Automated Model Building and Goodness-of-fit via Quantile Regression

Haim Bar

September 9, 2022

Abstract

This repository contains code and data used in the paper *Automated Model Building and Goodness-of-fit via Quantile Regression* by Bar, Booth, and Wells (in preparation). Given P predictors x_i and n observations for each x_i and a response variable y, the goal is to build a model, $y = f(x_1, \ldots, x_P)$ where $f()$ consists of combinations of powers of the x_i's, which fits the data well across multiple quantiles.

1 Prerequisites

In order to run the code you must first install the **QREM** package. Since **QREM** has a model selection option for cases in which the number of predictors is large you also need to install the packages **edgefinder** and **SEMMS**. The recommended way to install these packages is from the GitHub repository:

```
devtools::install_github("haimbar/edgefinder")
devtools::install_github("haimbar/SEMMS")
devtools::install_github("haimbar/QREM")
```

The model building algorithm is implemented in a function called *fitQRloop* in the file runQREM.R. The function takes five arguments:

- M the data matrix with P columns and n rows.
- qns The quantile which will be used in the fitting algorithm.
- minDiff The minimal improvement in the overall goodness of fit in order to accept a new term.
- maxdeg The maximum degree of any term in the model.
- maxrows The maximum number of possible terms up to degree maxdeg.

The file initSim.R contains the values we used by default. It also contains three other variables which are used by **QREM** in the fitting process:

- mxm The maximum number of segments in the partition of the selected variable.
- alphaQ The level of the goodness of fit test.
- plotit A Boolean variable which tells the function *flatQQplot* whether to show intermediate diagnostic plots for each accepted new term in the model.

```
qns <- 1:5/6
k <- length(qns)
minDiff <- 4
maxdeg <- 15
maxrows <- 5000
mxm <- 30
```

```
alphaQ <- 0.01
plotit <- FALSE
```

2 A Univariate Example

The file Code/Univariate02.R contains the code for example #1 in the paper, where

$$f(x) = x^5 e^{-x}$$

and the random noise is normally distributed with mean 0 and standard deviation which grows linearly with x, $sd = 0.25(x + 0.05)$.

```
source("Code/initSim.R") # set up some global parameters
qns <- round(qns, digits=3)
N <- 5000
set.seed(211111)
x <- runif(N, min=0, max=4*pi)
y <- exp(-x)*x^5 + rnorm(N, 0, 0.25*(x+0.05)) # EXAMPLE 1 in the paper
M <- data.frame(y=y, x1=x)
res <- fitQRloop(M=M, qn = qns, maxdeg = maxdeg, minDiff = minDiff) # ❶
pdf("Figures/Uni02.pdf", width=5, height=5)
plot(x, y, cex=0.5, pch=19, col="grey66", axes=F)
axis(1); axis(2); grid()
for (i in 1:k) {
  lines(sort(x), res$qremFit[[i]]$fitted.mod$fitted.values[order(x)],
        col=2)
}
dev.off()
```

The fitting algorithm is invoked in the line denoted by ❶ in the code above. The fitted quantile regression models are shown as red curves in Figure 1.

3 A Multivariate Example

The program in Code/multivariate04.R generates a data matrix with 2,000 observations and 4 predictors, but only x_1, x_2, x_4 are related to the response, according to:

$$y = x_1 x_2 x_4 + \epsilon$$

and the random errors are i.i.d. $\epsilon \sim N(0, 0.1^2)$

```
source("Code/initSim.R")
set.seed(211013)
N <- 2000
M <- matrix(runif(N*4, min = -1, max=3), nrow=N, ncol=4)
y <- M[,1]*M[,2]*M[,4] + rnorm(N, 0, 0.1)
M <- as.data.frame(cbind(y, M))
colnames(M) <- c("y", "x1", "x2", "x3", "x4")
res <- fitQRloop(M=M, qn = qns, maxdeg = maxdeg, minDiff = minDiff,
                  maxrows = maxrows)
pdf("Figures/multivariate04.pdf", width=5, height=5)
i <- which(qns == 0.5)
plot(M$y, res$qremFit[[i]]$fitted.mod$fitted.values, cex=0.6, pch=19,
     xlab="Observed", ylab="predicted", axes=FALSE, col="grey66")
abline(0, 1, col=3, lwd=2); axis(1); axis(2)
```

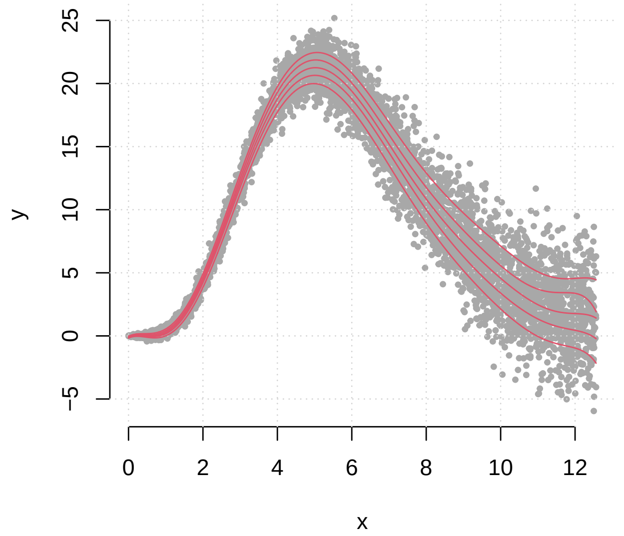

Figure 1: The fitted QR model for $q = 1/6, \ldots, 5/6$ where the true model is $f(x) = x^5 e^{-x}$ with random errors i.i.d. $N(0, [0.25(x + 0.05)]^2)$.

The fitted model found by our algorithm is:

```
x1*x2*x4 + I(x4^2) + x2*x4 + I(x2^2) + x1*x4 + x1*x2 + I(x1^2) + x4 + x2 + x1
```

The predicted values are very close to the observed one, as can be seen in Figure 2.

Examples which involve more complicated models with more than one predictor may take a few minutes to run, and sometimes even longer.

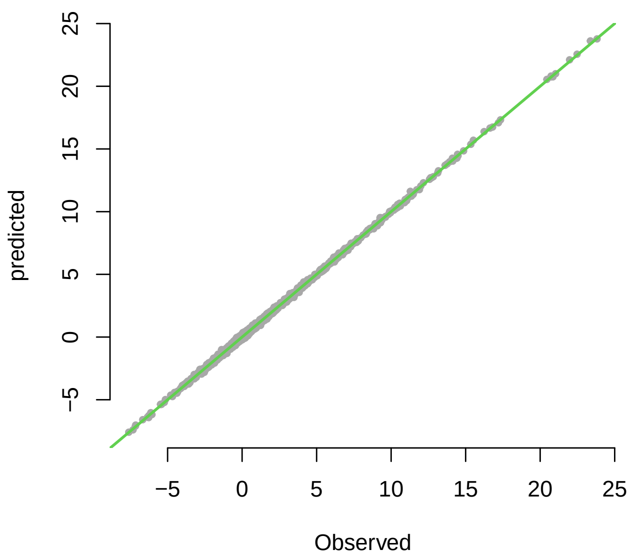

Figure 2: Fitted QR model vs. observed values for $q = 1/2$ where the true model is $f(x_1, x_2, x_3, x_4) = x_1 x_2 x_4$ with random errors i.i.d. $N(0, 0.1^2)$.

4 Case Studies

The repository contains several case studies:

- Concrete.R - predicting the strength of concrete. (Note that fitting a model to this dataset is time-consuming. Saved results can be found in concreteResultsR2s.RData).
- mpgreg.R - which factors contribute to gasoline consumption. (We run it 10 times, and it takes a minute or two to finish). The file MPG.py contains a python program which uses TensorFlow to fit the MPG data. The code was obtained from the TensorFlow documentation.
- bankNotes.R (a classification example, see `https://archive.ics.uci.edu/ml/datasets/`banknote+authentication).
- Lidar.R - Lidar readings data (univariate, demonstrating data augmentation).
- uscrime.R - FBI rape rate data by state (demonstraing regression discontinuity).
- Ozone.R - Ozone data.

In this section we use the MPG data
The fitted model is:

```
                 Estimate Std. Error t value  Pr(>|t|)
(Intercept)        20.677     0.1335 154.911 2.42e-257
I(x6^2)             0.509     0.0296  17.207  9.09e-45
x5                 -0.446     0.0538  -8.285  6.38e-15
x6                  1.937     0.0885  21.892  7.55e-61
x3                 -2.690     0.1504 -17.883  3.94e-47
x2                 -0.423     0.1403  -3.018  2.80e-03
x1(4.5,6.5]        -0.714     0.1217  -5.865  1.36e-08
x1(6.5,8.5]        -1.227     0.2670  -4.594  6.78e-06
x4                 -6.166     0.1504 -41.005 2.92e-115
x5:x6               0.699     0.0548  12.746  3.28e-29
x6:x3              -1.083     0.1228  -8.817  1.76e-16
x6:x2              -0.092     0.1124  -0.819  4.14e-01
x6:x1(4.5,6.5]      0.548     0.1125   4.870  1.95e-06
x6:x1(6.5,8.5]      1.732     0.2360   7.339  2.78e-12
x1(4.5,6.5]:x4      3.933     0.1792  21.947  4.92e-61
x1(6.5,8.5]:x4      5.483     0.2101  26.101  1.78e-74
```

The predicted values are plotted versus the observed one in Figure 3.

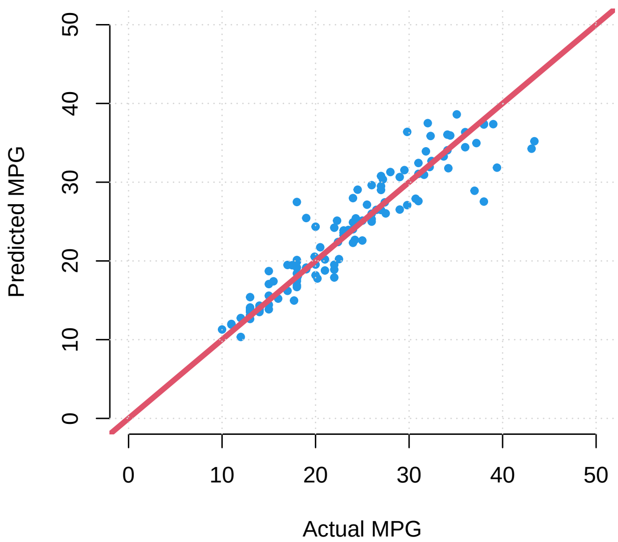

Figure 3: Fitted QR model vs. observed values for $q = 1/2$ for the MPG data.