Hinging Hyperplane Regression

A Presentation by

Abhisek Maiti & Avinandan Roy

Indian Statistical Institute

November 5, 2023

Table of Contents

Table of Contents

Introduction

- Hinging Hyperplanes (HHs) were proposed by Breiman (1993) in 1993 as versatile piece-wise linear models for regression, classification, and function approximation tasks. They offer flexibility and are particularly useful when dealing with complex data patterns, making them a valuable tool in machine learning and statistical modeling.

Introduction

- Hinging Hyperplanes (HHs) were proposed by Breiman (1993) in 1993 as versatile piece-wise linear models for regression, classification, and function approximation tasks. They offer flexibility and are particularly useful when dealing with complex data patterns, making them a valuable tool in machine learning and statistical modeling.
- There has been a large activity during the past years in the field of nonlinear function approximation. Many interesting results have been reported in connection with, for example the projection pursuit regression in neural network and the recent wavelets approach .These methods are closely related to the hinging hyperplane HH model investigated here.

Table of Contents

Setup and Definitions

Suppose we have an $(M+1)$-dimensional space

Setup and Definitions

Suppose we have an $(M+1)$-dimensional space $(y, x_1, x_2, \ldots, x_M)$.

Hinge Function and Hinge

A *hinge function* $y = h(x)$,consists of two hyperplanes continuously joined together. Taking (intercept term),$x_o \equiv 1$ and using $\cdot$ to denote the inner product of two vectors, if the two hyperplanes are given by:

$$y = \beta^+ \cdot \boldsymbol{x} \qquad y = \beta^- \cdot \boldsymbol{x}$$

they are joined together on $\{\boldsymbol{x} : (\beta^+ - \beta^-) \cdot \boldsymbol{x} = 0\}$ and we refer to $\Delta = (\beta^+ - \beta^-)$, or any multiple of Δ, as the *hinge* for the function.

Setup and Definitions

Suppose we have an $(M+1)$-dimensional space $(y, x_1, x_2, \ldots, x_M)$.

Hinge Function and Hinge

A *hinge function* $y = h(x)$,consists of two hyperplanes continuously joined together. Taking (intercept term),$x_o \equiv 1$ and using $\cdot$ to denote the inner product of two vectors, if the two hyperplanes are given by:

$$y = \beta^+ \cdot \boldsymbol{x} \qquad y = \beta^- \cdot \boldsymbol{x}$$

they are joined together on $\{\boldsymbol{x} : (\beta^+ - \beta^-) \cdot \boldsymbol{x} = 0\}$ and we refer to $\Delta = (\beta^+ - \beta^-)$, or any multiple of Δ, as the *hinge* for the function.

Explicit form of the Hinge

The explicit form of the hinge function is either
$\max(\beta^+ \cdot \boldsymbol{x}, \beta^- \cdot \boldsymbol{x})$ *or* $\min(\beta^+ \cdot \boldsymbol{x}, \beta^- \cdot \boldsymbol{x})$

Table of Contents

Hinge Finding Algorithm

1. First we set an initial Hinge $\Delta^{(0)}$.

Hinge Finding Algorithm

1. First we set an initial Hinge $\Delta^{(0)}$.
2. Make two set S_+ and S_- where $S_+ = \{x : \Delta^0.x \geq 0\}$ and $S_- = \{x : \Delta^0.x < 0\}$

Hinge Finding Algorithm

1. First we set an initial Hinge $\Delta^{(0)}$.
2. Make two set S_+ and S_- where $S_+ = \{x : \Delta^0.x \geq 0\}$ and $S_- = \{x : \Delta^0.x < 0\}$
3. Then, we fit linear model using OLS on each of the data-sets.

Hinge Finding Algorithm

1. First we set an initial Hinge $\Delta^{(0)}$.
2. Make two set S_+ and S_- where $S_+ = \{x : \Delta^0.x \geq 0\}$ and $S_- = \{x : \Delta^0.x < 0\}$
3. Then, we fit linear model using OLS on each of the data-sets.
4. Now, we take the least square coefficients of a hyperplane fitted to the y-values in S_+ are $\beta_+^{(1)}$ and those is the S_- fit are $\beta_-^{(1)}$.

Hinge Finding Algorithm

1. First we set an initial Hinge $\Delta^{(0)}$.
2. Make two set S_+ and S_- where $S_+ = \{x : \Delta^0.x \geq 0\}$ and $S_- = \{x : \Delta^0.x < 0\}$
3. Then, we fit linear model using OLS on each of the data-sets.
4. Now, we take the least square coefficients of a hyperplane fitted to the y-values in S_+ are $\beta_+^{(1)}$ and those is the S_- fit are $\beta_-^{(1)}$.
5. Now, we update our hinge by: $\Delta^{(1)} = \beta_+^{(1)} - \beta_-^{(1)}$

Hinge Finding Algorithm

1. First we set an initial Hinge $\Delta^{(0)}$.
2. Make two set S_+ and S_- where $S_+ = \{x : \Delta^0.x \geq 0\}$ and $S_- = \{x : \Delta^0.x < 0\}$
3. Then, we fit linear model using OLS on each of the data-sets.
4. Now, we take the least square coefficients of a hyperplane fitted to the y-values in S_+ are $\beta_+^{(1)}$ and those is the S_- fit are $\beta_-^{(1)}$.
5. Now, we update our hinge by: $\Delta^{(1)} = \beta_+^{(1)} - \beta_-^{(1)}$
6. Again we return to step 2 and execute the process again and again

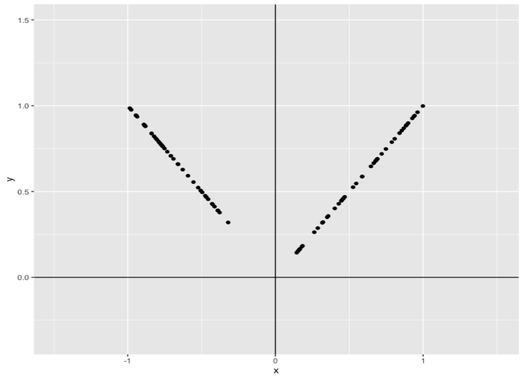
Hinge Finding Algorithm

1. First we set an initial Hinge $\Delta^{(0)}$.
2. Make two set S_+ and S_- where $S_+ = \{x : \Delta^0.x \geq 0\}$ and $S_- = \{x : \Delta^0.x < 0\}$
3. Then, we fit linear model using OLS on each of the data-sets.
4. Now, we take the least square coefficients of a hyperplane fitted to the y-values in S_+ are $\beta_+^{(1)}$ and those is the S_- fit are $\beta_-^{(1)}$.
5. Now, we update our hinge by: $\Delta^{(1)} = \beta_+^{(1)} - \beta_-^{(1)}$
6. Again we return to step 2 and execute the process again and again
7. Now, let at k-th step we get $\Delta^{(k)}$ as our updated hinge.We find $cos(\Delta^{(k)}, \Delta^{(k-1)})$. If it becomes ≥0.99 we stop other wise we go to step 2 and continue our process.

Hinge Finding Algorithm

Example

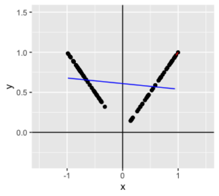
Suppose in a 2 dimensional space a hinge function $y = h(\boldsymbol{x})$ exists and we have the observations $(y_1, x_1), \ldots, (y_n, x_n)$ from $y = |x|$. Upon plotting in 2D let the data set looks like:



Hinge Finding Algorithm

Example

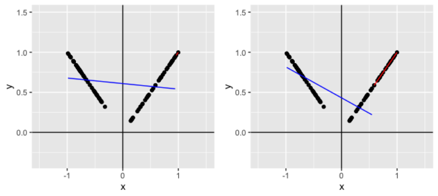
Now, we run the Hinge finding algorithm and the simulations are shown here:



Hinge Finding Algorithm

Example

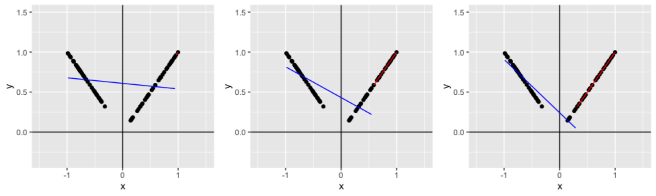
Now, we run the Hinge finding algorithm and the simulations are shown here:



Hinge Finding Algorithm

Example

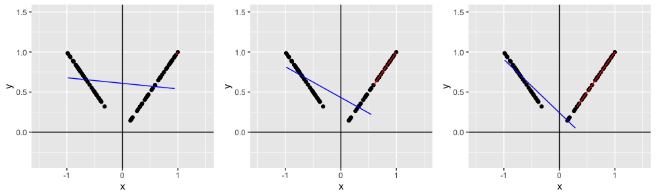
Now, we run the Hinge finding algorithm and the simulations are shown here:

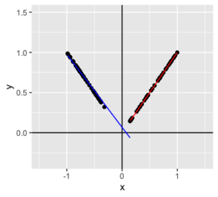


Hinge Finding Algorithm

Example

Now, we run the Hinge finding algorithm and the simulations are shown here:





Hinge Finding Algorithm

Example

Now, we run the Hinge finding algorithm and the simulations are shown here:

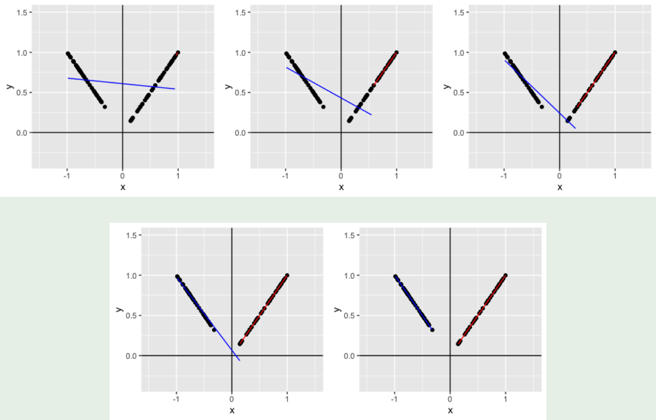


Table of Contents

Motivation towards Regression

- In a simple linear regression model, we assume that the relationship between variables is linear.

Motivation towards Regression

- In a simple linear regression model, we assume that the relationship between variables is linear.
- However, real-world scenarios often involve non-linear relationships, challenging this assumption.

Motivation towards Regression

- In a simple linear regression model, we assume that the relationship between variables is linear.
- However, real-world scenarios often involve non-linear relationships, challenging this assumption.
- For instance, Boyle's law describes how the *pressure* of an ideal gas relates inversely to its *volume*, a non-linear relationship.

Motivation towards Regression

- In a simple linear regression model, we assume that the relationship between variables is linear.
- However, real-world scenarios often involve non-linear relationships, challenging this assumption.
- For instance, Boyle's law describes how the *pressure* of an ideal gas relates inversely to its *volume*, a non-linear relationship.
- Another example can be the *distance* covered by a projectile is a sin function of *2*angel of projection*

Motivation towards Regression

- In a simple linear regression model, we assume that the relationship between variables is linear.
- However, real-world scenarios often involve non-linear relationships, challenging this assumption.
- For instance, Boyle's law describes how the *pressure* of an ideal gas relates inversely to its *volume*, a non-linear relationship.
- Another example can be the *distance* covered by a projectile is a sin function of *2*angel of projection*
- In such cases we try to get the relation between the response variable and the co-variate using regression.But instead of using simple linear models we approach for non-linear regression techniques

Motivation towards Regression

- In a simple linear regression model, we assume that the relationship between variables is linear.
- However, real-world scenarios often involve non-linear relationships, challenging this assumption.
- For instance, Boyle's law describes how the *pressure* of an ideal gas relates inversely to its *volume*, a non-linear relationship.
- Another example can be the *distance* covered by a projectile is a sin function of *2*angel of projection*
- In such cases we try to get the relation between the response variable and the co-variate using regression.But instead of using simple linear models we approach for non-linear regression techniques
- Hinging Hyperplane is one of the choices for non-linear regression methods and it fits a piece-wise continuous function to the data

Table of Contents

Noiseless Function Approximation

- Up to now we have discussed how to find a hinge for a hinge function when the hinge is unknown.

Noiseless Function Approximation

- Up to now we have discussed how to find a hinge for a hinge function when the hinge is unknown.
- Now we can think any function as a addition of multiple hinge functions.

Noiseless Function Approximation

- Up to now we have discussed how to find a hinge for a hinge function when the hinge is unknown.
- Now we can think any function as a addition of multiple hinge functions.
- We fit hinge function to get one corner point but in non-linear functions we may need multiple corner point to approximate the function

Noiseless Function Approximation

- Up to now we have discussed how to find a hinge for a hinge function when the hinge is unknown.
- Now we can think any function as a addition of multiple hinge functions.
- We fit hinge function to get one corner point but in non-linear functions we may need multiple corner point to approximate the function
- In the following slides, we shall discuss the method to fit multiple hinge functions.

Noiseless Function Approximation

Algorithm for fitting two hinge functions

- Given a set of observations from $(M+1)$ dimensional space, $\{(y_i, \boldsymbol{x_i})\}_{i=1}^{N}$

Noiseless Function Approximation

Algorithm for fitting two hinge functions

- Given a set of observations from $(M+1)$ dimensional space, $\{(y_i, \boldsymbol{x_i})\}_{i=1}^{N}$
- Fit h_1 by running the HFA on the data

Noiseless Function Approximation

Algorithm for fitting two hinge functions

- Given a set of observations from $(M+1)$ dimensional space, $\{(y_i, \boldsymbol{x_i})\}_{i=1}^{N}$
- Fit h_1 by running the HFA on the data
- After that, we calculate the residuals,i.e $\tilde{y}_{[1]} = y - h_1$

Noiseless Function Approximation

Algorithm for fitting two hinge functions

- Given a set of observations from $(M+1)$ dimensional space, $\{(y_i, \boldsymbol{x_i})\}_{i=1}^{N}$
- Fit h_1 by running the HFA on the data
- After that, we calculate the residuals,i.e $\tilde{y}_{[1]} = y - h_1$
- Again we run the HFA on residual data $\{\tilde{y}_{[1]}, \boldsymbol{x}\}$ to fit the second hinge h_2

Noiseless Function Approximation

Algorithm for fitting two hinge functions

- Given a set of observations from $(M+1)$ dimensional space, $\{(y_i, \boldsymbol{x_i})\}_{i=1}^{N}$
- Fit h_1 by running the HFA on the data
- After that, we calculate the residuals,i.e $\tilde{y}_{[1]} = y - h_1$
- Again we run the HFA on residual data $\{\tilde{y}_{[1]}, \boldsymbol{x}\}$ to fit the second hinge h_2
- Refit: Run the HFA on residuals $\tilde{y}_{[2]} = y - h_2$ to re-estimate h_1.

Noiseless Function Approximation

Algorithm for fitting two hinge functions

- Given a set of observations from $(M+1)$ dimensional space, $\{(y_i, \boldsymbol{x_i})\}_{i=1}^N$
- Fit h_1 by running the HFA on the data
- After that, we calculate the residuals,i.e $\tilde{y}_{[1]} = y - h_1$
- Again we run the HFA on residual data $\{\tilde{y}_{[1]}, \boldsymbol{x}\}$ to fit the second hinge h_2
- Refit: Run the HFA on residuals $\tilde{y}_{[2]} = y - h_2$ to re-estimate h_1.
- Refit h_2 again on the new updated residual data $\{\tilde{y}_{[1]}, \boldsymbol{x}\}$

Noiseless Function Approximation

Algorithm for fitting two hinge functions

- Given a set of observations from $(M+1)$ dimensional space, $\{(y_i, \boldsymbol{x_i})\}_{i=1}^N$
- Fit h_1 by running the HFA on the data
- After that, we calculate the residuals,i.e $\tilde{y}_{[1]} = y - h_1$
- Again we run the HFA on residual data $\{\tilde{y}_{[1]}, \boldsymbol{x}\}$ to fit the second hinge h_2
- Refit: Run the HFA on residuals $\tilde{y}_{[2]} = y - h_2$ to re-estimate h_1.
- Refit h_2 again on the new updated residual data $\{\tilde{y}_{[1]}, \boldsymbol{x}\}$
- Update h_1 and h_2 in this process until there is no significant change in residual sum of squares in the iterations

Noiseless Function Approximation

Algorithm for fitting K hinge functions

Now, we shall describe the algorithm to fit K many hinge functions to our data, assuming we have fitted $(K-1)$ many hinge functions to our data

1. We have residuals $\tilde{y}_{[1,2,\dots,(K-1)]} = f(x) - \sum_{i=1}^{K-1} h_i(x)$

Noiseless Function Approximation

Algorithm for fitting K hinge functions

Now, we shall describe the algorithm to fit K many hinge functions to our data, assuming we have fitted $(K-1)$ many hinge functions to our data

1. We have residuals $\tilde{y}_{[1,2,\ldots,(K-1)]} = f(x) - \sum_{i=1}^{K-1} h_i(x)$
2. Then we fit the K th hinge on the residual data $\{\tilde{y}_{[1,2,\ldots,(K-1)]}, \boldsymbol{x}\}$ to get h_k

Noiseless Function Approximation

Algorithm for fitting K hinge functions

Now, we shall describe the algorithm to fit K many hinge functions to our data, assuming we have fitted $(K-1)$ many hinge functions to our data

1. We have residuals $\tilde{y}_{[1,2,\dots,(K-1)]} = f(x) - \sum_{i=1}^{K-1} h_i(x)$
2. Then we fit the K th hinge on the residual data $\{\tilde{y}_{[1,2,\dots,(K-1)]}, \boldsymbol{x}\}$ to get h_k
3. Now we calculate the residual data $\tilde{y}_{[1,2,\dots,K]}$

Noiseless Function Approximation

Algorithm for fitting K hinge functions

Now, we shall describe the algorithm to fit K many hinge functions to our data, assuming we have fitted $(K-1)$ many hinge functions to our data

1. We have residuals $\tilde{y}_{[1,2,\ldots,(K-1)]} = f(x) - \sum_{i=1}^{K-1} h_i(x)$
2. Then we fit the K th hinge on the residual data $\{\tilde{y}_{[1,2,\ldots,(K-1)]}, \boldsymbol{x}\}$ to get h_k
3. Now we calculate the residual data $\tilde{y}_{[1,2,\ldots,K]}$
4. We update h_1 refitting the difference $\tilde{y}_{[2,3,\ldots,K]} = f(x) - \sum_{i=2}^{K-1} h_i(x)$

Noiseless Function Approximation

Algorithm for fitting K hinge functions

Now, we shall describe the algorithm to fit K many hinge functions to our data, assuming we have fitted $(K-1)$ many hinge functions to our data

1. We have residuals $\tilde{y}_{[1,2,\ldots,(K-1)]} = f(x) - \sum_{i=1}^{K-1} h_i(x)$
2. Then we fit the K th hinge on the residual data $\{\tilde{y}_{[1,2,\ldots,(K-1)]}, \boldsymbol{x}\}$ to get h_k
3. Now we calculate the residual data $\tilde{y}_{[1,2,\ldots,K]}$
4. We update h_1 refitting the difference $\tilde{y}_{[2,3,\ldots,K]} = f(x) - \sum_{i=2}^{K-1} h_i(x)$
5. Update the residual $\tilde{y}_{[1,2,\ldots,K]}$

Noiseless Function Approximation

Algorithm for fitting K hinge functions

Now, we shall describe the algorithm to fit K many hinge functions to our data, assuming we have fitted $(K-1)$ many hinge functions to our data

1. We have residuals $\tilde{y}_{[1,2,\ldots,(K-1)]} = f(x) - \sum_{i=1}^{K-1} h_i(x)$
2. Then we fit the K th hinge on the residual data $\{\tilde{y}_{[1,2,\ldots,(K-1)]}, \boldsymbol{x}\}$ to get h_k
3. Now we calculate the residual data $\tilde{y}_{[1,2,\ldots,K]}$
4. We update h_1 refitting the difference $\tilde{y}_{[2,3,\ldots,K]} = f(x) - \sum_{i=2}^{K-1} h_i(x)$
5. Update the residual $\tilde{y}_{[1,2,\ldots,K]}$
6. Again we update h_2 refitting the difference:$\tilde{y}_{[1,3,4,\ldots,K]} = f(x) - h_1(x) - \sum_{i=3}^{K-1} h_i(x)$

Noiseless Function Approximation

Algorithm for fitting K hinge functions

Now, we shall describe the algorithm to fit K many hinge functions to our data, assuming we have fitted $(K-1)$ many hinge functions to our data

1. We have residuals $\tilde{y}_{[1,2,\ldots,(K-1)]} = f(x) - \sum_{i=1}^{K-1} h_i(x)$
2. Then we fit the K th hinge on the residual data $\{\tilde{y}_{[1,2,\ldots,(K-1)]}, \boldsymbol{x}\}$ to get h_k
3. Now we calculate the residual data $\tilde{y}_{[1,2,\ldots,K]}$
4. We update h_1 refitting the difference $\tilde{y}_{[2,3,\ldots,K]} = f(x) - \sum_{i=2}^{K-1} h_i(x)$
5. Update the residual $\tilde{y}_{[1,2,\ldots,K]}$
6. Again we update h_2 refitting the difference:$\tilde{y}_{[1,3,4,\ldots,K]} = f(x) - h_1(x) - \sum_{i=3}^{K-1} h_i(x)$
7. In this procedure we update all the K hinge functions by refitting

Noiseless Function Approximation

Algorithm for fitting K hinge functions

- Repeat steps 4-7 until there is no significant change in the RSS.

Noiseless Function Approximation

Algorithm for fitting K hinge functions

- Repeat steps 4-7 until there is no significant change in the RSS.

Theorem:Convergence of Multiple Hinge Algorithm

Let P be any measure with compact support on E^M and $f(x)$ any sufficiently smooth function.Let $\tilde{f}(w)$ be the Fourier transformation of $f(x)$ If support of P contained in the sphere of radius R,and if

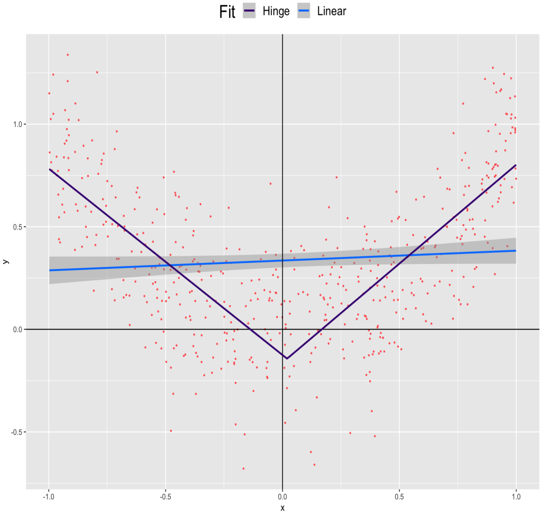
$$\int ||w||^2 |f(w)|\, dw = c < \infty,$$

then there are hinge $h_1, h_2 ...., h_K$ such that

$$\left|\left| f - \sum_{i=1}^{K} h_i \right|\right|^2 \leq \frac{(2R)^4 c^2}{K}$$

Table of Contents

$y = x^2 + \epsilon$,$\epsilon \sim N(0, 1/4)$



- SLR solution:

$$\hat{y} = \begin{pmatrix} 0.335 \\ 0.047 \end{pmatrix}^T \cdot \boldsymbol{x}$$

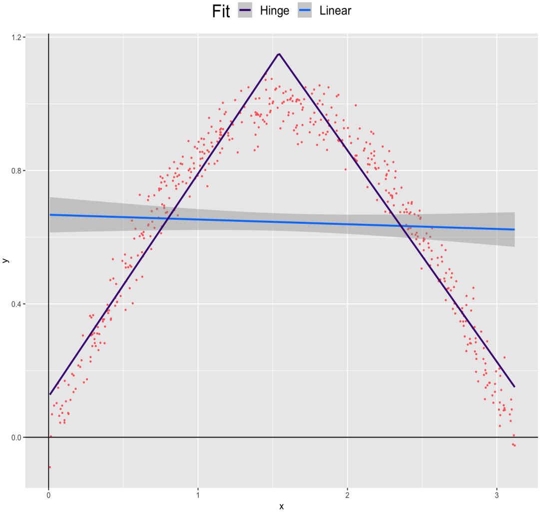
$RSS = 70.80669$

- HHR solution:

$$\hat{y} = \max \left\{ \begin{pmatrix} -0.125 \\ -0.908 \end{pmatrix}^T \cdot \boldsymbol{x}, \begin{pmatrix} -0.162 \\ 0.964 \end{pmatrix}^T \cdot \boldsymbol{x} \right\}$$

$RSS = 33.659$

$y = sin(x) + \epsilon \ , \epsilon \sim N(0, 1/4)$



- SLR solution:

$$\hat{y} = \begin{pmatrix} 0.667 \\ -0.014 \end{pmatrix}^T \cdot \boldsymbol{x}$$

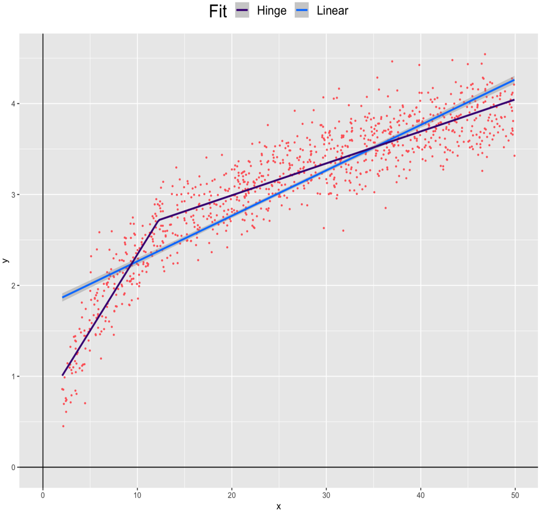
$RSS = 44.10121$

- HHR solution:

$$\hat{y} = \min \left\{ \begin{pmatrix} 0.122 \\ 0.668 \end{pmatrix}^T \cdot \boldsymbol{x}, \begin{pmatrix} 2.128 \\ -0.634 \end{pmatrix}^T \cdot \boldsymbol{x} \right\}$$

$RSS = 3.33$

$y = log(x) + \epsilon \ ,\epsilon \sim N(0, 1/4)$



- SLR solution:

$$\hat{y} = \begin{pmatrix} 1.764 \\ 0.050 \end{pmatrix}^T \cdot \boldsymbol{x}$$

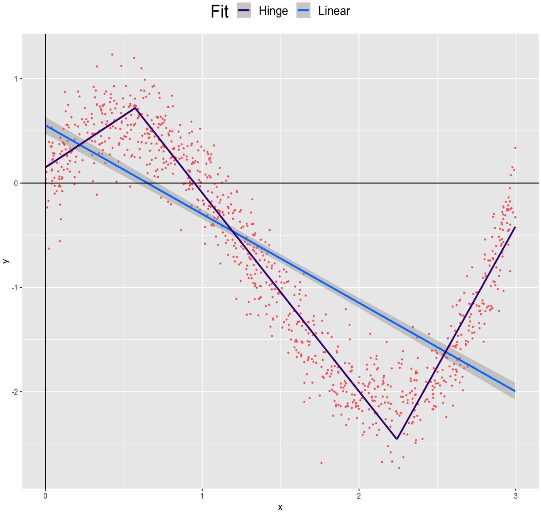
$RSS = 122.452$

- HHR solution:

$$\hat{y} = \min \left\{ \begin{pmatrix} 0.660 \\ 0.167 \end{pmatrix}^T \cdot \boldsymbol{x}, \begin{pmatrix} 2.287 \\ 0.035 \end{pmatrix}^T \cdot \boldsymbol{x} \right\}$$

$RSS = 66.72372$

$y = x(x-1)(x-3) + \epsilon$,$\epsilon \sim N(0, 1/4)$



- SLR solution:

$$\hat{y} = \begin{pmatrix} 0.5529 \\ -0.8506 \end{pmatrix}^T \cdot \boldsymbol{x}$$

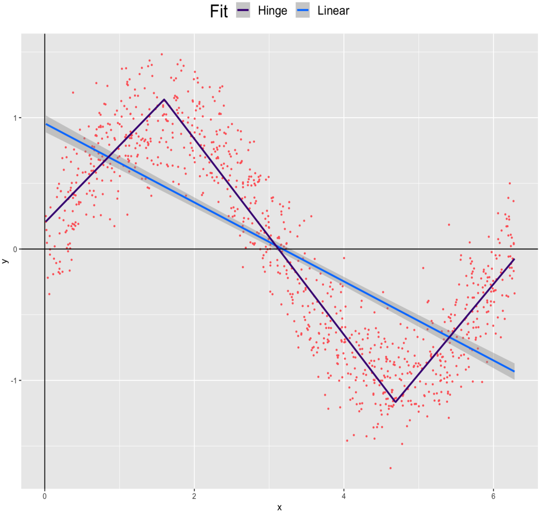
$RSS = 435.1966$

- HHR solution:

$$\hat{y} = \min\left\{ \begin{pmatrix} -0.538 \\ 1.772 \end{pmatrix}^T \cdot \boldsymbol{x}, \begin{pmatrix} 1.118 \\ -1.124 \end{pmatrix}^T \cdot \boldsymbol{x} \right\}$$
$$+ \max\left\{ \begin{pmatrix} 0.689 \\ -0.778 \end{pmatrix}^T \cdot \boldsymbol{x}, \begin{pmatrix} -9.619 \\ 3.821 \end{pmatrix}^T \cdot \boldsymbol{x} \right\}$$

$RSS = 73.407$

$y = sin(x) + \epsilon \ , \epsilon \sim N(0, 1/4), x \in [0, 2\pi]$



- SLR solution:

$$\hat{y} = \begin{pmatrix} 0.956 \\ -0.301 \end{pmatrix}^T \cdot \boldsymbol{x}$$

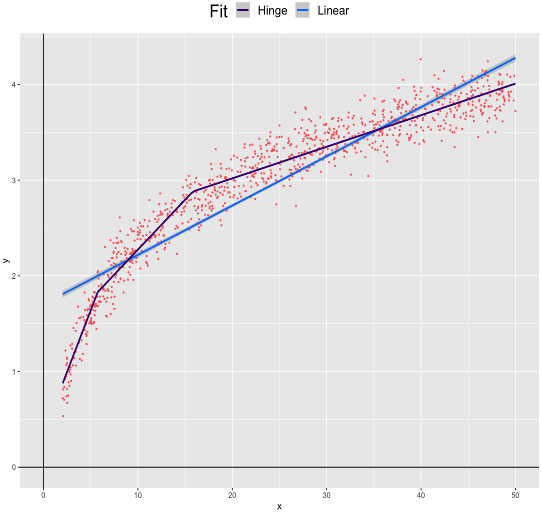
$RSS = 259.7786$

- HHR solution:

$$\hat{y} = \min\left\{ \begin{pmatrix} -0.206 \\ 0.827 \end{pmatrix}^T \cdot \boldsymbol{x}, \begin{pmatrix} 1.867 \\ -0.505 \end{pmatrix}^T \cdot \boldsymbol{x} \right\} + \max\left\{ \begin{pmatrix} 0.462 \\ -0.240 \end{pmatrix}^T \cdot \boldsymbol{x}, \begin{pmatrix} -6.250 \\ 1.191 \end{pmatrix}^T \cdot \boldsymbol{x} \right\}$$

$RSS = 70.15768$

$y = \log(x) + \epsilon$,$\epsilon \sim N(0, 1/6)$



- SLR solution:

$$\hat{y} = \begin{pmatrix} 0.317 \\ 0.20 \end{pmatrix}^T \cdot \boldsymbol{x}$$

$RSS = 71.693$

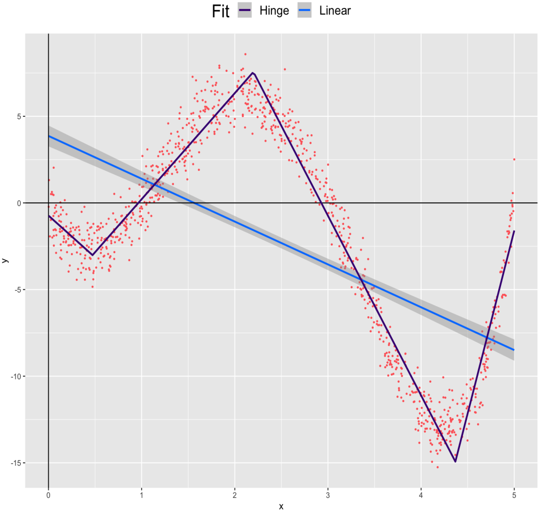
- HHR solution:

$$\hat{y} = \min\left\{ \begin{pmatrix} 0.926 \\ 0.201 \end{pmatrix}^T \cdot \boldsymbol{x}, \begin{pmatrix} 1.803 \\ 0.048 \end{pmatrix}^T \cdot \boldsymbol{x} \right\}$$

$$+ \min\left\{ \begin{pmatrix} 0.554 \\ -0.015 \end{pmatrix}^T \cdot \boldsymbol{x}, \begin{pmatrix} -0.572 \\ 0.055 \end{pmatrix}^T \cdot \boldsymbol{x} \right\}$$

$RSS = 29.08094$

$y = x(x-1)(x-3)(x-5) + \epsilon \ , \epsilon \sim N(0,1)$



- SLR solution:

$$\hat{y} = \begin{pmatrix} 3.866 \\ -2.472 \end{pmatrix}^T \cdot \boldsymbol{x}$$

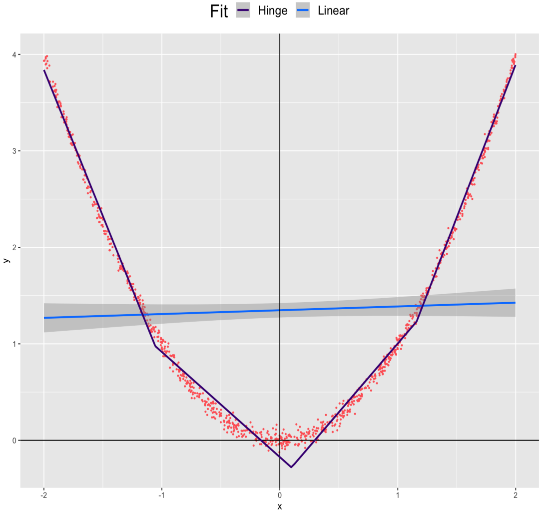
$RSS = 24600.87$

- HHR solution:

$$\hat{y} = \min\left\{ \begin{pmatrix} 29.374 \\ -9.355 \end{pmatrix}^T \cdot \boldsymbol{x}, \begin{pmatrix} -6.916 \\ 7.142 \end{pmatrix}^T \cdot \boldsymbol{x} \right\}$$
$$+ \max\left\{ \begin{pmatrix} 2.011 \\ -1.313 \end{pmatrix}^T \cdot \boldsymbol{x}, \begin{pmatrix} -135.478 \\ 30.163 \end{pmatrix}^T \cdot \boldsymbol{x} \right\}$$
$$+ \max\left\{ \begin{pmatrix} 4.174 \\ -10.654 \end{pmatrix}^T \cdot \boldsymbol{x}, \begin{pmatrix} -1.022 \\ 0.294 \end{pmatrix}^T \cdot \boldsymbol{x} \right\}$$

$RSS = 1439.839$

$y = x^2 + \epsilon, \epsilon \sim N(0, 1/20)$



- SLR solution:

$$\hat{y} = \begin{pmatrix} 1.348 \\ 0.039 \end{pmatrix}^T \cdot \boldsymbol{x}$$

$RSS = 1447.701$

- HHR solution:

$$\hat{y} = \max \left\{ \begin{pmatrix} -0.047 \\ 1.369 \end{pmatrix}^T \cdot \boldsymbol{x}, \begin{pmatrix} 0.202 \\ -1.146 \end{pmatrix}^T \cdot \boldsymbol{x} \right\}$$
$$+ \max \left\{ \begin{pmatrix} -0.158 \\ -0.209 \end{pmatrix}^T \cdot \boldsymbol{x}, \begin{pmatrix} -2.183 \\ 1.537 \end{pmatrix}^T \cdot \boldsymbol{x} \right\}$$
$$+ \max \left\{ \begin{pmatrix} -0.217 \\ 0.265 \end{pmatrix}^T \cdot \boldsymbol{x}, \begin{pmatrix} -2.268 \\ -1.677 \end{pmatrix}^T \cdot \boldsymbol{x} \right\}$$

$RSS = 10.17562$

Table of Contents

Number of Hinges used in Regression

- A very sensitive question arises that:"*How many hinges should we use in regression ?*"

Number of Hinges used in Regression

- A very sensitive question arises that:"*How many hinges should we use in regression ?*"
- If we fit less number of hinges it may lead to large value of MRSS

Number of Hinges used in Regression

- A very sensitive question arises that:"*How many hinges should we use in regression ?*"
- If we fit less number of hinges it may lead to large value of MRSS
- However, if we fit too many hinge functions it will lead to over-fitting

Number of Hinges used in Regression

- A very sensitive question arises that:"*How many hinges should we use in regression ?*"
- If we fit less number of hinges it may lead to large value of MRSS
- However, if we fit too many hinge functions it will lead to over-fitting
- Breiman et al. describes a quantity based on MRSS

$$\text{PE}_{GCV} = \frac{MRSS}{(1 - N_p/N)^2}, \quad \text{where } N_p = \frac{K+1}{M+1}$$

$$N = \#\text{of observations}$$

$$K = \#\text{of Hinges}$$

$$M = \text{dimension}$$

Number of Hinges used in Regression

- A very sensitive question arises that:"*How many hinges should we use in regression ?*"
- If we fit less number of hinges it may lead to large value of MRSS
- However, if we fit too many hinge functions it will lead to over-fitting
- Breiman et al. describes a quantity based on MRSS

$$\text{PE}_{GCV} = \frac{MRSS}{(1 - N_p/N)^2}, \quad \text{where } N_p = \frac{K+1}{M+1}$$

$$N = \#\text{of observations}$$

$$K = \#\text{of Hinges}$$

$$M = \text{dimension}$$

- We increase number of hinges and we notice at which K there is no significant change in PE_{GCV} further. We choose that first K as our total number of hinges.

Table of Contents

Derivation of Algorithm

- Assume that a data set $\{(y_i, \boldsymbol{x}_i)\}_{i=1}^{N}$ is given, and the objective is to fit a hinge function using HFA

Derivation of Algorithm

- Assume that a data set $\{(y_i, \boldsymbol{x}_i)\}_{i=1}^{N}$ is given, and the objective is to fit a hinge function using HFA
- Note that, we were using HFA where we were dividing the whole data set into two parts and we fitted least square estimates

Derivation of Algorithm

- Assume that a data set $\{(y_i, \boldsymbol{x}_i)\}_{i=1}^N$ is given, and the objective is to fit a hinge function using HFA
- Note that, we were using HFA where we were dividing the whole data set into two parts and we fitted least square estimates
- Our objective was to minimize

$$V_N(\boldsymbol{\beta}) = \frac{1}{2}\sum_{i=1}^{N}(y_i - h(x_i;\boldsymbol{\beta}))^2 \qquad \text{where,} \quad \boldsymbol{\beta} = \begin{pmatrix}\beta_+ \\ \beta_-\end{pmatrix}$$

Derivation of Algorithm

- Assume that a data set $\{(y_i, \boldsymbol{x}_i)\}_{i=1}^{N}$ is given, and the objective is to fit a hinge function using HFA
- Note that, we were using HFA where we were dividing the whole data set into two parts and we fitted least square estimates
- Our objective was to minimize

$$V_N(\boldsymbol{\beta}) = \frac{1}{2}\sum_{i=1}^{N}(y_i - h(x_i; \boldsymbol{\beta}))^2 \qquad \text{where,} \quad \boldsymbol{\beta} = \begin{pmatrix}\beta_+ \\ \beta_-\end{pmatrix}$$

- We want to get an estimate of $\boldsymbol{\beta}$ such that

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} V_N(\boldsymbol{\beta})$$

Finding MLE

- We try to find the MLE of β using Newton-Raphson Method now

Finding MLE

- We try to find the MLE of $\boldsymbol{\beta}$ using Newton-Raphson Method now
- So we will need to calculate the first derivative and the hessian of V_N w.r.t $\boldsymbol{\beta}$

The first derivative

$$\nabla V_N = \begin{bmatrix} \frac{\partial}{\partial \beta_+} V_N \\ \frac{\partial}{\partial \beta_-} V_N \end{bmatrix} = \begin{bmatrix} -\sum_{\boldsymbol{x}_i \in S_+} \boldsymbol{x}_i (y_i - \boldsymbol{x}_i^T \beta_+) \\ -\sum_{\boldsymbol{x}_i \in S_-} \boldsymbol{x}_i (y_i - \boldsymbol{x}_i^T \beta_-) \end{bmatrix}$$

The Hessian

$$\nabla^2 V_N = \begin{bmatrix} \sum_{\boldsymbol{x}_i \in S_+} \boldsymbol{x}_i \boldsymbol{x}_i^T & 0 \\ 0 & \sum_{\boldsymbol{x}_i \in S_-} \boldsymbol{x}_i \boldsymbol{x}_i^T \end{bmatrix}$$

Newton-Raphson updating step

- Now, we will update $\boldsymbol{\beta}$ using Newton-Raphson method. Let at k th step we have $\boldsymbol{\beta}_k$

Update

$$\boldsymbol{\beta}^{(k+1)} = \boldsymbol{\beta}^{(k)} - (\nabla^2 V_N)^{-1} \nabla V_N$$

$$= \boldsymbol{\beta}^{(k)} + \begin{bmatrix} (\sum_{\boldsymbol{x}_i \in S_+} \boldsymbol{x}_i \boldsymbol{x}_i^T)^{-1} \sum_{\boldsymbol{x}_i \in S_+} \boldsymbol{x}_i y_i - \boldsymbol{\beta}_+^{(k)} \\ (\sum_{\boldsymbol{x}_i \in S_-} \boldsymbol{x}_i \boldsymbol{x}_i^T)^{-1} \sum_{\boldsymbol{x}_i \in S_-} \boldsymbol{x}_i y_i - \boldsymbol{\beta}_-^{(k)} \end{bmatrix}$$

$$\boldsymbol{\beta}^{(k+1)} = \boldsymbol{\beta}^{(k)} + (\boldsymbol{\beta}_{Br}^{(k)} - \boldsymbol{\beta}^{(k)})$$

Newton-Raphson updating step

- Now, we will update $\boldsymbol{\beta}$ using Newton-Raphson method. Let at k th step we have $\boldsymbol{\beta}_k$

Update

$$\boldsymbol{\beta}^{(k+1)} = \boldsymbol{\beta}^{(k)} - (\nabla^2 V_N)^{-1} \nabla V_N$$

$$= \boldsymbol{\beta}^{(k)} + \begin{bmatrix} \left(\sum_{\boldsymbol{x}_i \in S_+} \boldsymbol{x}_i \boldsymbol{x}_i^T\right)^{-1} \sum_{\boldsymbol{x}_i \in S_+} \boldsymbol{x}_i y_i - \boldsymbol{\beta}_+^{(k)} \\ \left(\sum_{\boldsymbol{x}_i \in S_-} \boldsymbol{x}_i \boldsymbol{x}_i^T\right)^{-1} \sum_{\boldsymbol{x}_i \in S_-} \boldsymbol{x}_i y_i - \boldsymbol{\beta}_-^{(k)} \end{bmatrix}$$

$$\boldsymbol{\beta}^{(k+1)} = \boldsymbol{\beta}^{(k)} + (\boldsymbol{\beta}_{Br}^{(k)} - \boldsymbol{\beta}^{(k)})$$

- Notice that using Newton-Raphson we get the same update for $\boldsymbol{\beta}^{(k)}$ as we got in case of HFA

Damped Newton-Raphson to update

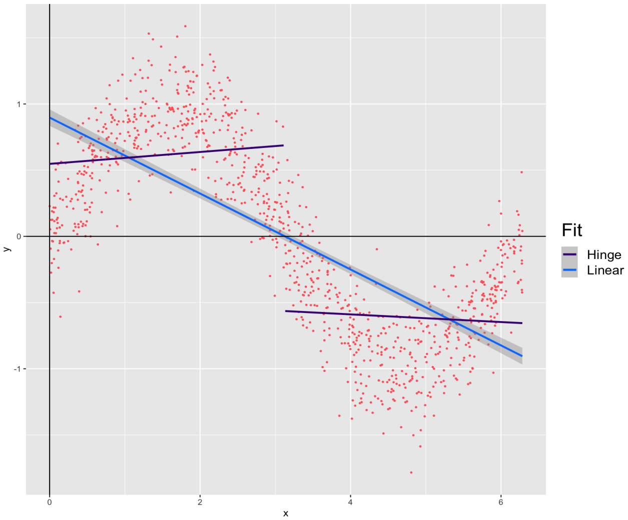
In damped Newton-Raphson we modify the step length and take relatively small step size

$$\boldsymbol{\beta}^{(k+1)} = \boldsymbol{\beta}^{(k)} + \mu(\boldsymbol{\beta}_{Br}^{(k)} - \boldsymbol{\beta}^{(k)})$$

we will take μ to be $1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}$..... and make step size small so that the method converge.

Table of Contents

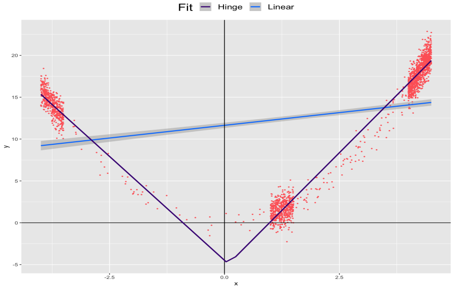
Futility:Convergence problem with any initial hinge

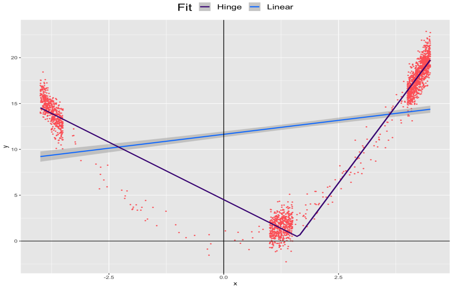


Example

- Here we have simulated data from $\{x, sin(x) + \epsilon\}$
- If we take our initial hinge at $x = \pi$, this HFA will never converge
- Breiman's algorithm of finding hinge fails here

Futility:Unstable Hinge Positions





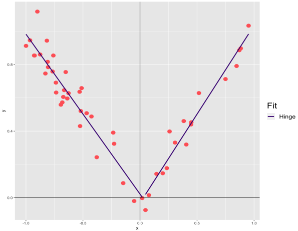
Example

- Suppose we have data from $\{x, x^2 + \epsilon\}$ and it is non-uniform over the intervals
- Then we get different convergent points. Hence the hinge becomes unstable.

Futility:Sensitive to outliers

Example

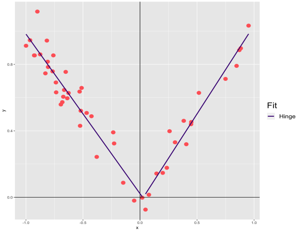
We have taken data from $\{x, |x| + \epsilon\}$.We have shown how it effects when there are outliers.

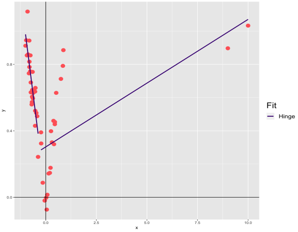


Futility:Sensitive to outliers

Example

We have taken data from $\{x, |x| + \epsilon\}$.We have shown how it effects when there are outliers.

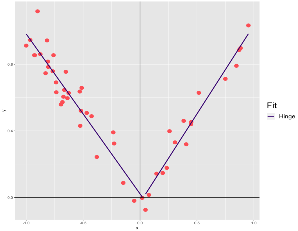


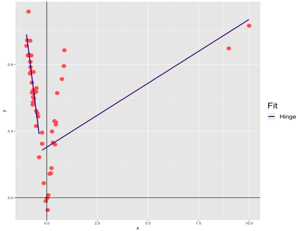


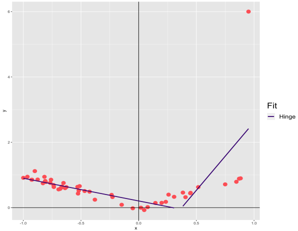
Futility:Sensitive to outliers

Example

We have taken data from $\{x, |x| + \epsilon\}$.We have shown how it effects when there are outliers.



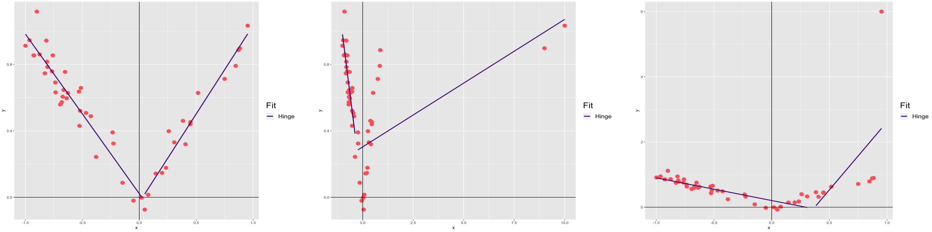




Futility:Sensitive to outliers

Example

We have taken data from $\{x, |x| + \epsilon\}$.We have shown how it effects when there are outliers.



Reason

We know that SLR is sensitive to outliers. As we are doing SLR in the partitioned sets in HHR , HHR becomes sensitive to outliers

Futility

- This method is computationally intensive
- Breiman's algorithm may lead to singularity problem on intermediate steps of HHR and may not converge

References

L. Breiman. Hinging hyperplanes for regression, classification, and function approximation. *IEEE Trans. Inf. Theory*, 39:999–1013, 1993. URL https://api.semanticscholar.org/CorpusID:12319558.

Thank You