REPORTS ON THE PROBLEM GIVEN BY VOLEON CAPITAL

YAJING LIU

ABSTRACT. Using the L^1 and Huber loss functions, we design robust linear regression models for the given 5 data sets with heavy tailed noise, and compare them with the L^2 least square model. We also assess model by computing the R^2 for the data after cleaning the noise. Also, we assess the robustness by compare the changes of the regression results before and after cleaning noise.

1. Robust Model

In a linear regression model, the robustness depends on the chosen loss function. When the distribution of the noise is heavy tailed, the square loss function is less robust to the noise. This is because the population solutions are $f(x) = E(Y|x)$ for squared-error loss. Thus, when the noise is not symmetric or very large, the ordinary least square puts more weights on the large noise.

When it comes to large noise, the L_1 loss function is usually a better choice in the interest of the robustness. See Page 349 Section 10.6 in [ESLII]. There are also other choices, such as the Huber Loss function. In this report, we test L^1, L^2, and Huber models.

1.1. Generating data.

In this question, we assume the random variables y, x, e has a linear relation:

$$y = ax + b + e,$$

where a, b are constants which are not dependent on y, x, e in each data set.

The noise e does not necessarily have the normal distribution $N(0, \sigma^2)$. Instead, to meet the requirement of the question, we assume e has a heavy-tailed distribution and $E(e) = 0$.

To generate the data, here is a procedure.

- Construct a random variable e with a heavy-tailed distribution and $E(e) = 0$. For example, we can pick a uniform distribution $U(-c, c)$ with $c \gg 0$. Other choices are combinations of uniform density and Gaussian density, such as $\frac{1}{2}(U(-a, a) + N(0, 100))$. The key point is to choose a large variance.
- Fix a, b.
- Construct a random variable x with uniform distribution $U(d, f)$, where d, f are arbitrary constants.
- Generate 100 random values of x and e : $\{x_i\}_{i<=100}$ and $\{e_i\}_{i<=100}$.
- $y_i = a \cdot x_i + b + (x - E(x))e_i$ (In the given data, it looks like the larger x is, the larger the noise is.)

However, the second question in the instruction is ambiguous.

'2. Formulate a model for the data-generating process.'

There are two ways to understand the question, and thus two ways to answer the question.

(1) Design a data set irrelevant to the given 5 data sets.
(2) Find the distribution of e in the given data sets, and then generate data using this distribution of e.

1.2. Choosing the loss functions.

We have three models for the following three loss functions.

1.2.1. *L^2 loss function: Least square.*

$$L(y, f(x)) = (y - f(x))^2.$$

In our question, given the $n + 1$ data (x_i, y_i) with $0 \leq i \leq n$, good estimators of a, b should minimize the following function:

$$f(a, b) = \sum_{i=0}^{n} L(y_i, ax_i + b) = \sum_{i=0}^{n} (y_i - ax_i - b)^2$$

1.2.2. *L^1 loss function.*

$$L(y, f(x)) = |y - f(x)|.$$

In our question, given (x_i, y_i) with $0 \leq i \leq n$, good estimators of (a, b) should minimize the following function:

$$f(a, b) = \sum_{i=0}^{n} L(y_i, ax_i + b) = \sum_{i=0}^{n} |y_i - ax_i - b|$$

1.2.3. *Huber loss function.*

$$L(y, f(x)) = \begin{cases} (y - f(x))^2 & \text{if } |y - f(x)| < \delta \\ 2\delta|y - f(x)| - \delta^2 & \text{if } |y - f(x)| \geq \delta \end{cases}$$

Given n data (x_i, y_i) with $0 \leq i \leq n$. We want to minimize the following function:

$$f(a, b) = \sum_{i=0}^{n} L(y_i, ax_i + b).$$

After plotting the data, we can see that the data is mainly supported in the strip with vertical width $\Delta y = 20$ and the noise are outside this strip. Thus, we manually choose $\delta = 10$.

1.3. **Optimization for loss function.** We use the functions in python package **scipy.optimize** to optimize the loss functions.

- Since the L^1 loss function is not smooth, we use the function **scipy.optimize.fmin** based on the simplex algorithm to optimize the L^1 loss function.
- Since the L^2 loss function and the Huber loss function are both smooth, we use the function **scipy.optimize.fmin_cg** based on the conjugate gradient algorithm to compute the minimum.

In order to get a good initial guess, we first plot the data and then give an estimate of line which penetrate these scattered points. The initial guess of (a, b) is $(2, 2)$.

1.4. **Assessment of the parameters.** Usually, we compute the R^2 to assess the model parameters, where

$$R^2 = \frac{\sum(\hat{y}_i - \bar{y})^2}{\sum(y_i - \bar{y})^2}.$$

Since here the noise is significant, to fit the data better is usually not feasible. Here, the R^2 would be rather small, since most the variations can not be explained by linear regression.

The data set 3, 4, and 5 have large noise. To determine whether the parameters are good chosen, we need to pick out the outliers manually from the data sets 3, 4,and 5. Then, we compute the R^2 after cleaning the noise. It would reflect the goodness of the model fitting.

To assess the robustness of the model, we compute the parameters (a, b) and (a', b') before and after cleaning the noise. Then, we compute the the distance between the parameters (a, b) and (a', b') for all three models. By assessing the change of the regression results, we can properly measure the robustness. The smaller the change is, the more robust the model is.

2. Data.

2.1. **Basic features.** We are given 5 data sets, with sample sizes being $100, 100, 50, 50, 50$. We can also see that the data set 3, 4, and 5 have significant noises. These 5 data sets have different data ranges. See Table 1.

2.2. **Fitting models.** By running ordinary least square regression, absolute loss regression, M-regression, we plot the fitted curves in the following graphs. In each graph, there are the given data set (x_i, y_i) and the three fitted lines models. See Figure 3, 4, 5, 6, 7.

2.3. **Python codes.** In the folder, we have two python codes. **solution.py** is the scripting codes for all the computations, **NonGaussion_linear_Model.py** is a class called non_Gaussian_model, which makes the scripts into a class. One can use the methods to access data.

Data set	size	range of x	range of y
1	100	[-4.86609666841, 4.9190609483]	[-27.0080348744, 18.3871737021]
2	100	[-25.9232766995 , 19.7133738622]	[-63.3599148151 , 77.6422592936]
3	50	[-24.4231132108 , 19.7157201448]	[-38.6778530484 , 78.2811359263]
4	50	[0.0919178628828 , 4.82285024365]	[-15.8285095565 , 13.9528639143]
5	50	[-16.7400244131,5.86122083912]	[-17.2852105949, 54.5991249576]

TABLE 1. **Table of the basic features of given data sets**

2.4. **Assessment for the parameters.** Since in these 5 data sets all the noises are significant, it is hard to fit the data well. Thus, the R^2 for the three models are all very small. However, the R^2 for different models are very close. Even sometimes L^1 and $Huber$ regression fit better, for example in the data set 1, we have

$$R^2_{L^1} = 23\%$$
$$R^2_{L^2} = 16\%$$
$$R^2_M = 20\%.$$

However, after cleaning the data, the

$$R^2_{L^1} = 39\%.$$

Therefore, we conclude that the absolute regression and M-regression preserve the same goodness of fitting as the Ordinary Least square regression, while they are more robust to noises.

Assessing the robustness, for data set 4, we pick out 7 outliers. Then, we do the L^1 and OLS regression again. We found that L^1 regression is almost the same with original regression with large noise. However, the OLS regression model changes a lot when erasing the outliers. See Figure 1, 2.

For the L^1 model, before cleaning noise, the regression coefficients are

$$(a, b) = (1.27973484095, 0.468683677855),$$

while after cleaning noise,

$$(a', b') = (1.39944999577, 0.245040665905).$$

Thus, the squared difference for L^1 loss model is

$$(a - a')^2 + (b - b')^2 = 0.064347915087128088$$

For the least square model, before cleaning noise, the regression coefficients are

$$(a, b) = (0.633375489474, 1.14813147631),$$

while after cleaning noise, the regression coefficients are

$$(a, b) = (1.25836752507, 0.16360760448).$$

Thus, the squared difference for OLS model is

$$(a - a')^2 + (b - b')^2 = 1.9920979980749423.$$

Thus, we can say the (a, b) for L^1 model is chosen well. We also plot the residuals for the three models for the data set 4 in Figure 9.

REFERENCES

[ESLII] Trevor Hastie, Robert Tibshirani, Jerome Friedman, *Elements of Statistical Learning, Data Mining, Inference, and Prediction*, Springer.

DEPARTMENT OF MATHEMATICS, UCLA, 520 PORTOLA PLAZA, LOS ANGELES, CA 90095, USA
E-mail address: yajingleo@math.ucla.edu

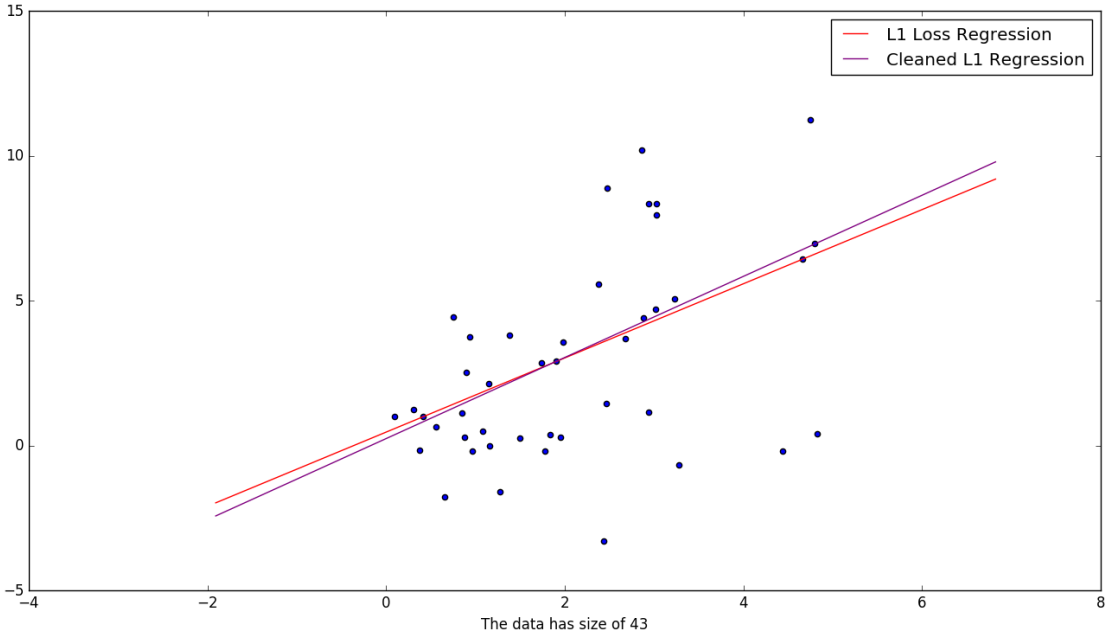

FIGURE 1. **The robustness of L^1 model for data set 4.** We found that the two lines using the data before and after cleaning the noise are almost the same. This verifies that it is robust to noise.

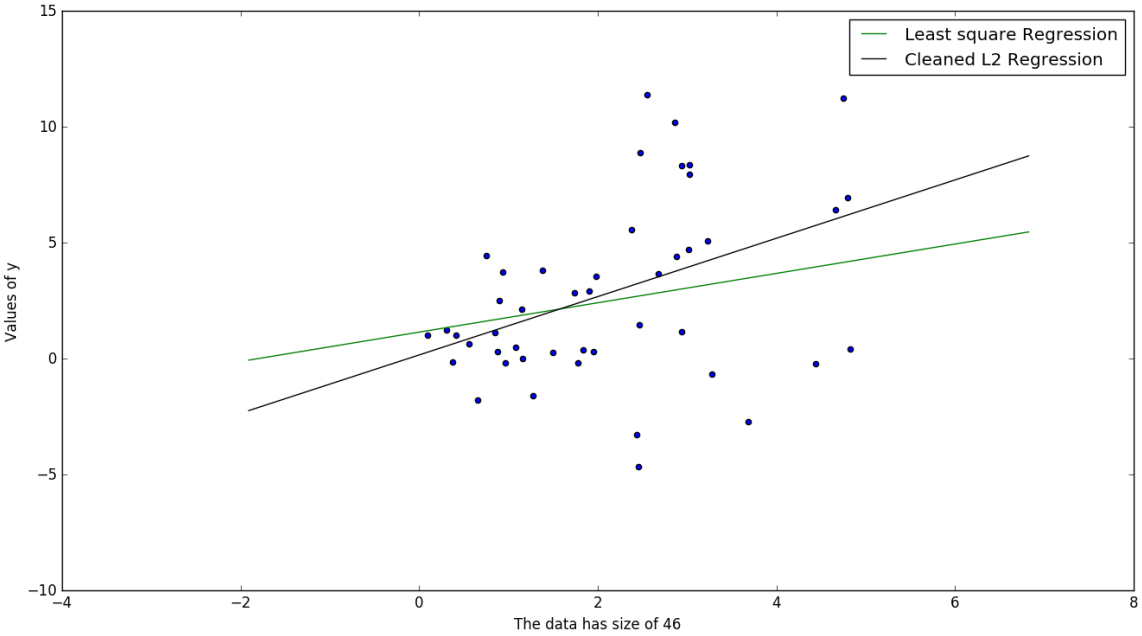

FIGURE 2. **Compare least square regressions for data set 4before and after cleaning noise.** The regression lines has a significant difference, which means the the OLS model is more sensitive to noise.

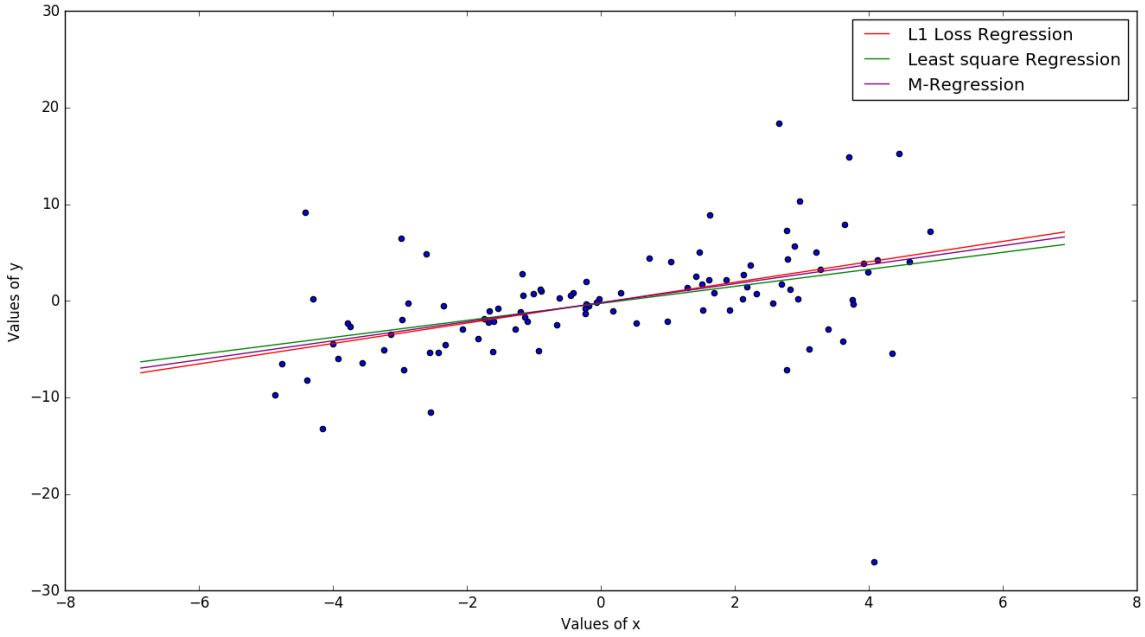

FIGURE 3. **Linear regression of the data set 1 using three models.**

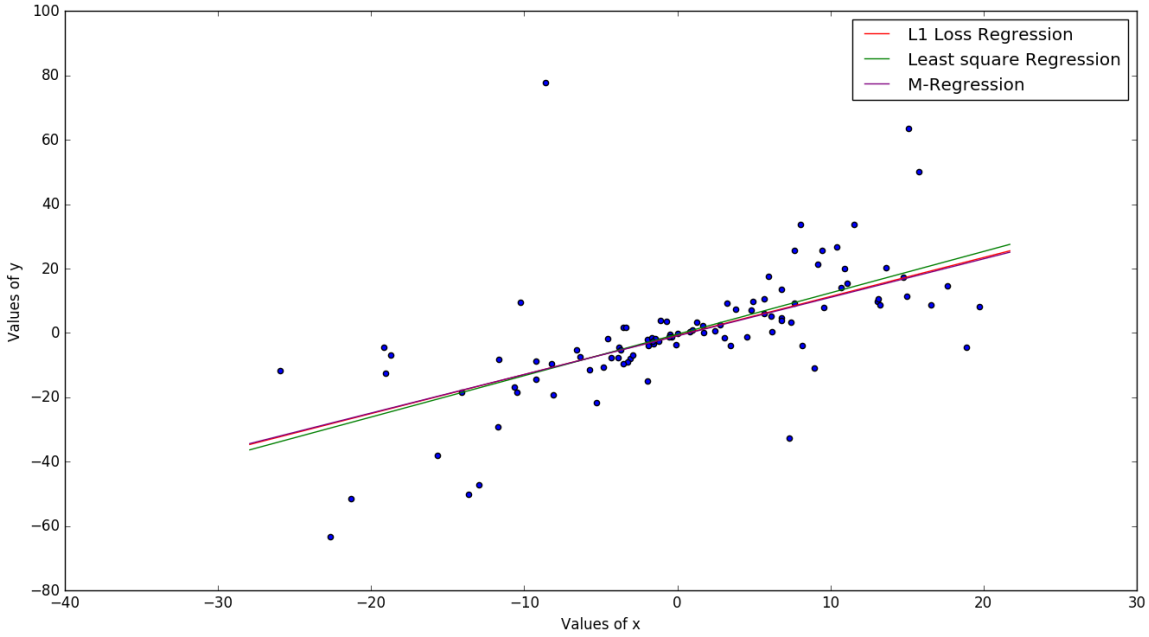

FIGURE 4. **Linear regression of the data set 2 using three models.**

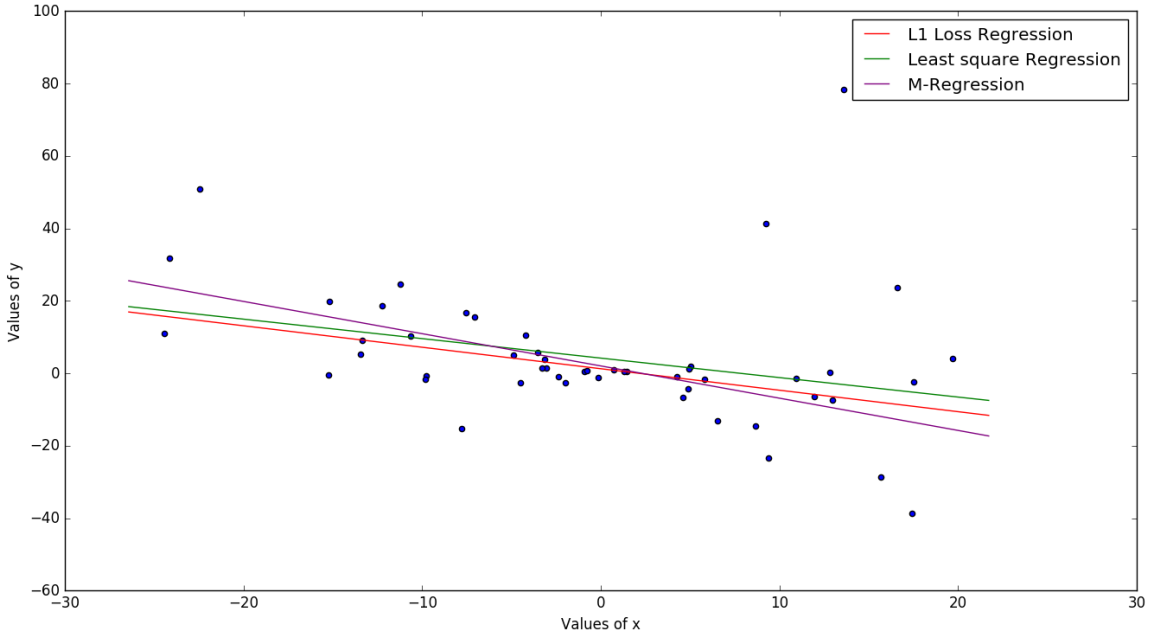

FIGURE 5. **Linear regression of the data set 3 using three models.**

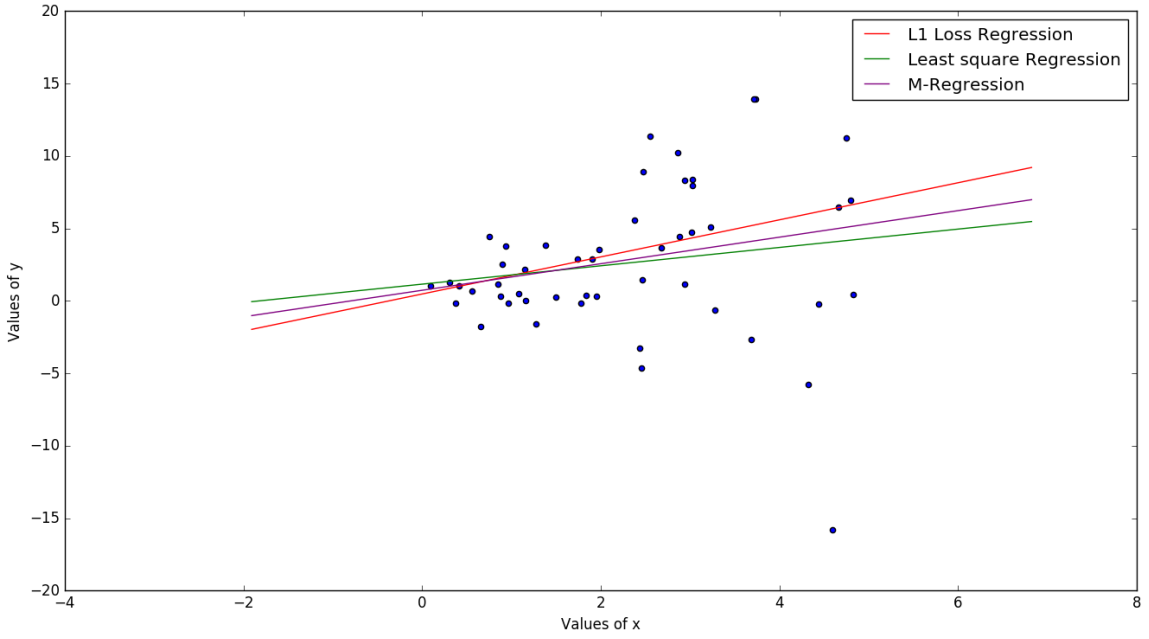

FIGURE 6. **Linear regression of the data set 4 using three models.**

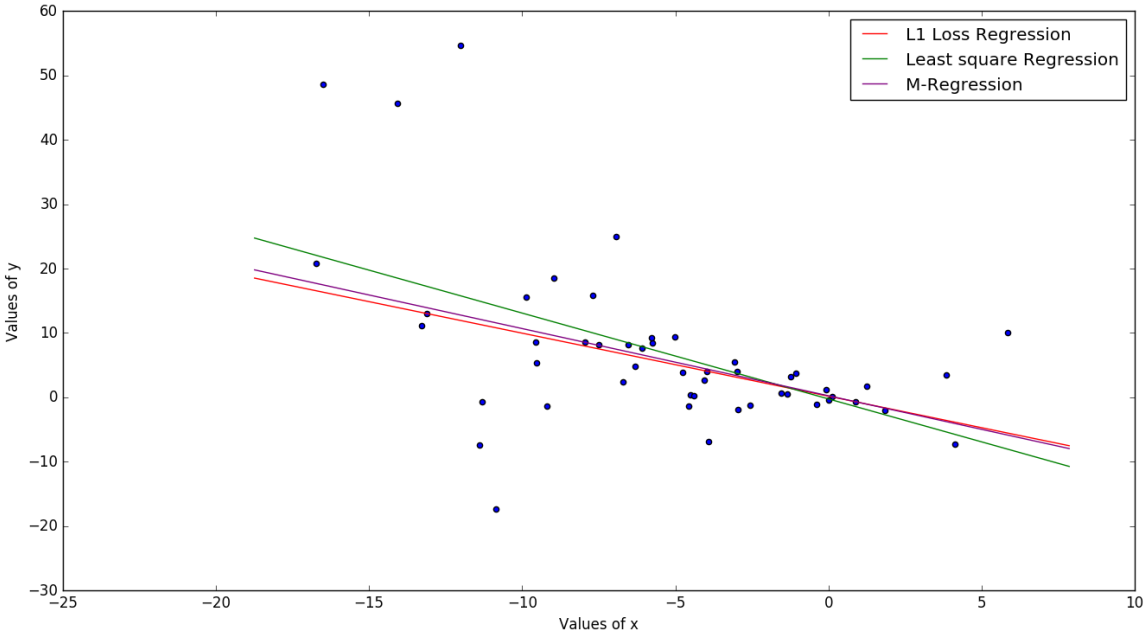

FIGURE 7. **Linear regression of the data set 5 using three models.**

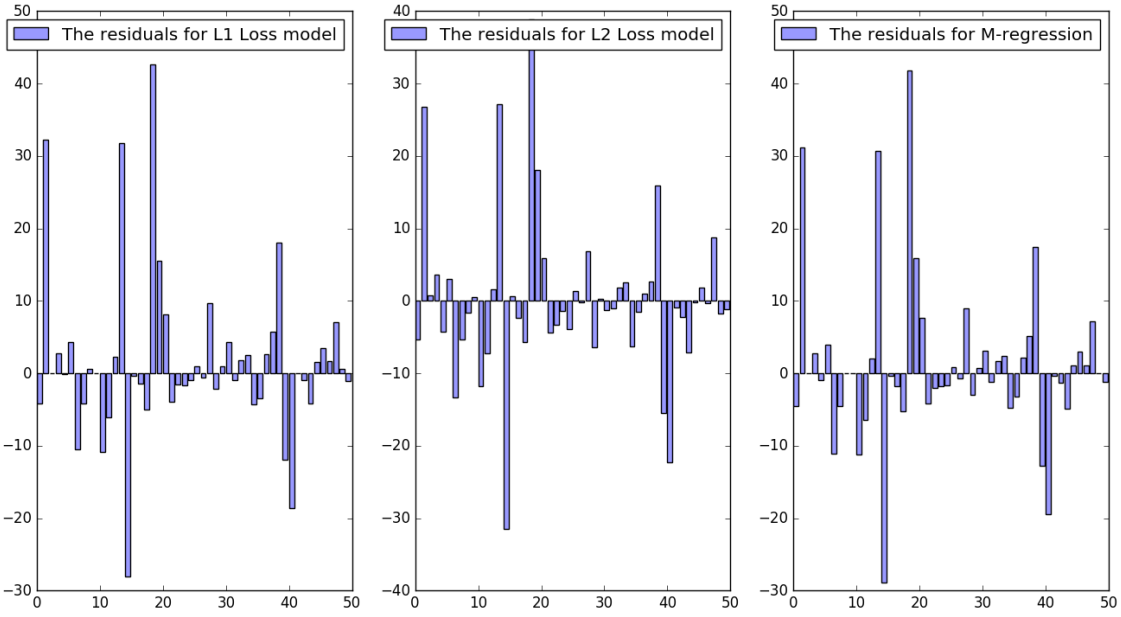

FIGURE 8. **Residuals of regressions for all the models in the data set 4.**