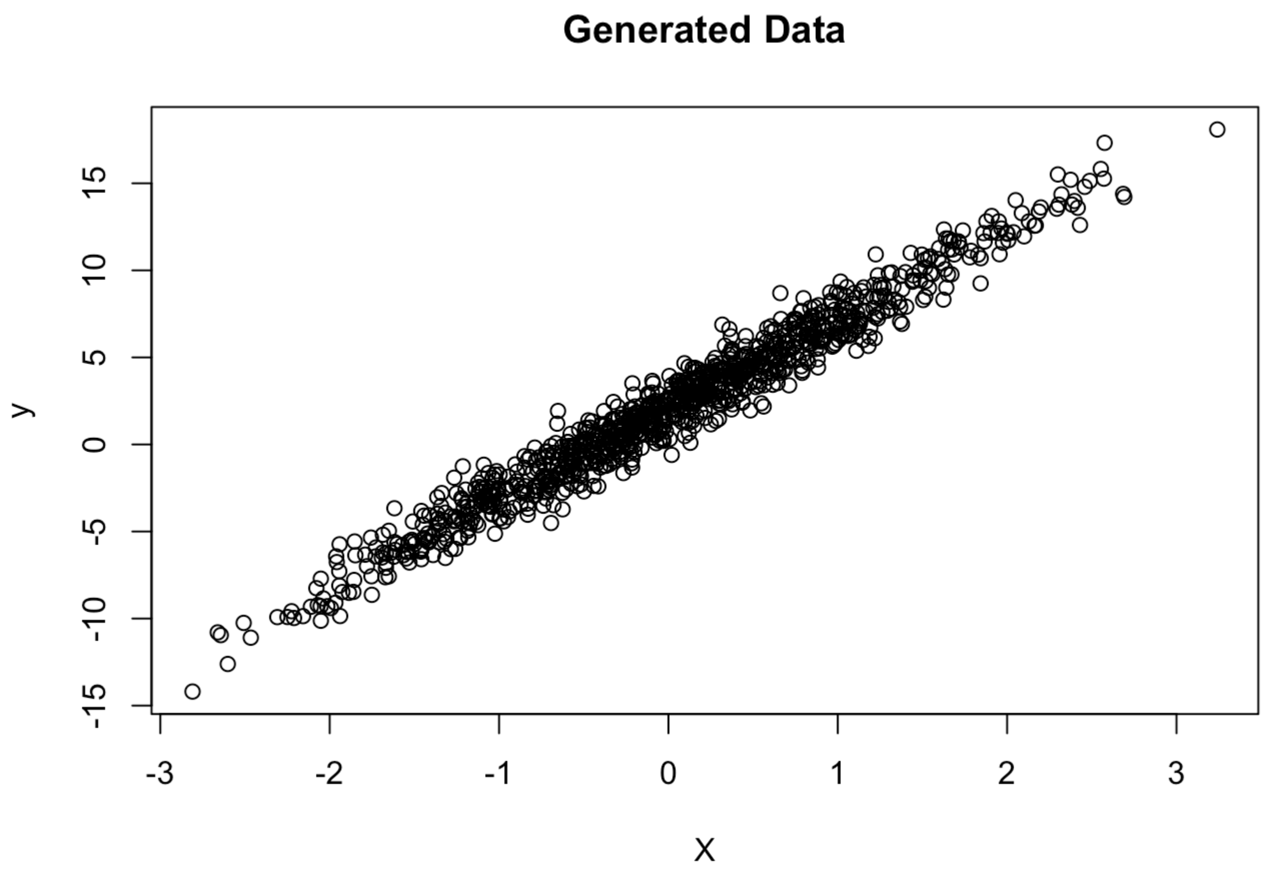
Experiment 1. 2-Variable data set

Generating a random data set

Here, I generate a random 1000 observation data-set with known values for both the target parameters a and b. I also visualize the data on a simple X, y plot.

```
set.seed(123)

### 1. Generate synthetic linear regression dataset
```

```r
n <- 1e3    # samples
p <- 2      # features

X <- matrix(rnorm(n * (p-1)), n, (p-1))
X <- cbind(1,X)
beta_true <- c(2,5)  # true coefficients
y <- X %*% beta_true + rnorm(n, sd = 1)

## Visualising the data
plot(X[,2], y,
     main = "Generated Data",
     xlab = "X",
     ylab = "y")
```



Sample run of my proposed algorithm to check the parameter fit.

```r
### 2. Optimization loop

## Loop Parameters
max_iter <- 100
a <- numeric(max_iter)
b <- numeric(max_iter)
a_curr <- 10
b_curr <- 10

x <- X[,2]
```

```r
## Comparision values to store
MAE.loss <- numeric(max_iter)
MAE.params <- numeric(max_iter)

## Optim Loop
for(i in 1:max_iter){
  ## Solving for b_i
  z <- y - a_curr
  z.bar <- z/x
  b_curr <- median(z.bar)
  b[i] <- b_curr

  ## Solving for
  w <- y - b_curr * x
  a_curr <- median(w)
  a[i] <- a_curr

  ## Storing the current MAE on parameters and predictions
  MAE.loss[i] <- mean(abs(y - a_curr - b_curr*x))
  MAE.params[i] <- mean(abs(c(a_curr, b_curr) - beta_true))
}

cat("True parameters are a = ", beta_true[1] , "and b = ", beta_true[2])
```

```
True parameters are a =  2 and b =  5
```

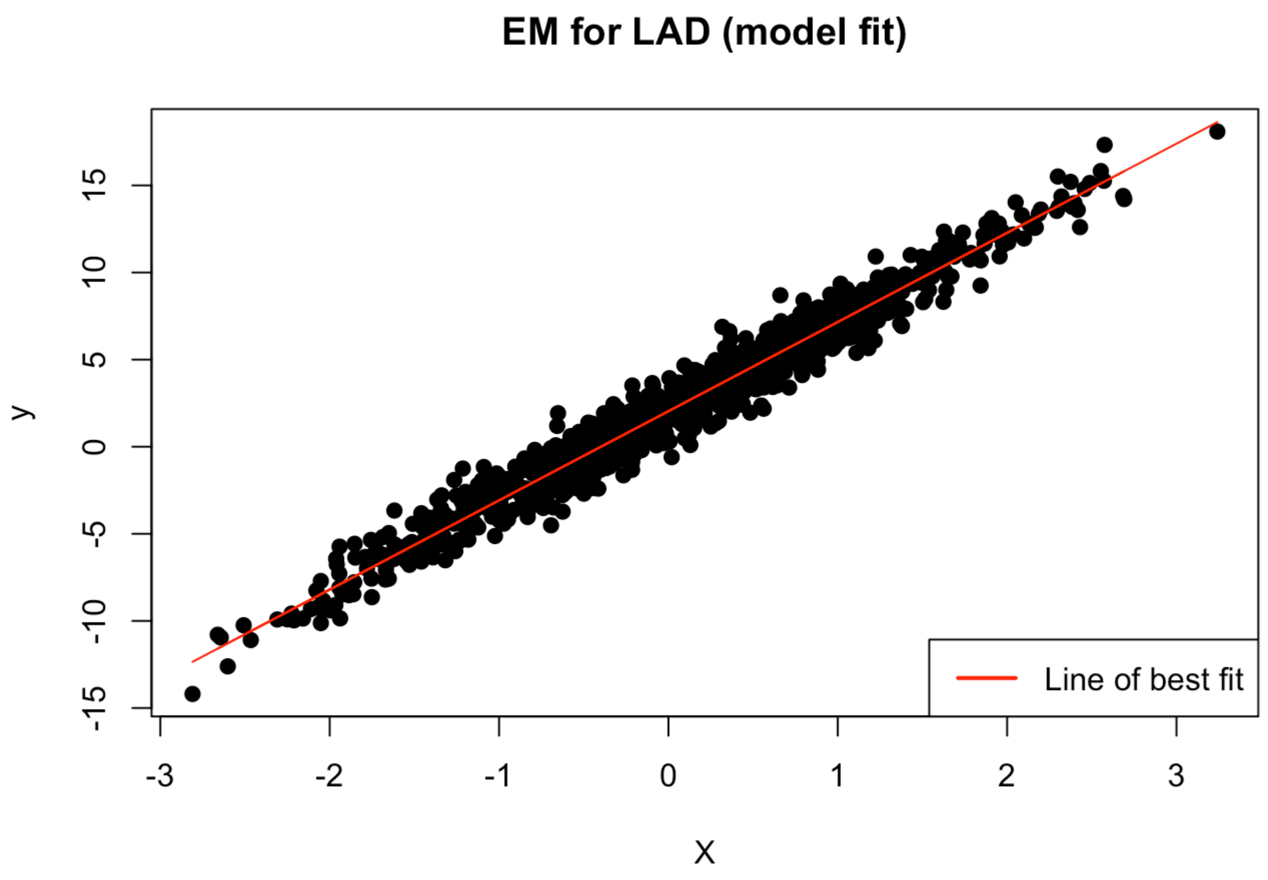
```r
cat("Post optimization we get a = ", a_curr , "and b = ", b_curr)
```

```
Post optimization we get a =  2.039266 and b =  5.120642
```

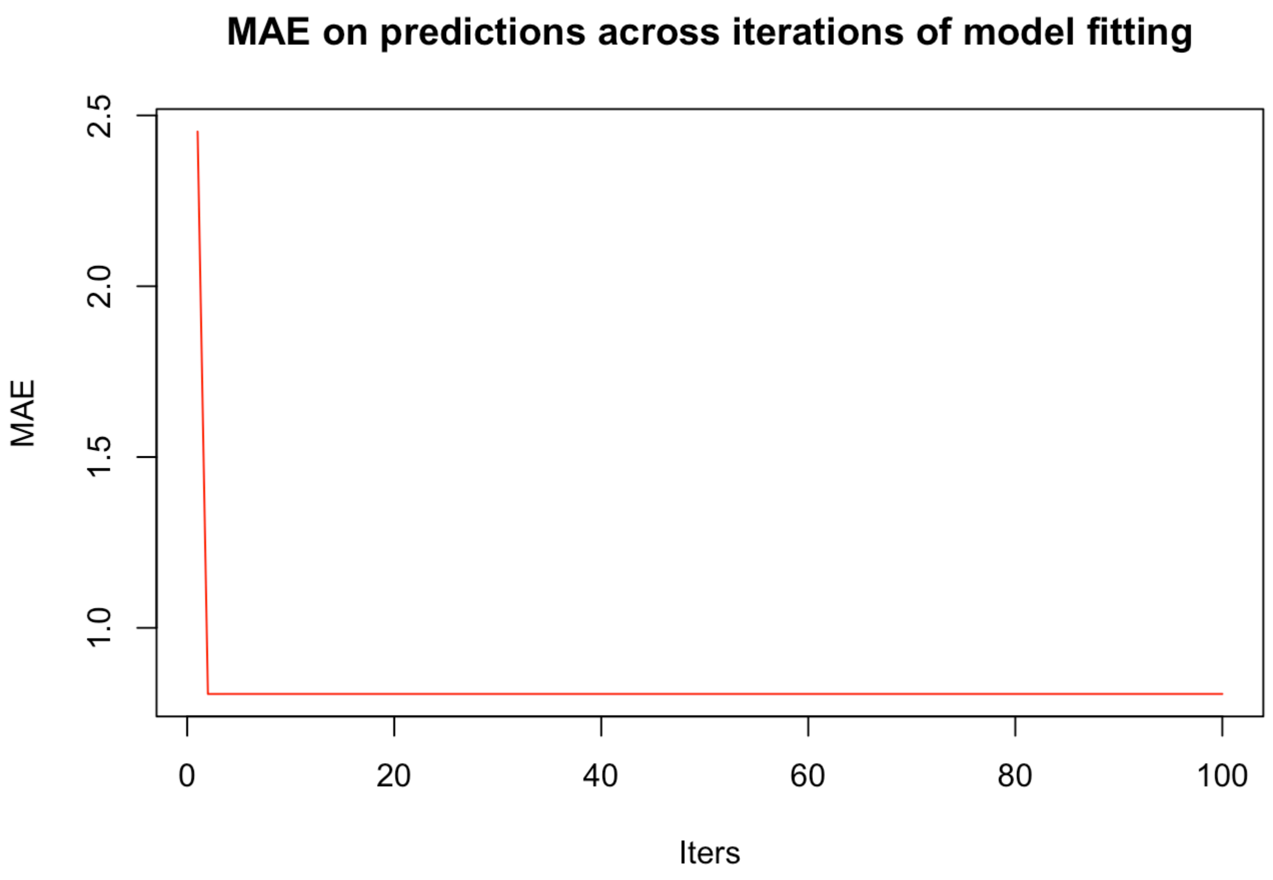
Visualizing the results

Here, I plot the MAE on parameters, prediction losses and the actual model fit to check how well the methodology performs on our generated data set. We can observe that we have a quick convergence of both the loss values and the MAE on parameters indicating a stable fit by the algorithm.

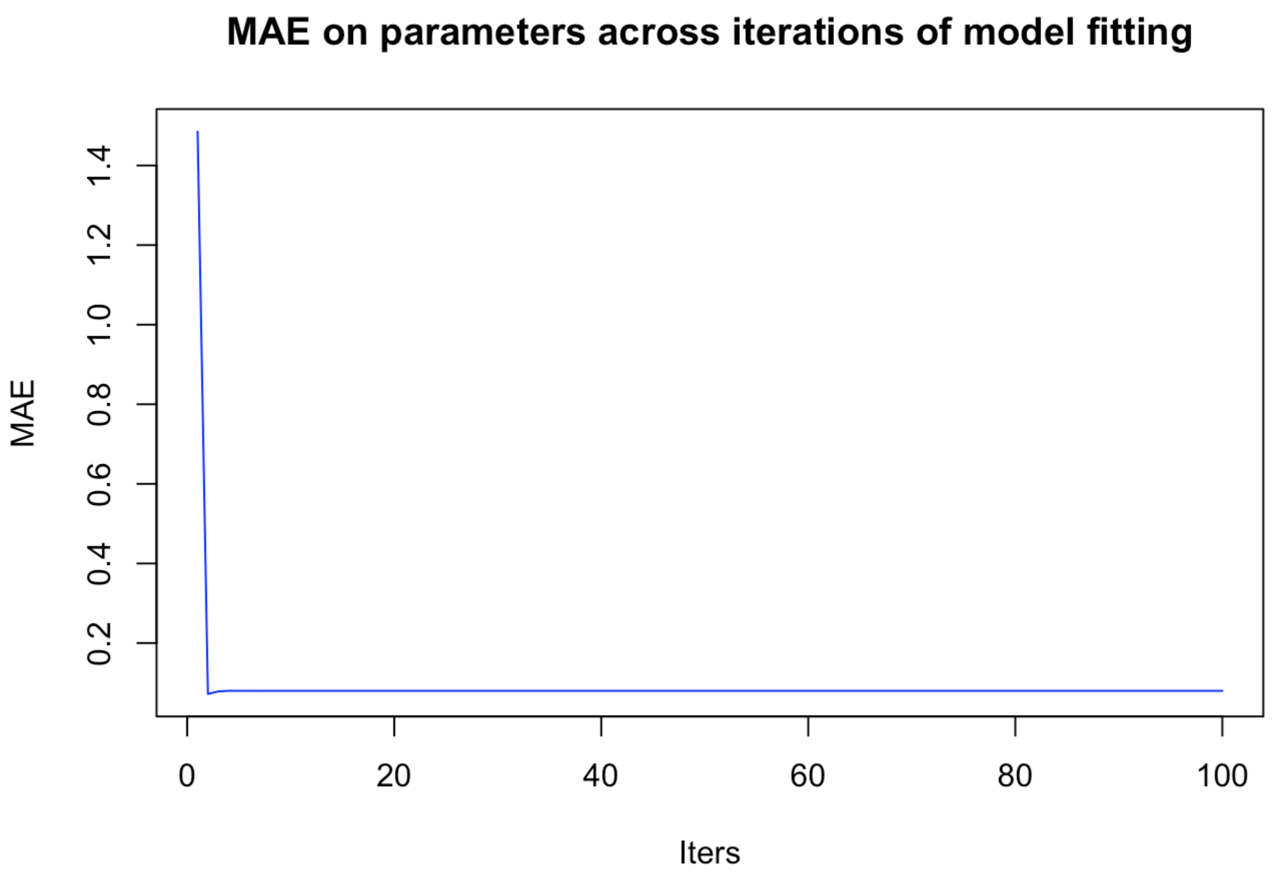
```r
### 3. Plots to visualise method performance

## Model Fit
plot(x,y,
     main = "EM for LAD (model fit)",
     xlab = "X",
     ylab = "y",
     pch = 19)
lines(x, a_curr + b_curr*x, col = "red")
legend("bottomright", c("Line of best fit"), col = ("red"), lwd = 2)
```



```
## Checking the MAE over predictions
plot(1:max_iter, MAE.loss[1:max_iter], col = "red", type = "l",
     main = "MAE on predictions across iterations of model fitting",
     xlab = "Iters",
     ylab = "MAE")
```



```r
## Checking the MAE loss over params
plot(1:max_iter, MAE.params[1:max_iter], col = "blue", type = "l",
     main = "MAE on parameters across iterations of model fitting",
     xlab = "Iters",
     ylab = "MAE")
```



Experiment 1. p-Covariate data set

Now, just to complicate the problem a little bit, I generate a data with p many dimensions in X and try to estimate all the p parameters using this new methodology. The overall approach remains the same, we fix all other parameters while trying to optimize for an isolated parameter.

```
set.seed(123)

### 1. Generate synthetic linear regression dataset
n <- 1e3    # samples
p <- 10      # features

X <- matrix(rnorm(n * (p-1)), n, (p-1))
X <- cbind(1,X)
beta_true <- c(2,1:(p-1))  # true coefficients
y <- X %*% beta_true + rnorm(n, sd = 1)

### 2. Optimization loop

## Loop Parameters
max_iter <- 100
a <- numeric(max_iter)
b <- matrix(0, nrow = max_iter, ncol = p-1)
a_curr <- 1
b_curr <- rep(1, p-1)
```

```
## Comparision values to store
MAE.loss <- numeric(max_iter)
MAE.params <- numeric(max_iter)

## Optim Loop
for(i in 1:max_iter){
  for(j in 1:(p-1)){
    x.j <- X[,1+j]
    X.rest <- X[,-c(1,1+j)]
    ## Solving for b_i,j
    z <- y - a_curr - X.rest %*% b_curr[-j]
    z.bar <- z/x.j
    b_curr[j] <- median(z.bar)
    b[i,] <- b_curr
  }

  ## Solving for a_i
  w <- y - X[,-1] %*% b_curr
  a_curr <- median(w)
  a[i] <- a_curr

  ## Storing the current MAE on parameters and predictions
  MAE.loss[i] <- mean(abs(y - a_curr - X[,-1] %*% b_curr))
  MAE.params[i] <- mean(abs(c(a_curr, b_curr) - beta_true))
}
### CHECKING THE VALUES OF ALL CALCULATED PARAMETERS
print("True Parameters:")
```

```
[1] "True Parameters:"
```

```
print(beta_true)
```

```
 [1] 2 1 2 3 4 5 6 7 8 9
```

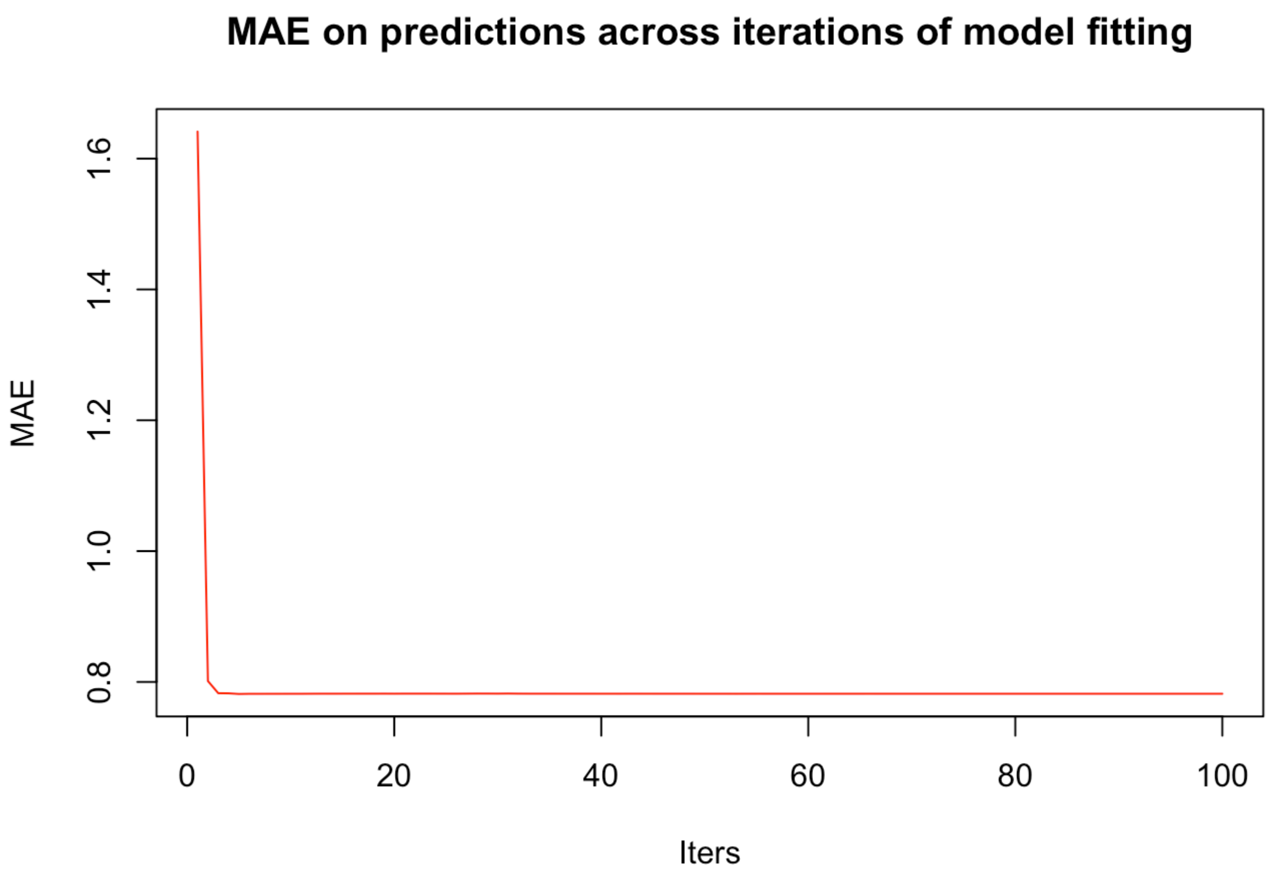
```
print("Estimated Parameters:")
```

```
[1] "Estimated Parameters:"
```

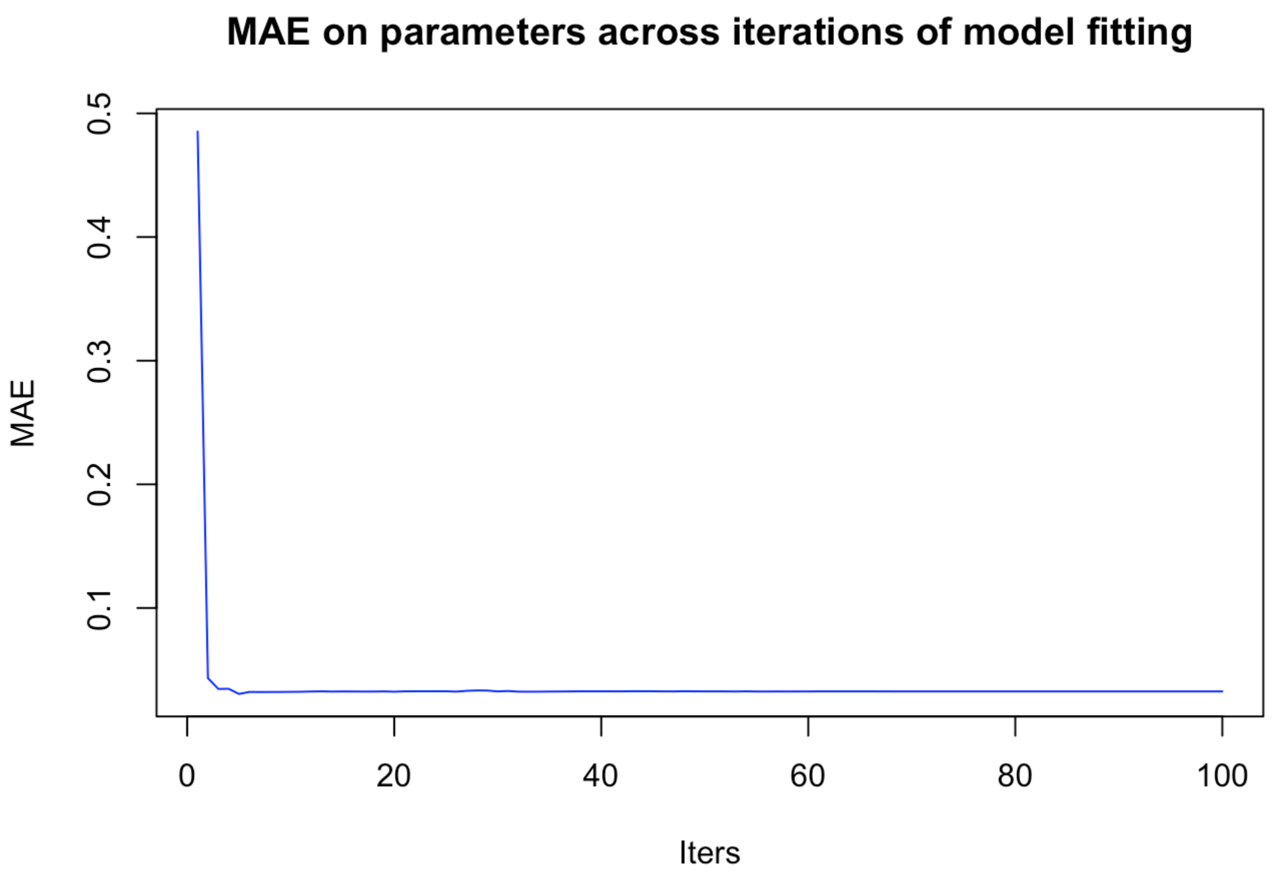
```
print(c(a_curr, b_curr))
```

```
 [1] 1.936893 1.004593 2.043029 2.954753 4.078318 4.970971 5.981237 7.002671
 [9] 7.967983 9.008431
```

```
### 3. Plotting the results
## Checking the MAE over predictions
plot(1:max_iter, MAE.loss[1:max_iter], col = "red", type = "l",
     main = "MAE on predictions across iterations of model fitting",
     xlab = "Iters",
     ylab = "MAE")
```



```
## Checking the MAE loss over params
plot(1:max_iter, MAE.params[1:max_iter], col = "blue", type = "l",
     main = "MAE on parameters across iterations of model fitting",
     xlab = "Iters",
     ylab = "MAE")
```



Clearly, we are able to compute all the parameters with reasonable accuracy and the convergence of MAEs for both parameters and prediction losses indicate a stable fit.

Conclusion

This is a promising method for estimating the LAD estimates for parameters. Given that we are able to work out the underlying theory to prove the descent condition for the algorithm set-up, this method can provide a reliable framework to optimize regression problems in the absolute deviation framework which often lacks an analytical solution.