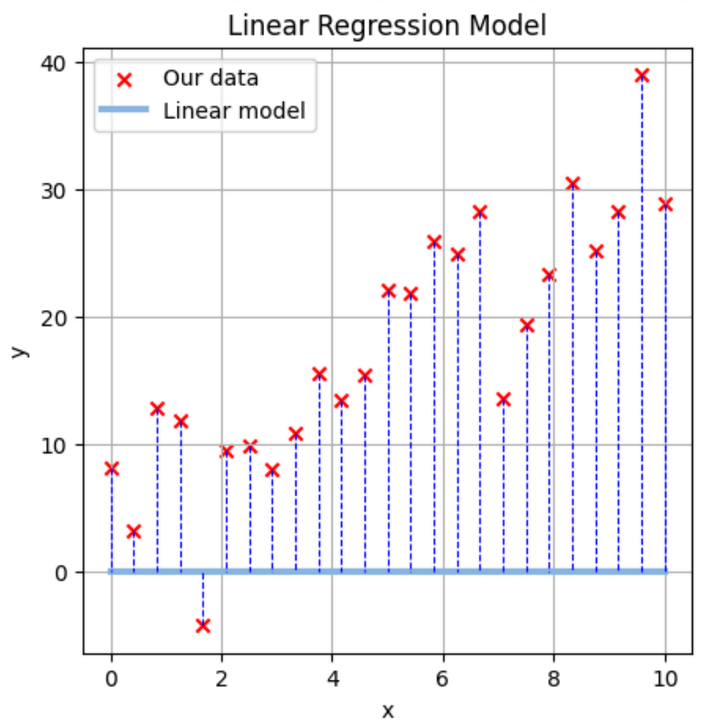
Regression Evaluation & Lasso, Ridge models

Presented by
FCAI-CU-AI-Community

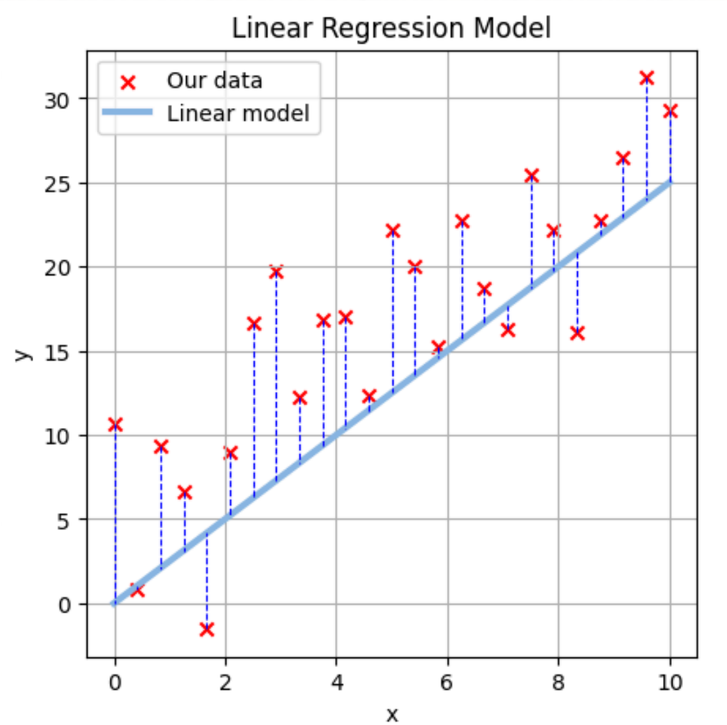
In Previous Session.

Linear Regression simply is just finding best line to fit our data

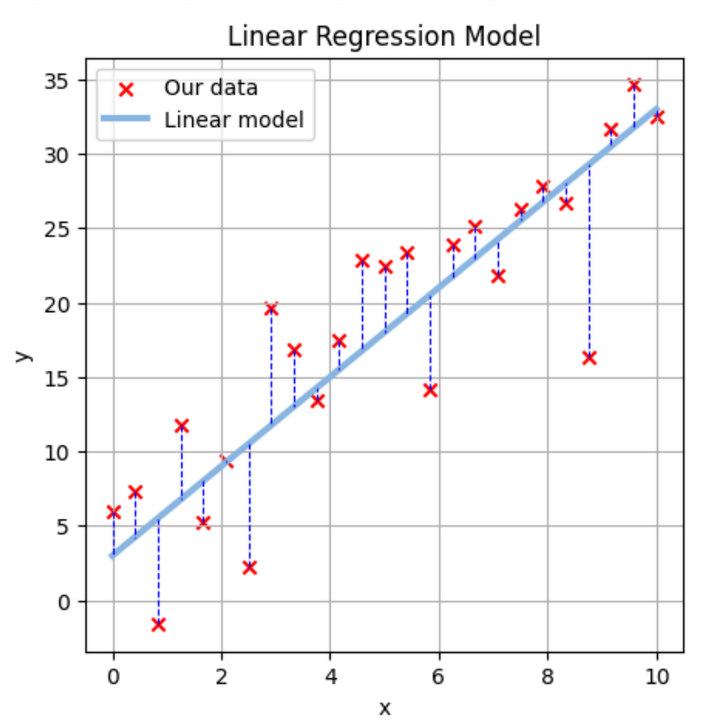
Is measuring distance between actual and predict can help us?



W = 0, b = 0



W = 2.5, b = 0



W = 3, b = 3

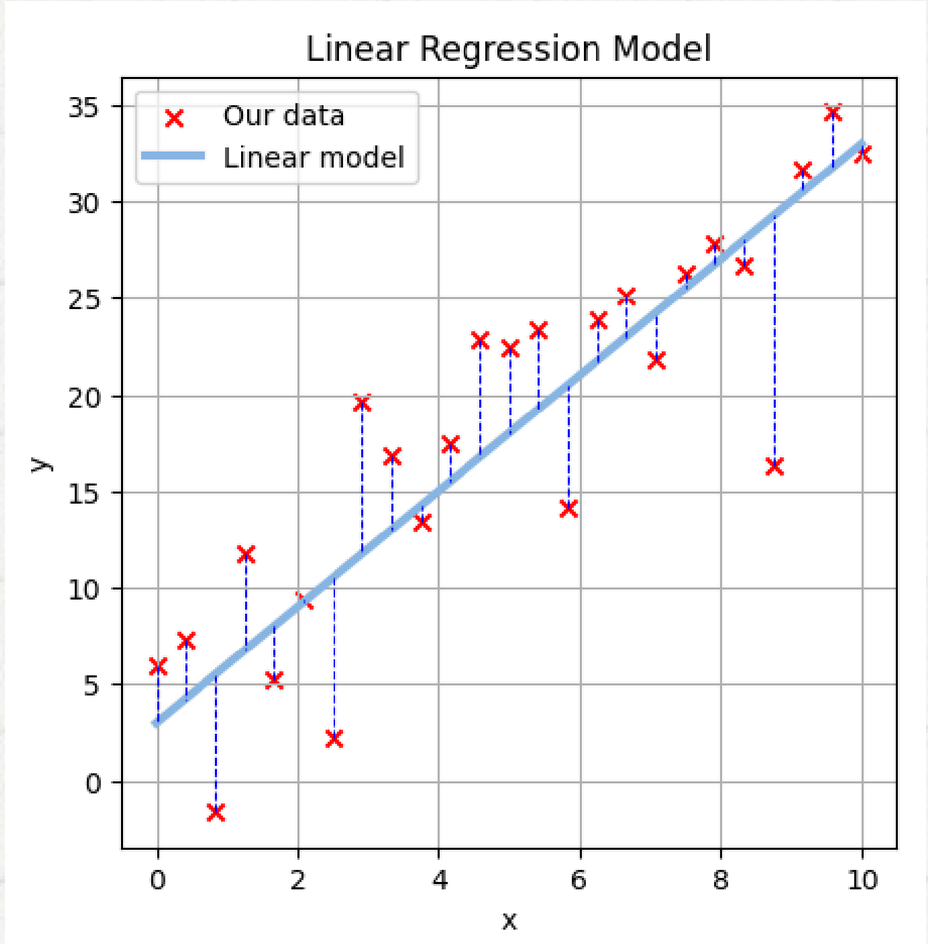
$$\hat{y} = W * X + b \quad y$$

$$distance = \hat{y} - y$$

$$distance = (\hat{y} - y)^2$$

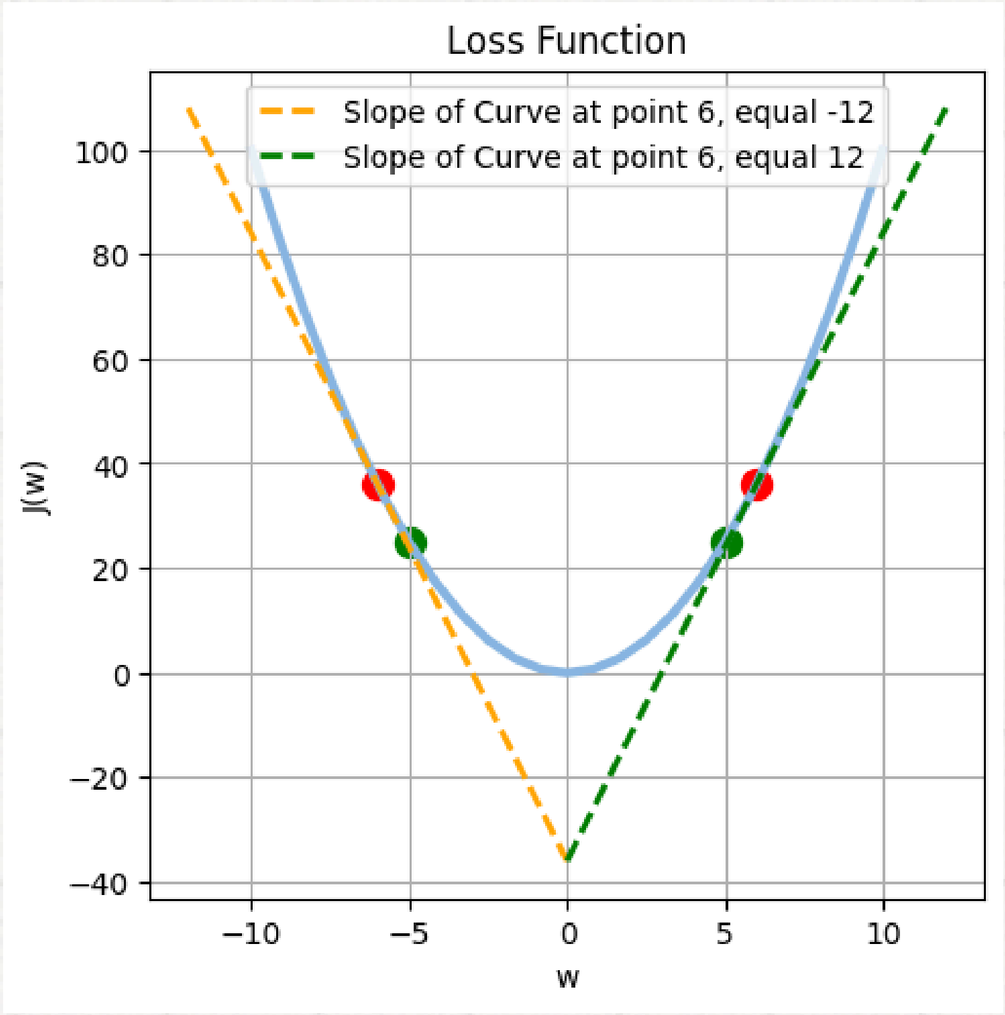
$$Total\,distance = \sum_{i=0}^{m} (\hat{y}_i - y_i)^2$$

$$MSE = \frac{1}{2m} \sum_{i=0}^{m} (\hat{y}_i - y_i)^2$$



$$\frac{dJ}{dw} = \frac{1}{m}\sum_{i=0}^{m}((w * x_i) - y_i) * x_i$$

$$w_{new} = w_{old} - \alpha * \frac{dJ}{dw}$$



Gradient descent steps:

1- Start with random w and b

2- compute cost $J(w,b) = \frac{1}{2m}\sum_{i=0}^{m}((w * x_i + b) - y_i)^2$

3- compute derivative of cost with respect to w and b

3.1- $\frac{dJ}{dw} = \frac{1}{m}\sum_{i=0}^{m}((w * x_i + b) - y_i) * x_i$

3.2- $\frac{dJ}{db} = \frac{1}{m}\sum_{i=0}^{m}((w * x_i + b) - y_i)$

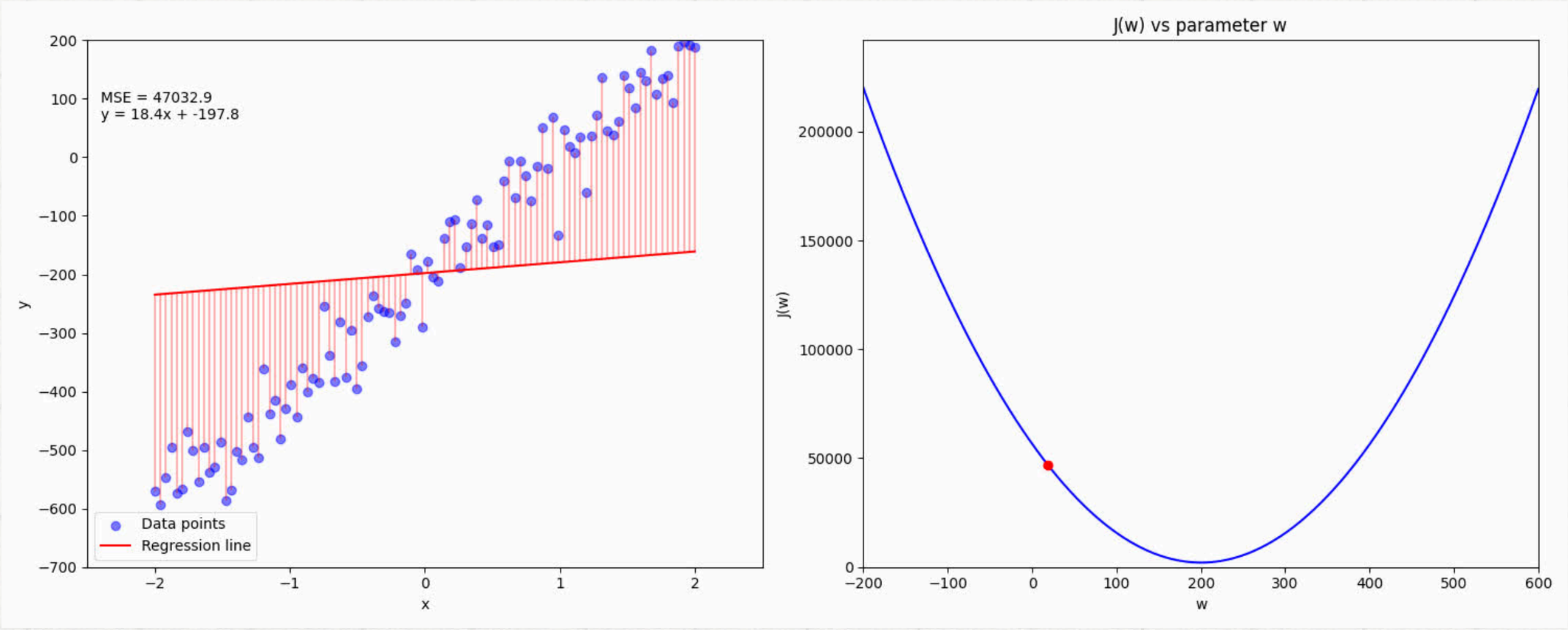
4- use this derivative to update weight and bias

4.1- $w_{new} = w_{old} - \alpha\frac{dJ}{dw}$

4.2- $b_{new} = b_{old} - \alpha\frac{dJ}{db}$

5- repeat 2,3,4 until converge or for some number of iterations

Gradient descent in action



After bulding our model we need to test it

There are a lot of metrics we can use to evaluate our model:

- MAE (Mean Absolute Error)
- MSE (Mean Square Error)
- RMSE (Root Mean Square Error)
- R2 (R-Squared value)

Advantages:

- same unit as the output variable
- Robust to outliers

Disadvantages:

- Larger errors don’t get penalized as heavily

$$MAE = \frac{1}{m}\sum_{i=1}^{m}\left|\hat{y}_i - y_i\right|$$

Advantages:

- Large errors get penalized as we are squaring

$$MSE = \frac{1}{m}\sum_{i=1}^{m}(\hat{y}_i - y_i)^2$$

Disadvantages:

- Sensitive to outliers
- value of MSE is a squared unit of output. Ex. the output variable is in meter(m) then after calculating MSE the output we get is in meter squared.

Advantages:

- Large errors get penalized as we are squaring
- same units as output variables

Disadvantages:

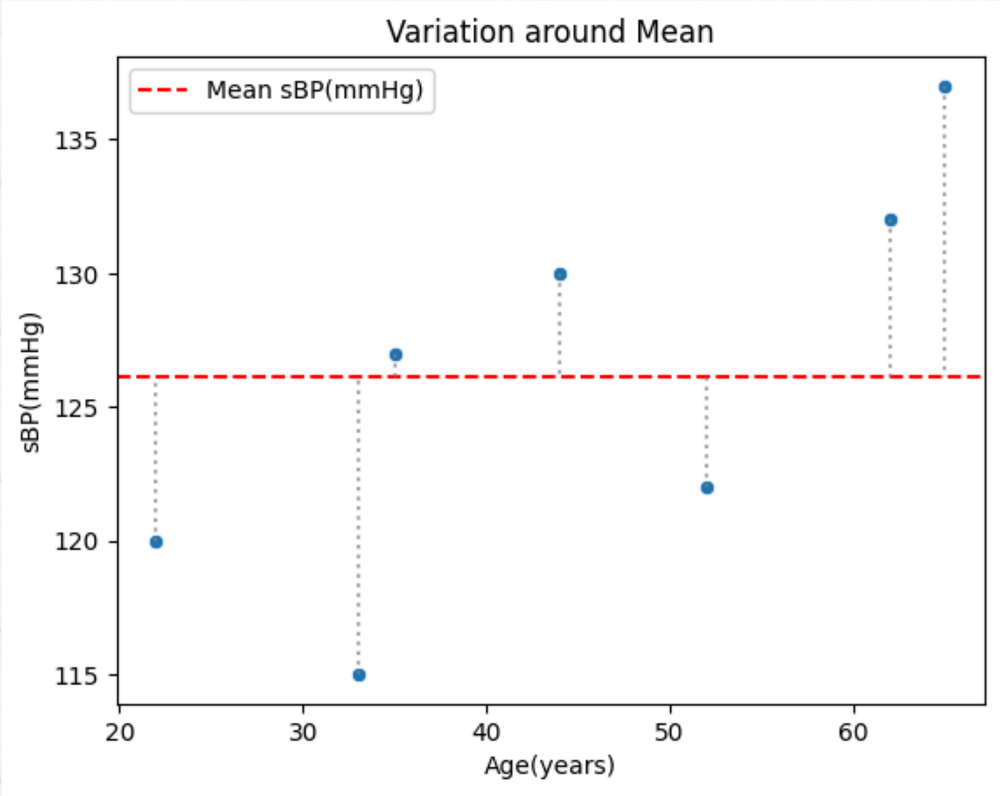
- Sensitive to outliers

$$RMSE = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(\hat{y}_i - y_i)^2}$$

Previous measures focused on loss but what if I want to see performance

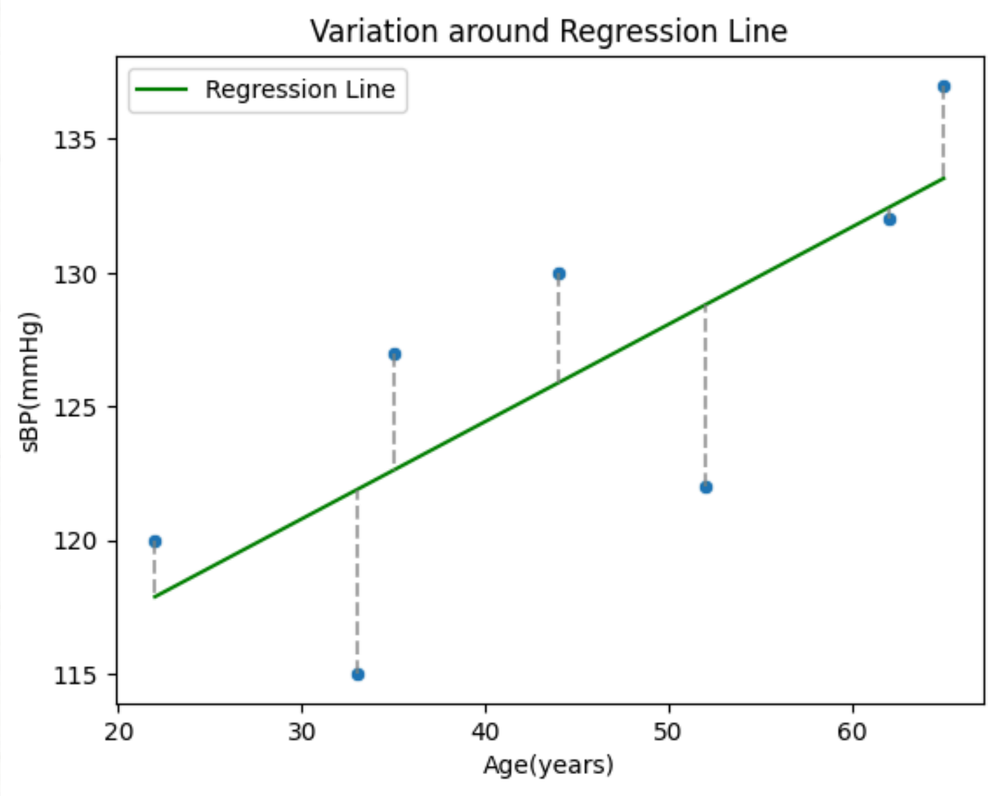
$$SST = \frac{1}{m}\sum_{i=1}^{m}(\bar{y}_i - y_i)^2$$

SST refers to sum of square total which gets error around mean line
In other words it is variance of data



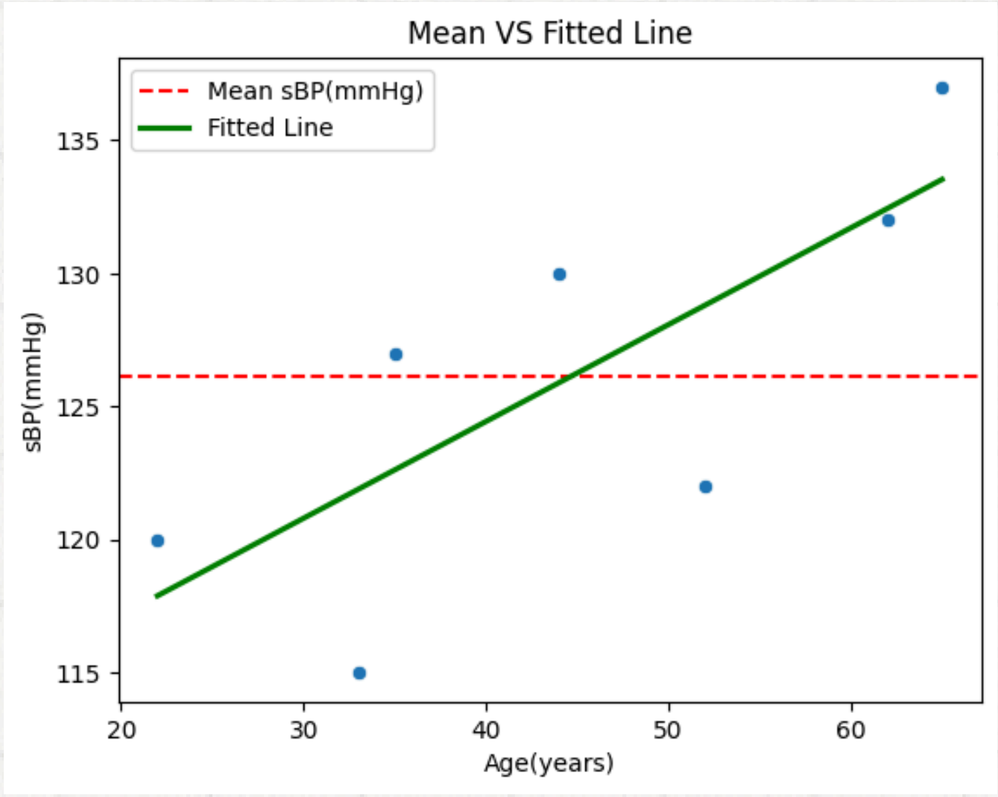
$$SSE = \frac{1}{m}\sum_{i=1}^{m}(\hat{y}_i - y_i)^2$$

SSE refers to sum of square estimated which calculate error of our regression model



$$R^2 = \frac{SST - SSE}{SST} = 1 - \frac{SSE}{SST}$$

R2 refers to R-squared it measures how our model performs better than base line (mean)
Its values range from 0 to 1 as we approach 1 our model performs good

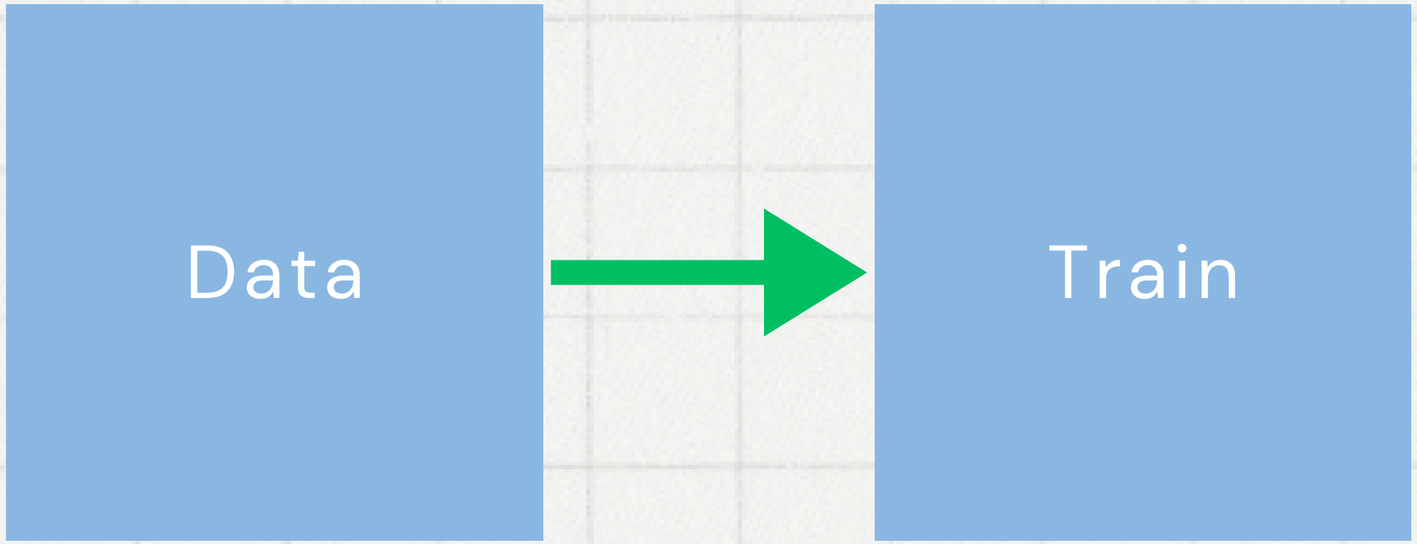


Summary

Metric	Range	When to use	Sensitive to Outliers
MAE	$[0, \infty[$	for models where robustness to outliers is important	NO
MSE	$[0, \infty[$	use for applications where large errors need to be penalized	YES
RMSE	$[0, \infty[$		YES
R2	$[0, 1]$	Use R^2 to assess the overall goodness-of-fit of the model	NO

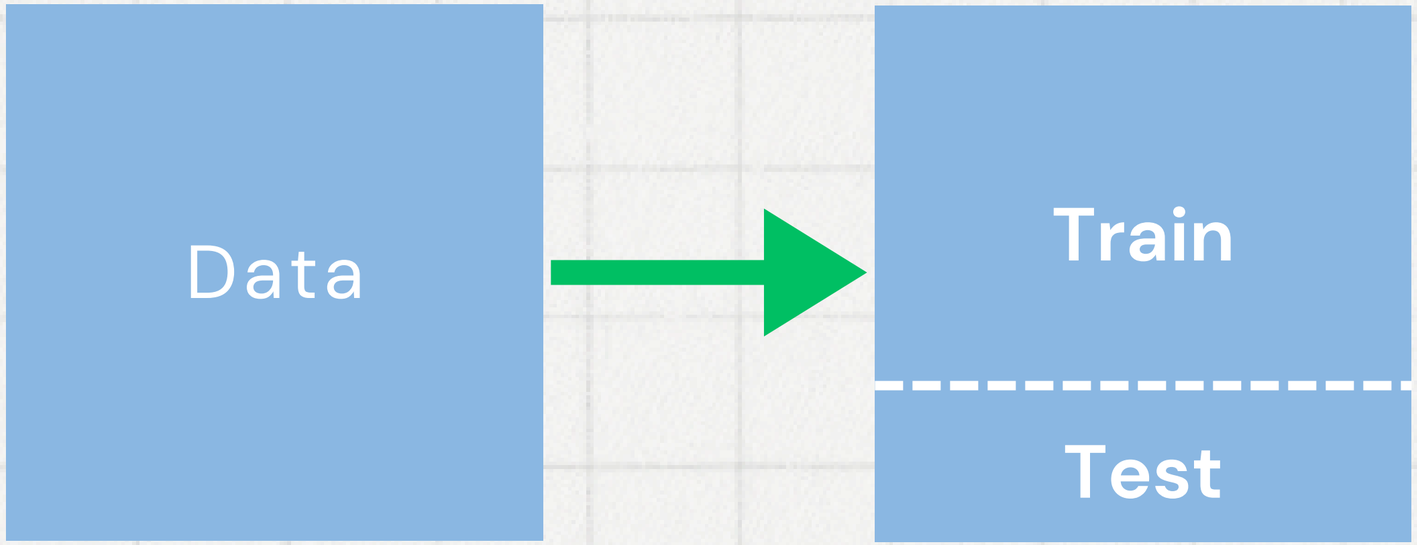
After Learning Evaluation lets see how we deal with our data to train and evaluate our model

Training our model on entire data



Is it a good choice to train model on entire data?

Training our model on training subset and evaluate it on test subset



This data subset is better than taking entire data for training, but there is a hidden problem

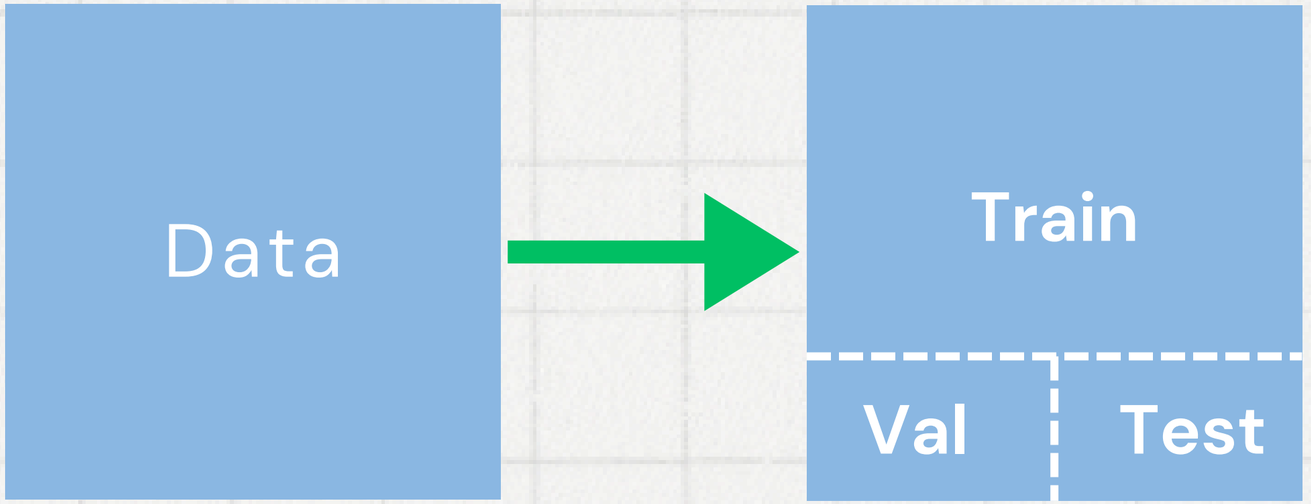
What is that problem?

Repetition of evaluation on same test data subset leads to that model will try to adapt for this test subset and it will not generalize

So, what is the solution?

Solution:

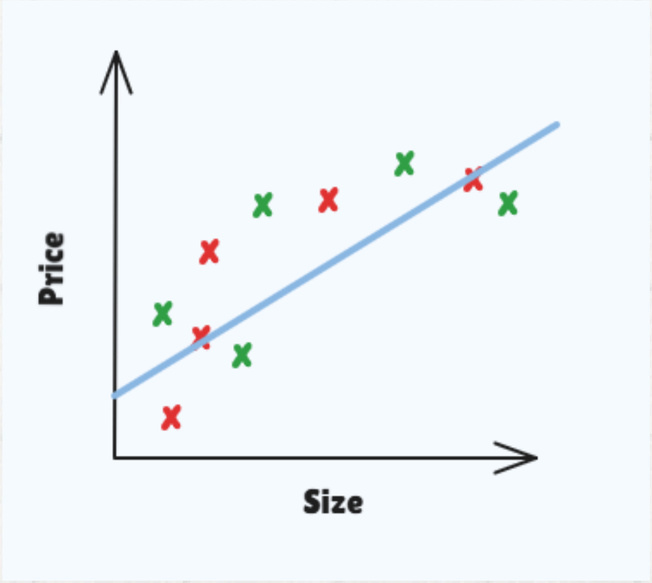
- **Train on train subset**
- **Evaluate on val. subset**
- **Test on test subset**



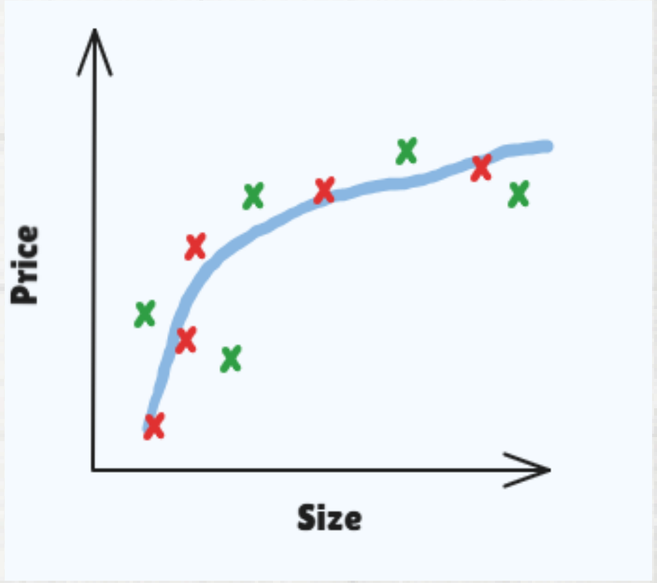
So, What is ratio of each subset

- usually, 80% train, 10% validation and 10% test
- But this ratio can change depends on problem and size of data

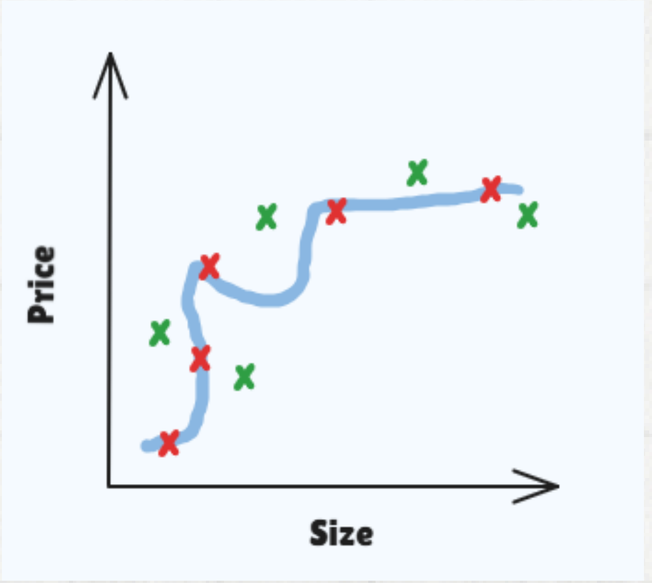
After knowing evaluation metrics and train, test split you will always meet on of following scenarios



Train loss High
Val loss High
“Underfit”



Train loss Low
Val loss Low
“Normal”



Train loss Very low
Val loss High
“Overfit”

In normal case how model take this shape:

$$\hat{y} = w_1 * x + b$$

this equation can’t fit data as data is not linear we need to introduce non-linear to our equation

$$\hat{y} = w_1 * x + w_2 * x^2 + b$$

We can square size so that we change our equation from linear to polynomial

Possible solutions for Overfitting:

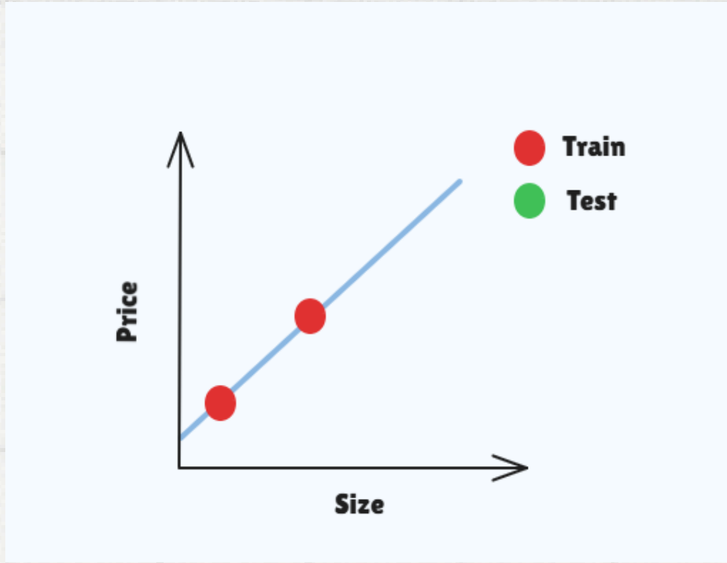
- Get more training data
- try smaller set of features
- Use Regularization $(Increase\ \lambda)$

Possible solutions for underfitting:

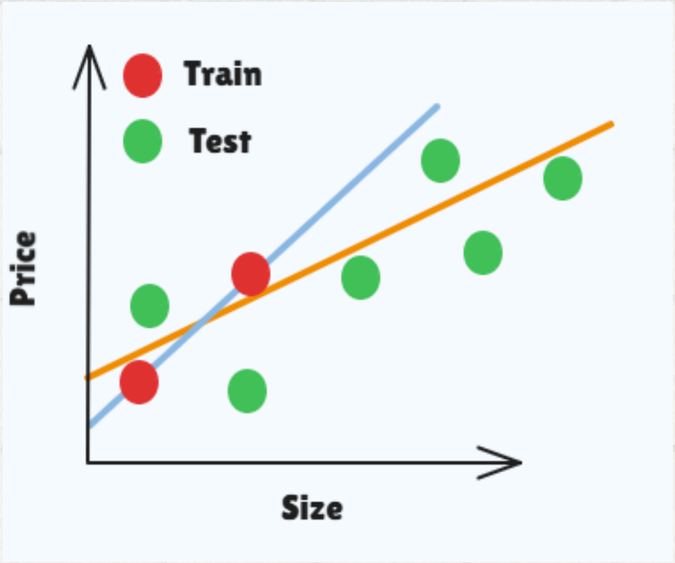
- try getting additional features
- try adding polynomial features
- Use Regularization ($Decrease\ \lambda$)

Overfitting in depth:

In figure (1):
train loss is approximately 0
but when we test our model
we can see in figure (2)
test loss is high and orange
line is what we expected from
training not blue line. we can
solve this by using Regularization



(1)



(2)

So, what is Regularization?

Regularization: is simply adding constrain to our optimization function

gradient descent in Linear Regression try to minimize:

$$\frac{1}{2m}\sum_{i=1}^{m}(y-(w_1x_1+\ldots+w_nx_m+b))^2$$

gradient descent in Ridge Regression try to minimize:

$$\frac{1}{2m}\sum_{i=1}^{m}(y-(w_1x_1+\ldots+w_nx_m+b))^2 \text{ with the constraint } \sum_{i=1}^{n}w_i^2 \leq t$$

gradient descent in Lasso Regression try to minimize:

$$\frac{1}{2m}\sum_{i=1}^{m}(y-(w_1x_1+\ldots+w_nx_m+b))^2 \text{ with the constraint } \sum_{i=1}^{n}|w_i| \leq t$$

Simple Example:

We want to minimize: $f(x, y) = x + y$ under the constraints $x + y = 1$

Geometric solution

Algebraic Solution:

By lagrange multiplier methods, minimizing

$f(x, y) = x + y$ under the constraints $x + y = 1$

is equivalent to

$$l(x, y, \lambda) = x^2 + y^2 + \lambda(x + y - 1)$$

Simple Example:

$$l(x, y, \lambda) = x^2 + y^2 + \lambda(x + y - 1)$$

$$\frac{dl}{dx} = 2x + \lambda = 0 \qquad \frac{dl}{dy} = 2y + \lambda = 0 \qquad \frac{dl}{d\lambda} = x + y - 1 = 0$$

$$2x = -\lambda$$
$$2y = -\lambda$$
$$x = y$$

$$y + y - 1 = 0$$
$$2y = 1$$

$$x = 0.5, y = 0.5$$

By lagrange multiplier methods, minimizing

$\frac{1}{2m}\sum_{i=1}^{m}(y-(w_1x_1+\ldots+w_nx_m+b))^2$ under the constraints

$$\sum_{i=1}^{n} w_i^2 \leq t \quad \sum_{i=1}^{n} |w_i| \leq t$$

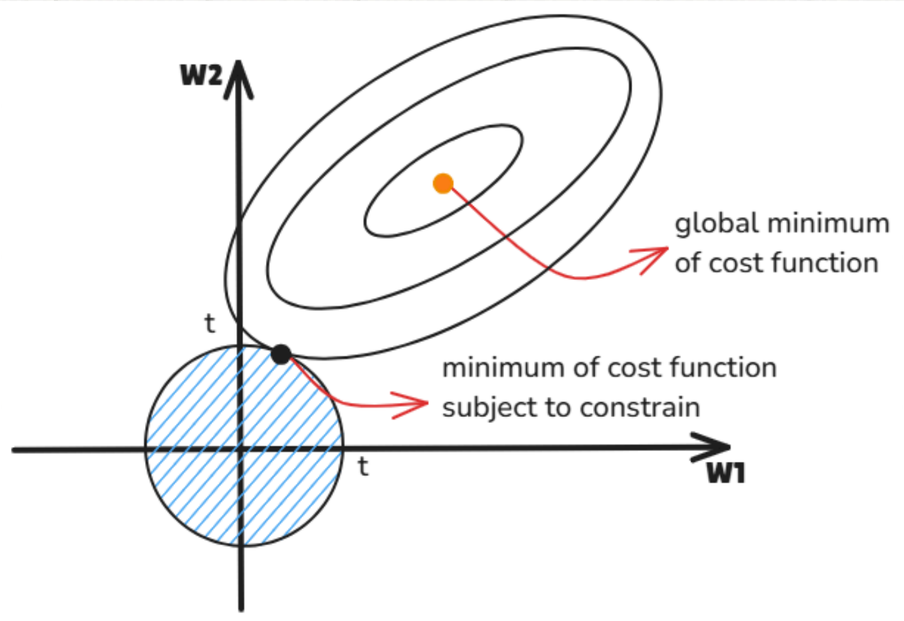
is equivalent to

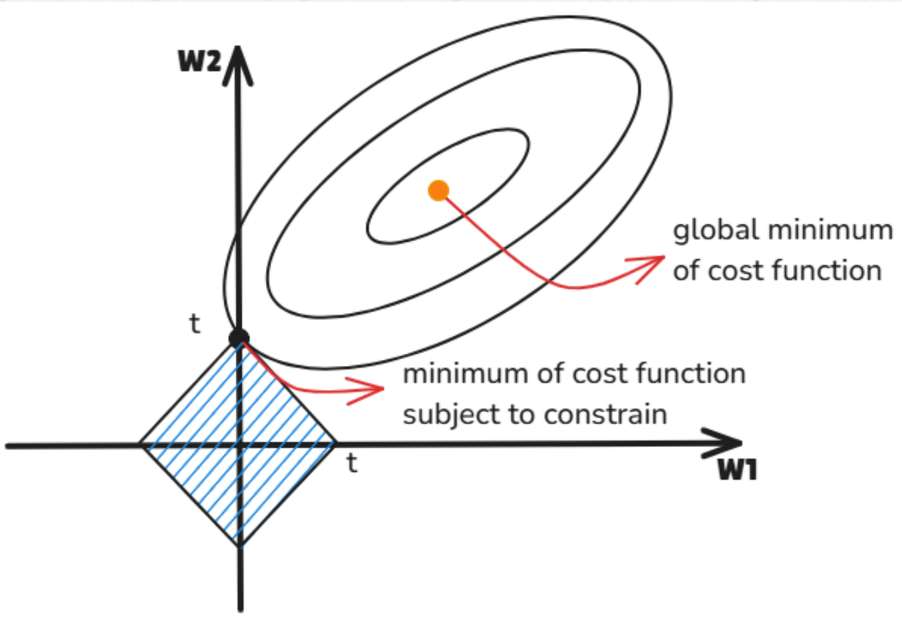
Ridge Regression, minimizing

$$\frac{1}{2m}\sum_{i=1}^{m}(y-(w_1x_1+\ldots+w_nx_m+b))^2+\lambda\sum_{i=1}^{n} w_i^2$$

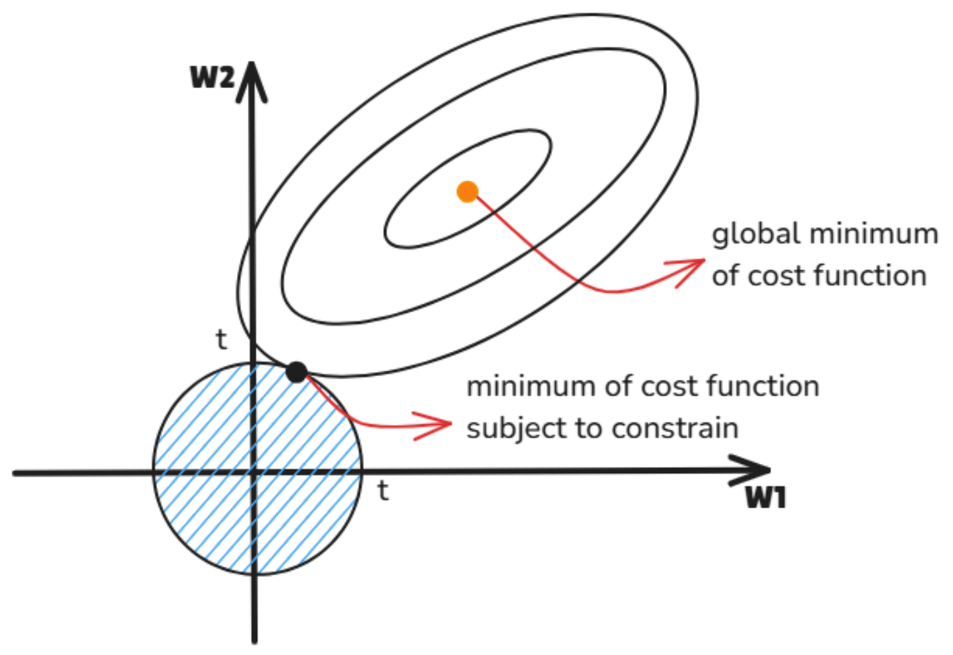
Lasso, minimizing

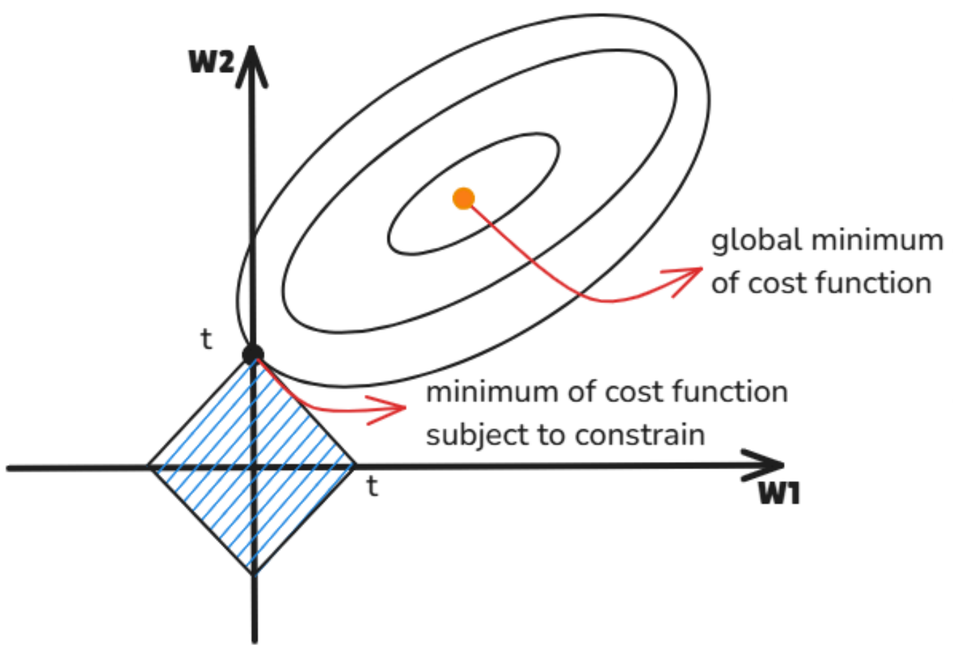
$$\frac{1}{2m}\sum_{i=1}^{m}(y-(w_1x_1+\ldots+w_nx_m+b))^2+\lambda\sum_{i=1}^{n} |w_i|$$





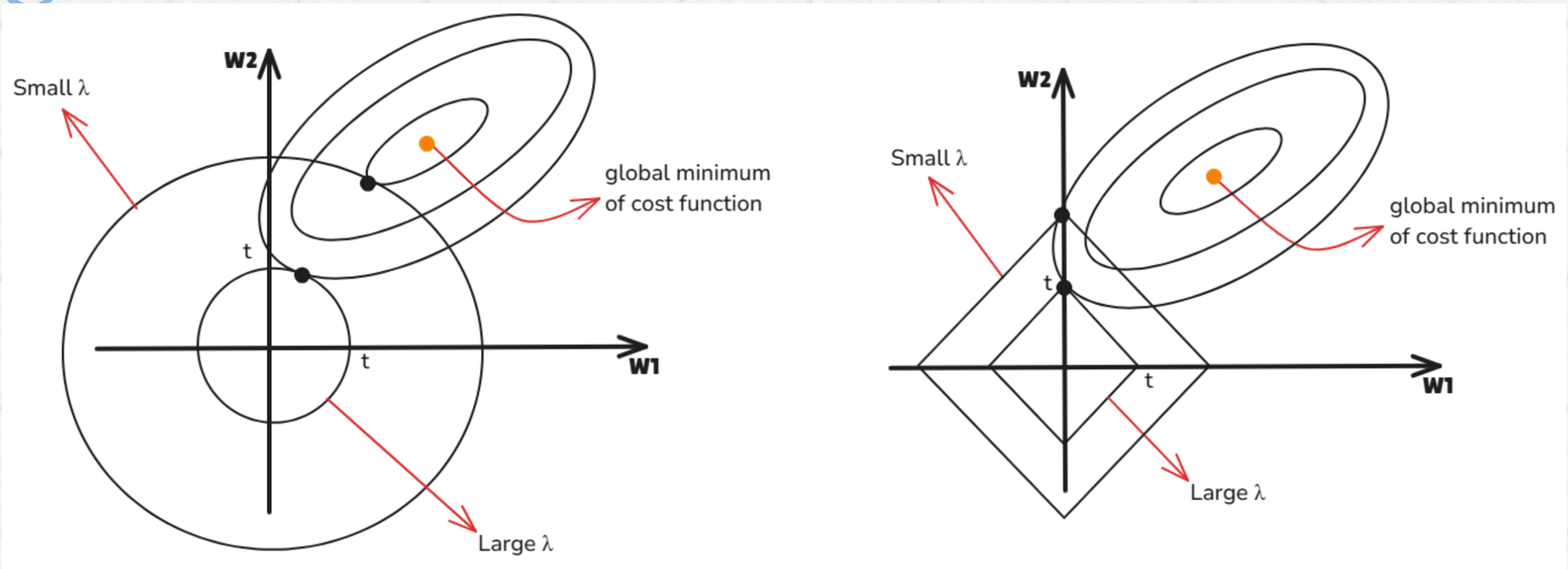
- Ellipses are the contours of $\frac{1}{2m}\sum_{i=1}^{m}(y_i + (w_1 x_i + w_2 x_i + b))^2$
- (Left) Ellipse intersects the circle of radius t at the Ridge estimate
- (Right) Ellipse intersects the square $(|w_1| + |w_2| < t)$ at the Lasso estimate





- we can notice that **Ridge** can shrink weights to approximately zero
- **Lasso** can shrink weights to zero so it can be used as feature selection

Last thing, how we can set λ?



- λ values ranges from 0 to ∞
- As λ ↓ 0 or t ↑ ∞, the Ridge and Lasso estimates become the ordinal MSE eq.
- As λ ↑ ∞ or t ↓ 0, the Ridge and Lasso estimates to zero (all weights become near zero in Ridge case and zero lasso case)

Thank you very much!