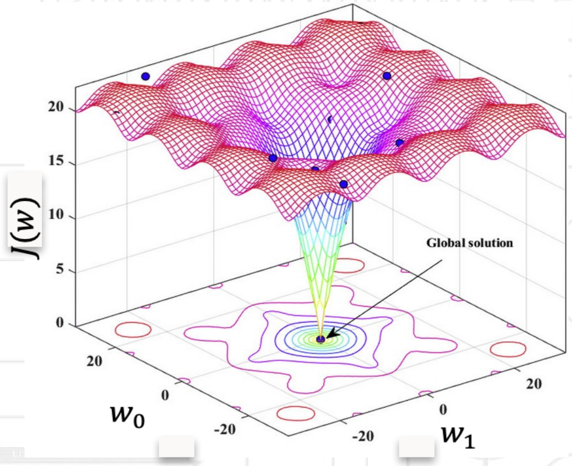
Module 1

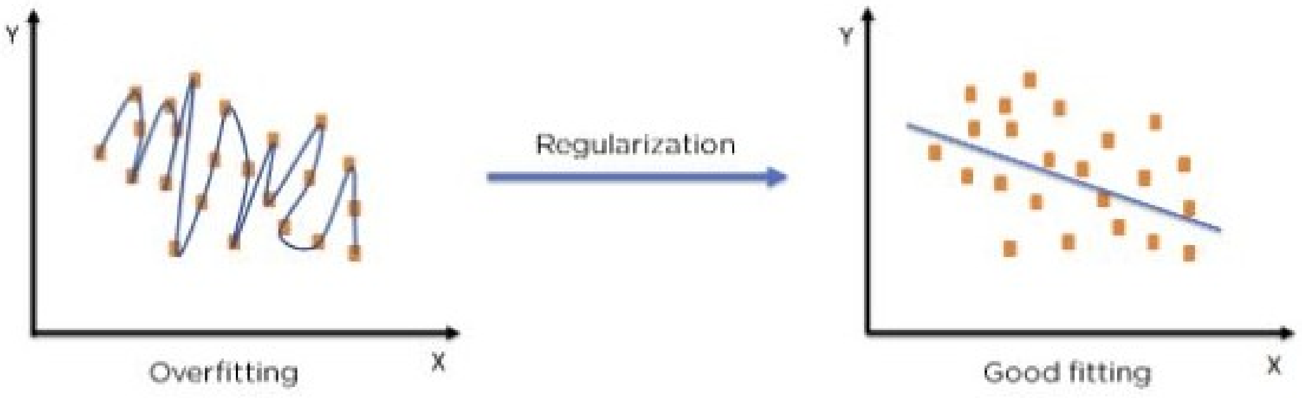
Regularization

Regularization



Regularization

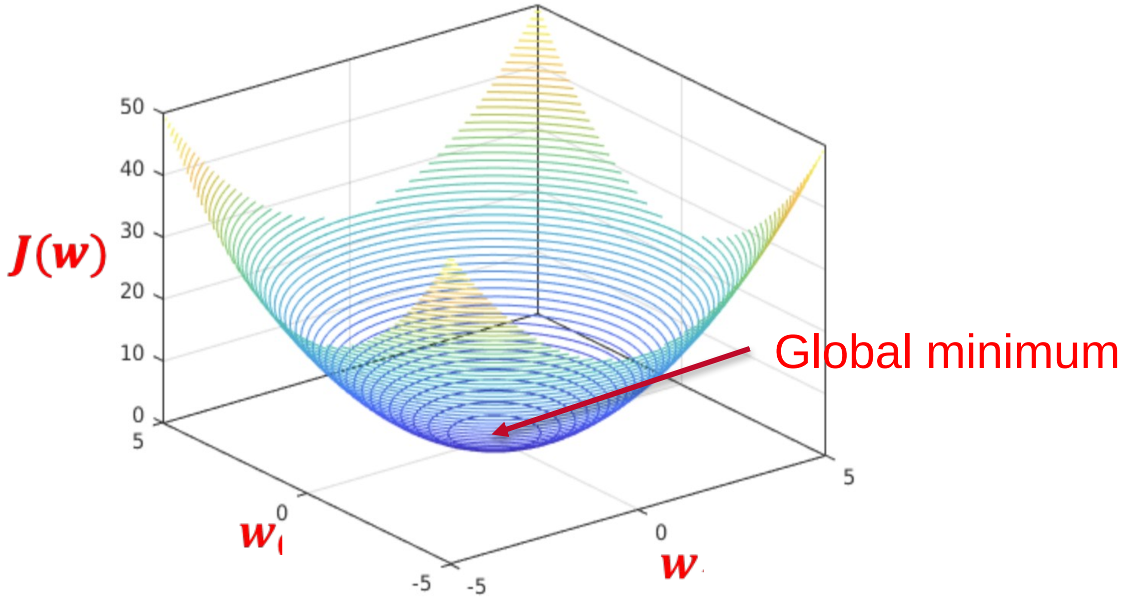
- Regularization is a set of techniques used in ML to improve a model's ability to generalize to new data.
  - Cost Function Penalties (linear models + gradient boosting)
  - Pruning (trees)
  - Dropout + Cost Function Penalties (neural networks)



Linear Regression Cost Function

- E.g., for linear regression, the cost function is convex:

$$\mathbf{J(w)} = (\mathbf{y} - \mathbf{Xw})^{\top}(\mathbf{y} - \mathbf{Xw}) = \|\mathbf{y} - \mathbf{Xw}\|_2^2$$

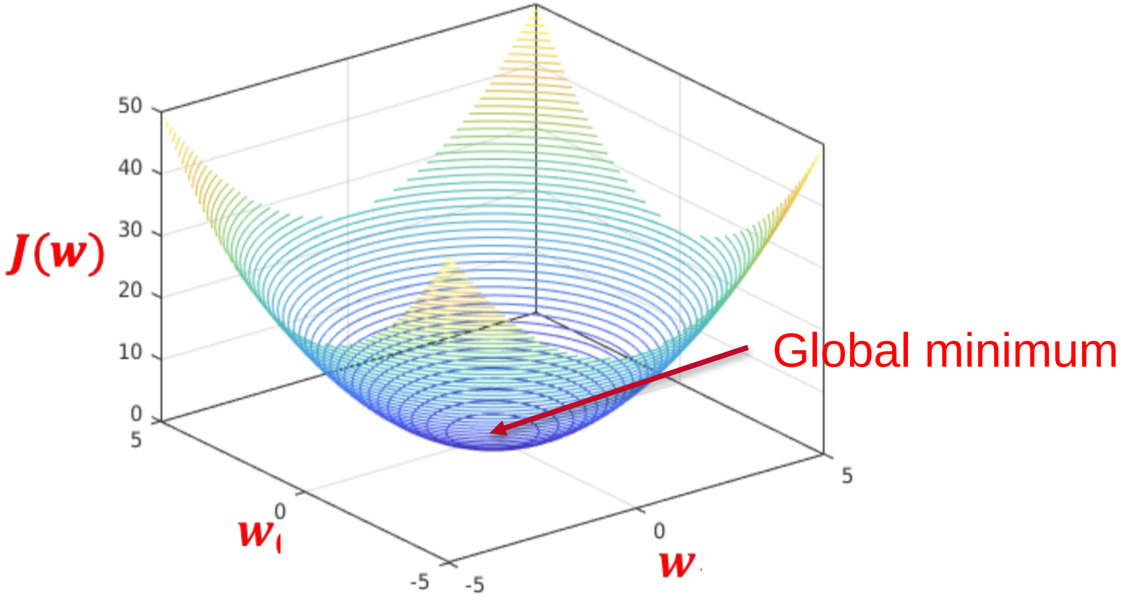


Linear Regression Cost Function

- E.g., for linear regression, the cost function is convex:

$$\mathbf{J(w)} = (\mathbf{y} - \mathbf{Xw})^\top(\mathbf{y} - \mathbf{Xw}) = \|\mathbf{y} - \mathbf{Xw}\|_2^2$$

$$\mathbf{X} = \begin{bmatrix} \boxed{1 \quad X_{11} \quad X_{12} \quad \cdots \quad X_{1k}} \\ 1 \quad X_{21} \quad X_{22} \quad \cdots \quad X_{2k} \\ \cdots \quad \cdots \quad \cdots \quad \cdots \quad \cdots \\ 1 \quad X_{n1} \quad X_{n2} \quad \cdots \quad X_{nk} \end{bmatrix} \qquad \boldsymbol{w} = \boxed{\begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_k \end{bmatrix}}$$



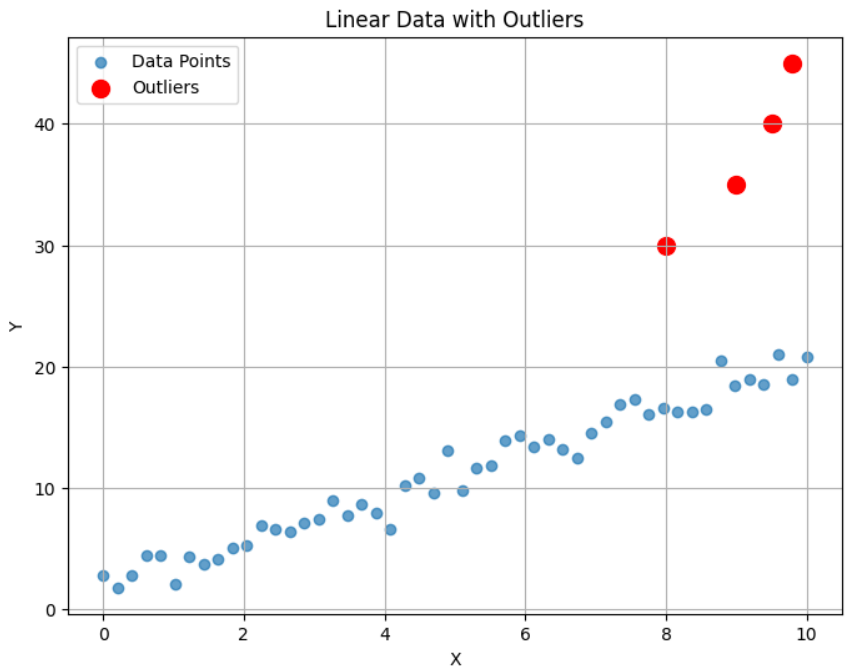
We focus on regularizing (controlling) the weights

Linear Regression Cost Function

- E.g., for linear regression, the cost function is convex:

$$\mathbf{J}(\mathbf{w}) = (\mathbf{y} - \mathbf{X}\mathbf{w})^{\top}(\mathbf{y} - \mathbf{X}\mathbf{w}) = \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2$$

$$X = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ 1 & x_3 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix} \quad \boldsymbol{w} = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix}$$



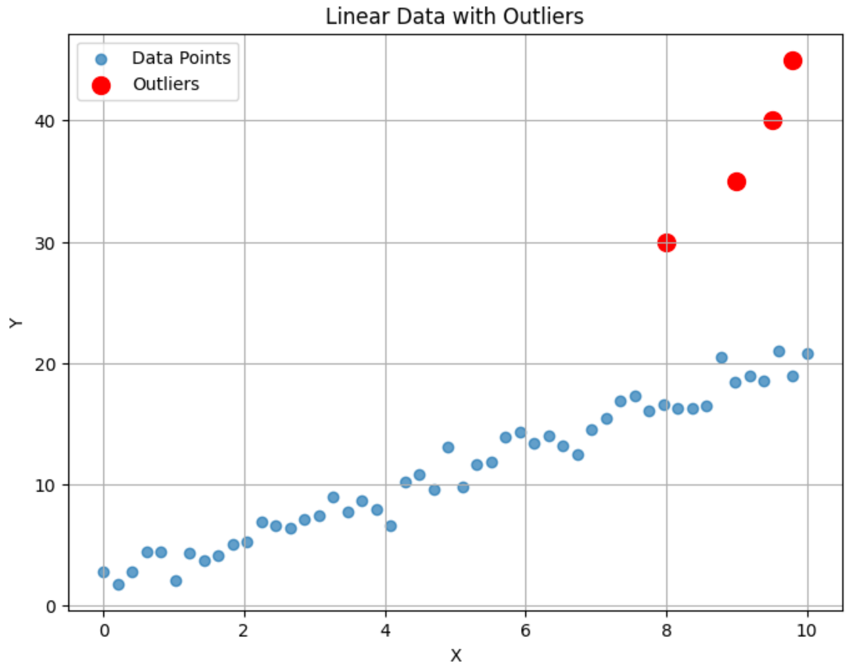
We focus on regularizing (controlling) the weights

Linear Regression Cost Function

- E.g., for linear regression, the cost function is convex:

$$\mathbf{J}(\mathbf{w}) = (\mathbf{y} - \mathbf{X}\mathbf{w})^{\top}(\mathbf{y} - \mathbf{X}\mathbf{w}) = \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2$$

$$\mathbf{X} = \begin{bmatrix} 1 & X_{11} & X_{12} & \cdots & X_{1k} \\ 1 & X_{21} & X_{22} & \cdots & X_{2k} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 1 & X_{n1} & X_{n2} & \cdots & X_{nk} \end{bmatrix} \qquad \boldsymbol{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_k \end{bmatrix}$$



We focus on regularizing (controlling) the weights

Linear Regression Cost Function

- E.g., for linear regression, the objective function is:

$$\min_{\boldsymbol{w}} \boldsymbol{J}(\boldsymbol{w})$$

- Substituting the cost function:

$$\min_{\boldsymbol{w}} ||\mathbf{y} - \mathbf{X}\mathbf{w}||_2^2$$

Ridge Regression - Adding an L_2 Penalty

- Ridge Regression uses a modification to the cost function
- The objective will become

$$\min_{\boldsymbol{w}} ||\mathbf{y} - \mathbf{X}\mathbf{w}||_2^2 + \underbrace{\alpha ||w||_2^2}_{\text{L2 penalty}}$$

- Forces the model to have parameters **closer to** zero
- α controls how much penalty. Can be any value $\alpha > 0$
- If $\alpha = 0$, then we are back to regular regression

Python

Ridge Regression with Cross-Validation

- The objective for Ridge Regression will become

$$\min_{\boldsymbol{w}} ||\mathbf{y} - \mathbf{X}\mathbf{w}||_2^2 + \boxed{\alpha ||w||_2^2}$$

L2 penalty

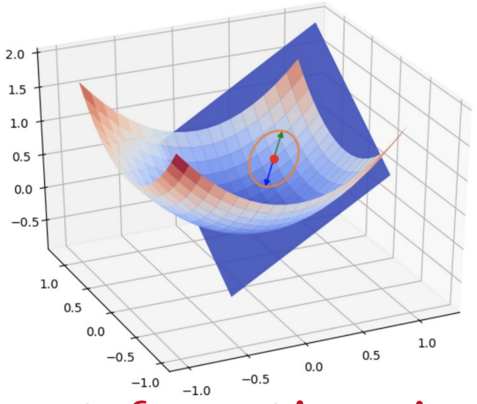
- Determining hyper-parameter $\boldsymbol{\alpha}$ is important
- We can use cross-validation to determine $\boldsymbol{\alpha}$

Python

Ridge Regression Gradient

- The gradient for Ridge Regression is:

$$\frac{\delta J(\mathbf{w})}{\delta \mathbf{w}} = \mathbf{X}^\top(\mathbf{X}\mathbf{w} - y) + 2\alpha\mathbf{w}$$



- The cost function is still convex (FAST COMPUTATIONALLY)
- The minimum can be found with the formula below:

$$\mathbf{w} = (\alpha I + \mathbf{X}^\top\mathbf{X})^{-1}(\mathbf{X}^\top y)$$

Lasso Regression - Adding an L_1 Penalty

- Lasso Regression uses a modification to the cost function
- The objective will become

$$\min_{w} \frac{1}{2n} ||\mathbf{y} - \mathbf{X}\mathbf{w}||_2^2 + \boxed{\alpha ||w||_1}$$

L1 penalty

- Forces the model to have parameters **equal to** zero if not essential
- α controls how much penalty. Can be any value $\alpha > 0$
- If $\alpha = 0$, then we are back to regular regression

Python

Lasso Regression with Cross-Validation

- The objective for Ridge Regression will become

$$\min_{w} \frac{1}{2n} ||\mathbf{y} - \mathbf{X}\mathbf{w}||_2^2 + \boxed{\alpha ||w||_1}$$

L1 penalty

- Determining hyper-parameter α is important
- We can use cross-validation to determine α

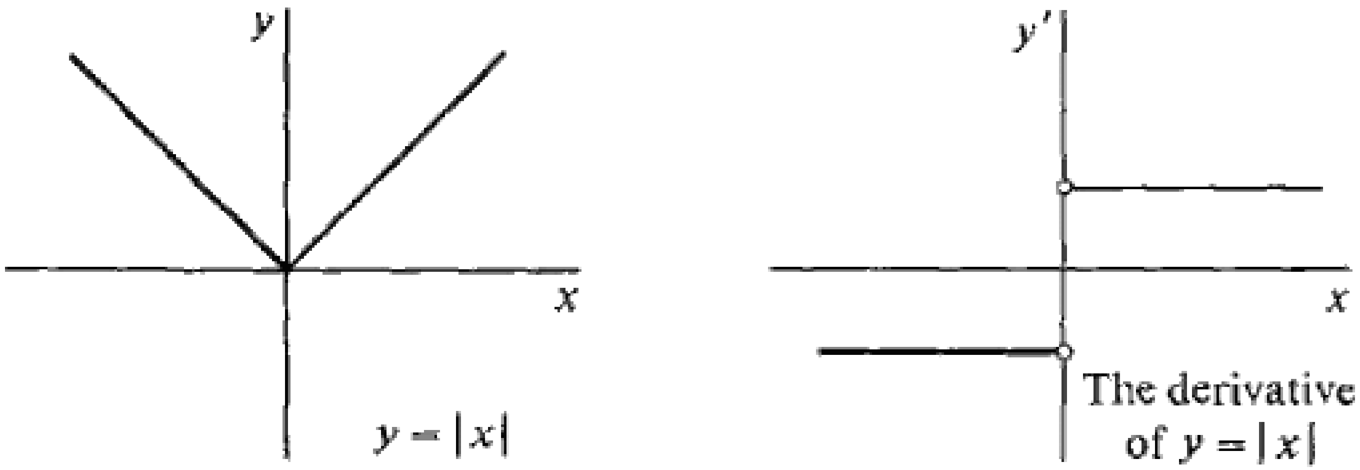
Python

Lasso Regression with Cross-Validation

- The objective for Ridge Regression will become

$$\min_{\boldsymbol{w}} \frac{1}{2n} ||\mathbf{y} - \mathbf{X}\mathbf{w}||_2^2 + \alpha||w||_1$$

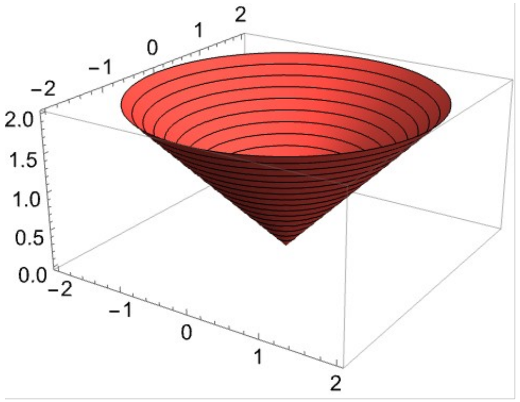
L1 penalty



Lasso Regression Gradient

- The gradient for Ridge Regression does not exist at zero:

$$\frac{\delta J(\mathbf{w})}{\delta \mathbf{w}} = \mathbf{X}^\top(\mathbf{X}\mathbf{w} - y) + \alpha \,\mathrm{sign}(w)$$



- The cost function is not strictly convex
- We use coordinate descent and optimization to obtain minimum

ElasticNet Regression - L_1 and L_2 Penalty

- ElasticNet Regression uses a modification to the cost function
- The objective will become

$$\min_{w} \frac{1}{2n} ||\mathbf{y} - \mathbf{X}\mathbf{w}||_2^2 + \boxed{\alpha \, \mathrm{L_1Ratio} \, ||w||_1 + 0.5\alpha \, (1 - \mathrm{L_1Ratio})||w||_2^2}$$

L1 and L2 penalties

- Forces the model to have parameters **equal to** zero (unimportant)
- Forces the model to have parameters **closer to** zero (higher order)
- α controls how much penalty. Can be any value $\alpha > 0$
- If $\alpha = 0$, then we are back to regular regression

Python

Huber Regression

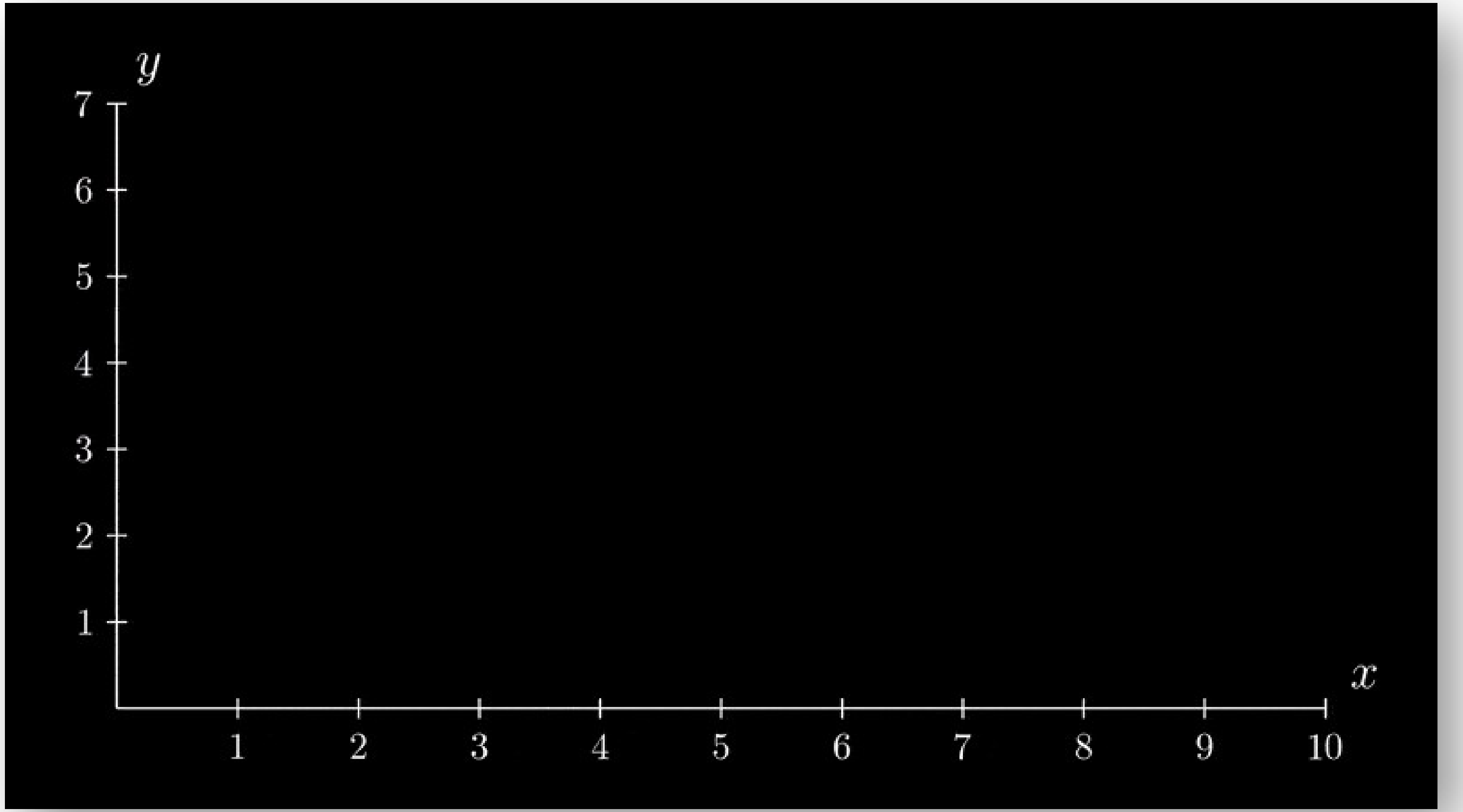
- Huber Regression uses a modification to the cost function
- The objective will become

$$\min_{\mathbf{w}} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 \text{ for observations with } \left|\frac{y-Xw}{\sigma}\right| < \epsilon$$

$$\min_{\mathbf{w}} |\mathbf{y} - \mathbf{X}\mathbf{w}| \text{ for observations with } \left|\frac{y-Xw}{\sigma}\right| < \epsilon$$

- Meant to be use for datasets with big outliers

Huber Regression



Python