Robust Regression (Huber Regression)

March 8, 2021

1 Introduction

Let's make a quick summary of what we have:
A set of data points $\{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_n, y_n)\}$ where $\boldsymbol{x}_i \in \mathbb{R}^d$. Huber regression is solving this optimization problem:

$$\text{minimize} \sum_{i=1}^{n} \phi(\boldsymbol{\theta}^T \boldsymbol{x}_i - y_i) \ (1)$$

where variable $\boldsymbol{\theta} \in \mathbb{R}^d$ anh ϕ is a Huber penalty function:

$$\phi(x) = \begin{cases} x^2, \text{ if } |x| \leq M, \\ M(2|x| - M) \text{ otherwise} \end{cases}$$

Reader can easily verify that Huber function is a convex function therefore this problem is a convex optimization problem
Note that, in this document, whenever I mention vector, it means that it is a column vector.
Before we left, let's take a look to the differences between Huber Regression and Linear Regression

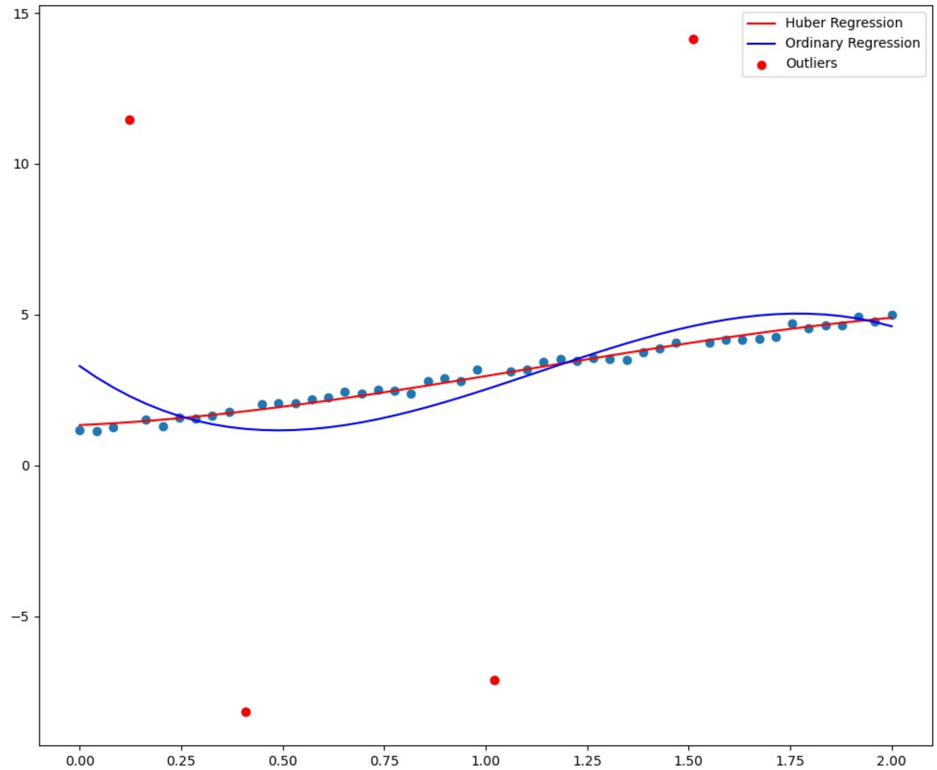

Figure 1: Compare Huber Polynomial Regression and Ordinary Polynomial Regression

As you can see, Huber Regression is **sensitive with outliers** whereas the Ordinary Regression is easily 'tilted' by outliers

2 Rewrite problem

Simply let $u_i = \boldsymbol{\theta}^T \boldsymbol{x}_i - y_i \ \forall i$, then the original problem will become:

$$\begin{aligned} &\text{minimize } \sum_{i=1}^{n} \phi(u_i) \\ &\text{subject to } u_i = \boldsymbol{\theta}^T \boldsymbol{x}_i - y_i \ \forall i \ (2) \end{aligned}$$

Let:

- $f : \mathbb{R}^{2d} \longrightarrow \mathbb{R}$ where $f(\boldsymbol{x}) = \sum_{i=1}^{n} \phi(x_i)$

- $\boldsymbol{t} = \begin{pmatrix} u_1 \\ u_2 \\ \cdots \\ u_n \\ \boldsymbol{\theta} \end{pmatrix}$

- $A = \begin{pmatrix} -\boldsymbol{I}_d & \mathbf{1} & \boldsymbol{X} \end{pmatrix}$ where matrix $\boldsymbol{X} = \begin{pmatrix} \boldsymbol{x}_1^T \\ \boldsymbol{x}_2^T \\ \cdots \\ \boldsymbol{x}_n^T \end{pmatrix}$

- $\boldsymbol{b} = \begin{pmatrix} y_1 \\ y_2 \\ \cdots \\ y_n \end{pmatrix}$

Then, problem (2) will become:

$$\begin{aligned} &\text{minimize } f(\boldsymbol{t}) \\ &\text{subject to } A\boldsymbol{t} = \boldsymbol{b} \text{ (3)} \end{aligned}$$

(3) is a equality constrain convex optimization problem that we need
If you read the code, there is a part where initial point might be infeasible due to the fact that we have to solve equation $A\boldsymbol{t} = b$ to get initial starting point. We can reformulate problem (3) as:

$$\begin{aligned} &\text{minimize } f(\boldsymbol{t}) \\ &\text{subject to } A\boldsymbol{t} = \boldsymbol{b} + \epsilon \text{ (3')} \end{aligned}$$

Whereas $\epsilon = A\boldsymbol{t_0} - b$, t_0 is a solution to least square problem $\|A\boldsymbol{t} - b\|$. Hence, t_0 is feasible for problem (3') (**not** problem (3))
With some re-assign some variables, problem (3') will go back to the problem (3) (however, it is **not an equivalent problem to original**). However, one can justify that the solution to problem (3') will be 'close' to the solution for problem (3) ('close' in the sense of Euclidean norm)

3 Compute its gradient ∇f and its Hessian $\nabla^2 f$

Reader can verify that:

$$\nabla f = \begin{pmatrix} \phi'(x_1) \\ \phi'(x_2) \\ \cdots \\ \phi'(x_n) \\ 0 \\ \cdots \\ 0 \end{pmatrix}$$

$$\nabla^2 f = \text{diag}(a_1, a_2, \cdots, a_{2n}) = \begin{pmatrix} a_1 & 0 & \cdots & 0 \\ 0 & a_2 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & a_{2n} \end{pmatrix} \text{ where } a_i = \begin{cases} 2, & \text{if } |x_i| \leq M, i = 1, 2, \cdots, n \\ 0, & \text{otherwise} \end{cases}$$

That's it! That's all you need to know to understand what I am doing in the code. Reader want to know about the algorithm can check [1] in the **References** section

References

[1] Stephen Boyd, Lieven Vandenberghe, *Convex Optimization*.

[2] My figure and code on GitHub:
`https://github.com/PhuThanh-Nguyen/Small-Projects/tree/main/Robust%20Regression`