# Internals of GBDT

Pseudo Residual

Let's understand the **pseudo code**

Input: training set $\{(x_i, y_i)\}_{i=1}^{n}$, a differentiable loss function $L(y, F(x))$, number of iterations $M$.

Algorithm:

1. Initialize model with a constant value:

$$F_0(x) = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, \gamma).$$

2. For $m$ = 1 to $M$:
   1. Compute so-called *pseudo-residuals*:

$$r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)} \quad \text{for } i = 1, \ldots, n.$$

   2. Fit a base learner (or weak learner, e.g. tree) closed under scaling $h_m(x)$ to pseudo-residuals, i.e. train it using the training set $\{(x_i, r_{im})\}_{i=1}^{n}$ .
   3. Compute multiplier $\gamma_m$ by solving the following one-dimensional optimization problem:

$$\gamma_m = \arg\min_{\gamma} \sum_{i=1}^{n} L\left(y_i, F_{m-1}(x_i) + \gamma h_m(x_i)\right).$$

   4. Update the model:

$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x).$$

3. Output $F_M(x)$.

Let's look into **inputs** first

We have been provided with
- Training Data ( $D_{train}$ )
- Differentiable loss function ( L ( y , F(x) ) )
- Number of models ( trees )

Input: training set $\{(x_i, y_i)\}_{i=1}^{n}$, a differentiable loss function $L(y, F(x))$, number of iterations $M$.

**Training Data**

**( $D_{train}$ )**

**Loss Function**

**L ( y , F(x) )**

**# Trees we will build**

# STEP 1

Algorithms:

- Initialize model with a constant value :

$$F_0(x) = \underset{\gamma}{argmin} \sum_{i=1}^{n} L(y_i, \gamma)$$

We start by creating **STAGE 0** model i.e mean model.

- But that's not written in STEP - 1.

**What does STEP 1 mean ?**

$-For\ a\ given\ loss\ L(y^i, \gamma),$

$find\ \gamma\ which\ minimizes\ the\ loss\ L(y^i, \gamma)$

Which value of $\gamma$ minimizes square loss ?

To find minima, we take derivative of loss function

$$F_0(x) = \underset{\gamma}{argmin} \sum_{i=1}^{n} L(y_i, \gamma)$$

$$F_0(x) = \underset{\gamma}{argmin} \sum_{i=1}^{n} (y_i - \gamma)^2$$

$$\frac{\partial L}{\partial \gamma} = \frac{\partial \sum_{i=1}^{n} (y_i - \gamma)^2}{\partial \gamma}$$

$$\frac{\partial L}{\partial \gamma} = \sum_{i=1}^{n} -2(y_i - \gamma)$$

To find minima, equate it to 0

$$\frac{\partial L}{\partial \gamma} = \sum_{i=1}^{n} -2(y_i - \gamma)$$

$$\frac{\partial L}{\partial \gamma} = \sum_{i=1}^{n} (y_i - \gamma) = 0$$

Taking, $\gamma$ out of summation,

$$\frac{\partial L}{\partial \gamma} = \sum_{i=1}^{n} y_i - n\gamma = 0$$

$$\frac{\partial L}{\partial \gamma} = \sum_{i=1}^{n} y_i = n\gamma$$


$$\gamma = \frac{1}{n} \sum_{i=1}^{n} y_i$$

→ Mean of all $y_i$

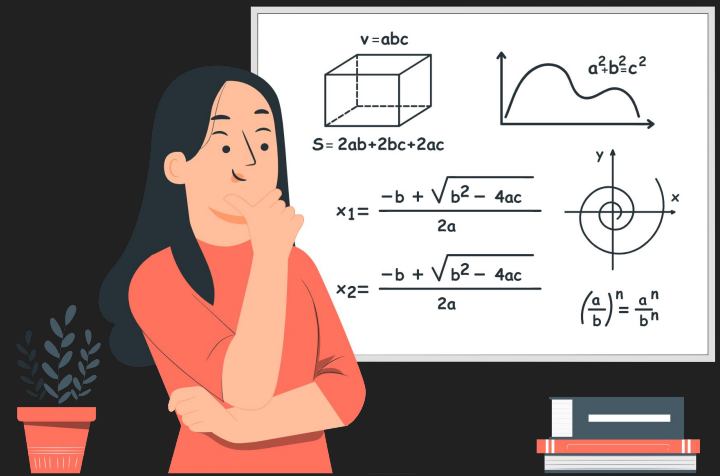
**Notice That:**

- $\gamma$ Which minimizes square loss is nothing but mean value
- The complex equation is simply telling us to create mean model

$$h_0(x) + \gamma_1 h_1(x) + \gamma_2 h_2(x)$$

**After finding value of $M_0$ (Stage 0)**

- We need to find $h_1(x)\ ,\ h_2(x) \ldots\ .\ h_m(x)$
- As well as weights $\gamma_1\ ,\ \gamma_2\ ,\ \gamma_3\ ,\ \ldots\ .\ \gamma_m$



# STEP 2

## Create M models ( Stage 1 to Stage M)

2. For $m$ = 1 to $M$:

    1. Compute so-called *pseudo-residuals*:

$$r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)} \quad \text{for } i = 1, \ldots, n.$$

    2. Fit a base learner (or weak learner, e.g. tree) closed under scaling $h_m(x)$ to pseudo-residuals, i.e. train it using the training set $\{(x_i, r_{im})\}_{i=1}^{n}$ .

    3. Compute multiplier $\gamma_m$ by solving the following one-dimensional optimization problem:

$$\gamma_m = \arg\min_{\gamma} \sum_{i=1}^{n} L\left(y_i, F_{m-1}(x_i) + \gamma h_m(x_i)\right).$$

    4. Update the model:

$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x).$$

# STEP 2 . 1

**In order to train new model**

- We calculate residual of previous model

2. For $m$ = 1 to $M$:

1. Compute so-called *pseudo-residuals*:

$$r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)} \quad \text{for } i = 1, \ldots, n.$$

Pseudo residual

Previous stage model

- **‘ i ’ represents datapoint**
- **M represents iteration**

Here , we are calculating pseudo residual $\frac{\partial L}{\partial F(x)}$ of previous stage model

- **Residual of each point $(\gamma_{im})$ is calculated.**

$$For\ m\ =\ 1$$

$$F(x) = F_{1-1}(x) = F_0(x)$$

# STEP 2 . 2

**After finding residual,**

- We fit the model using $( x_i , r_{im} )$
  - $r_{im}$ → Residual
  - $x_i$ → Features

2. Fit a base learner (or weak learner, e.g. tree) closed under scaling $h_m(x)$ to pseudo-residuals, i.e. train it using the training set $\{(x_i, r_{im})\}_{i=1}^{n}$

For m = 1

Fit model **M1** using $\{x_i, r_{i1}\}$

predicts

**$h_1(x)$**

$$h_0(x) + \boxed{V_1} h_1(x)$$

**After** learning h1 ( x ) - We need to find $\gamma$

## STEP 2 . 3

3. Compute multiplier $\gamma_m$ by solving the following one-dimensional optimization problem:

$$\gamma_m = \arg\min_{\gamma} \sum_{i=1}^{n} L\left(y_i, F_{m-1}(x_i) + \gamma h_m(x_i)\right).$$


**In simple terms, this equation is saying ;**

- To find $\gamma$ which minimizes given **loss**

$$L(y_i, F_{m-1}(x_i) + \gamma h_m(x_i))$$

*Actual Value* — $\hat{y}$ *(Predicted Value)*

To find

$$\hat{y} = F_{m-1}(x^i) + \gamma h_m(x^i)$$

$$For\ m = 1, \quad \hat{y} = F_{1-1}(x^i) + \gamma_1 h_1(x^i)$$

$$= F_0(x^i) + \gamma_1 h_1(x^i)$$

Stage 0 model (mean model)

Calculated is step 2.2

To find $\gamma$ which minimizes given **loss**

$$\frac{\partial L}{\partial \gamma_1} = \Rightarrow \quad = 0$$

- **Take derivative of loss w.r.t $\gamma$ & equate it to 0**

Only variable here is → $\gamma$

(Rest everything is constant - already calculated )

## STEP 2 . 4

$After\ finding\ h_1(x)\ and\ \gamma_1$

- We need to make final prediction

(i.e combine previous model with current model predictions)

4. Update the model:

$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x).$$

$$For\ m = 1$$

$$= \underbrace{F_0(x)} + \underbrace{\gamma_1 h_1(x)}$$

calculated in Step - 1

calculated in Step - 2.3

calculated in Step - 2.2

We keep doing sub-steps for M iteration i.e finding

- $h_1(x)\ , h_2(x)\ , h_3(x) \ldots h_m(x)$
- $\gamma_1\ , \gamma_2\ , \gamma_3 \cdots \gamma_m$

**Finally, we use these to make final model $F_M(x)$**

3. Output $F_M(x)$.

$$F_m(x) = h_0(x) + \gamma_1 h_1(x) + \gamma_2 h_2(x) + \ldots + \gamma_m h_m(x)$$

# Hyperparameter : # of base learners (M)

M ↑ ⟶ Overfit

As M increases , model will overfit .

**WHY ?**

- Because as base learners increase more likely training error will tend to 0

M ↓ ⟶ Underfit

As M decreases , model will underfit.

**WHY ?**

- Say M = 1 , Stage 0 & Stage 1 model.
- Prediction will be close to mean model - Underfit

# Hyperparameter

**Depth**

- As depth increases

→ Model will overfit

**Why ?**

- Increase in depth -> Variance Increase

→ Model will overfit quickly

$F_0(x) + \nu\left(\gamma_1 F_1(x) + \gamma_2 F_2(x)\right)$

$\nu$ → 0.001

# Regularization by Shrinkage

Final Model equation is : $F_m(x) = h_0(x) + \sum_{m=1}^{M} \gamma_m . h_m(x)$

To regularize, we add an regularization term i.e learning rate

$$F_m(x) = h_0(x) + \nu \sum_{m=1}^{M} \gamma_m . h_m(x)$$

$\nu$ : Learning Rate

$Range : 0 \leq \nu \leq 1$

Notes :

- Adding learning rate is reducing the impact of Mth .

  Hence, reducing overfit.

## Stochastic Gradient Boosting

**PROBLEM :** GBDT overfit a lot

( High Variance )

How to reduce variance ? - **Randomization** → **Row sampling + column sampling**

- Can use the some concept of randomization ( as used in RF ) to reduce variance

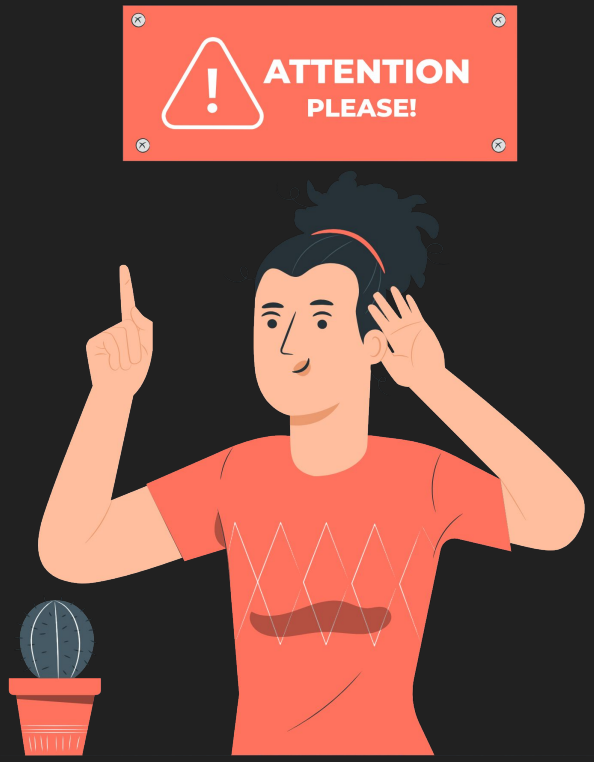
- This variation of GBDT is called ‘ **Stochastic Gradient Boosting’**

**GBDT → Pseudo Residual + Additive Combing**

**Stochastic GDBT → GBDT + Randomization**

Row Sampling + Column Sampling

- skLearn provides ability of row samping
  - Using **subsample** hyperparameter
  - And column sampling using **max_features** hyperparameter



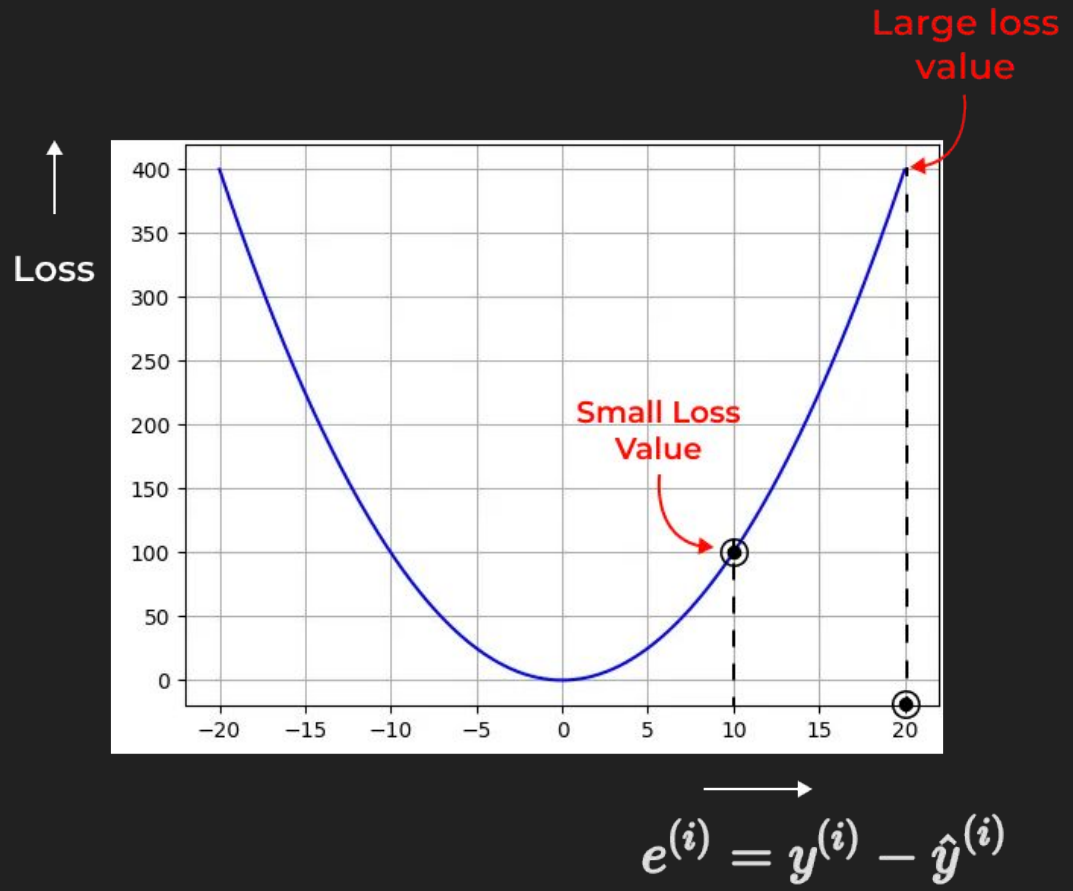

**Does outlier impact GBDT ?**

- As each model is fit on residual of previous model
- Outliers will have **high residual**

This causes GBDT to focus its attention on reducing these residual for outlier points.

**How to tackle this issue of OUTLIER ?**

- We can resolve this issue by changing loss function



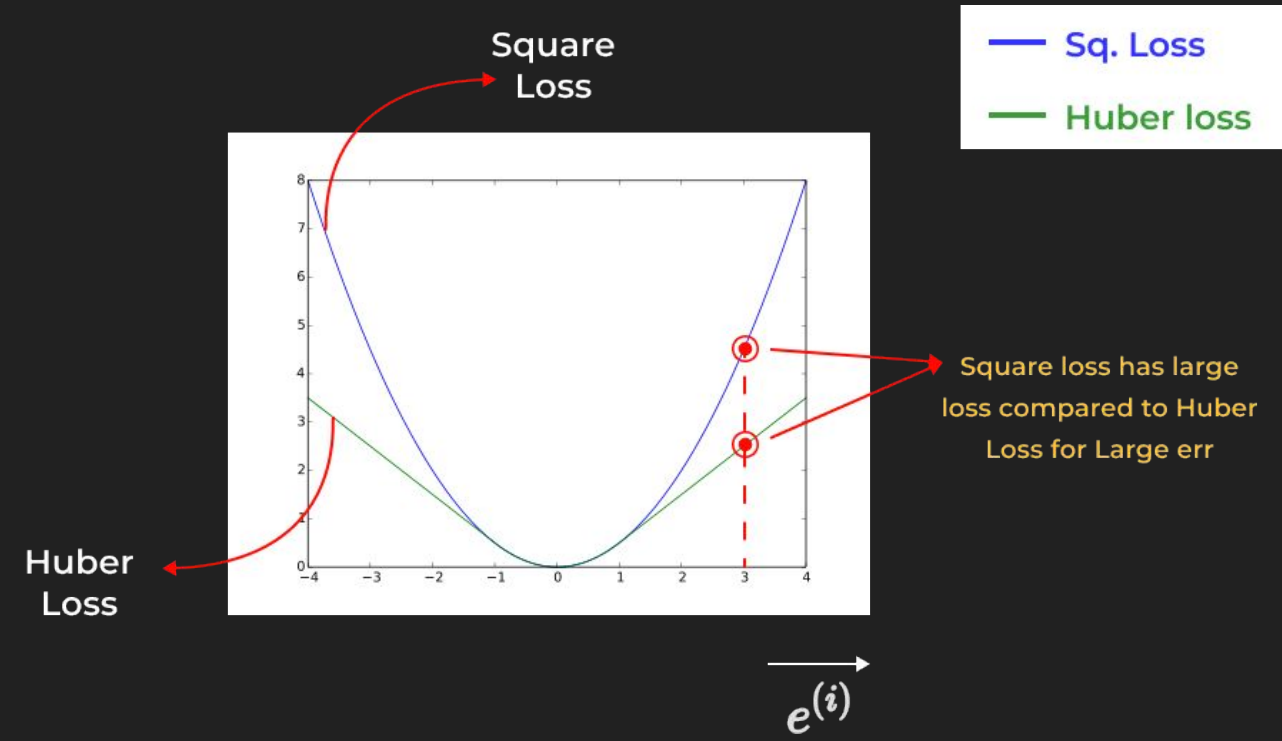

**When we use squared loss ;**

- The value of loss increases drastically if error value is large

Break : 8:15 am

**Instead of using squared loss ,**

- We can use **huber loss**




**Notice :**

- **For smaller residual values.**

  **Both squared loss & hyper loss have same values**

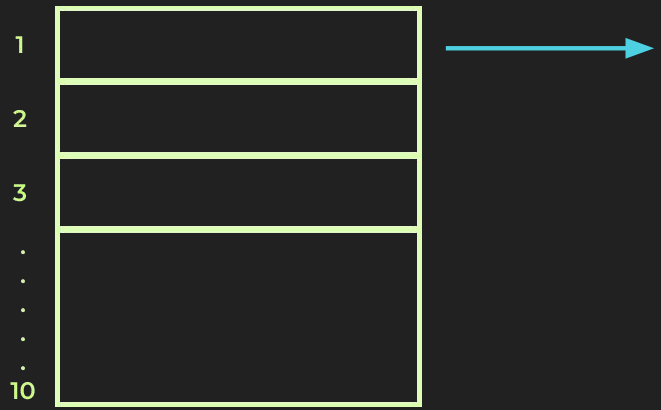
- **As value of residual increases**

  **Huber loss doesn't explode like squared loss**

## Summary of Pre-processing

- Chunking of **DATA**

- Data recorded at every 0.1 ms ] **Very small info**




- Take 10 continues interval

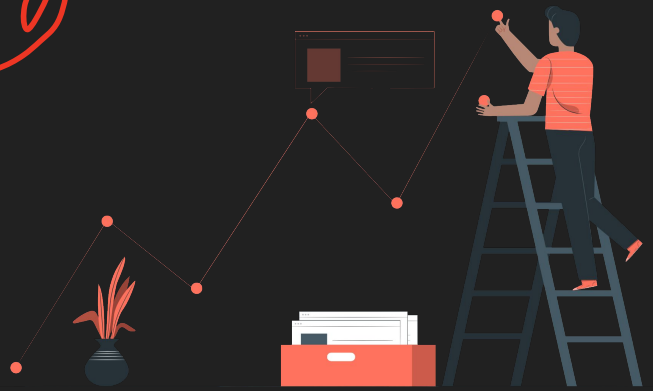
**replace it with mean/ median/ max**

Moving avg

- Target variable encoding

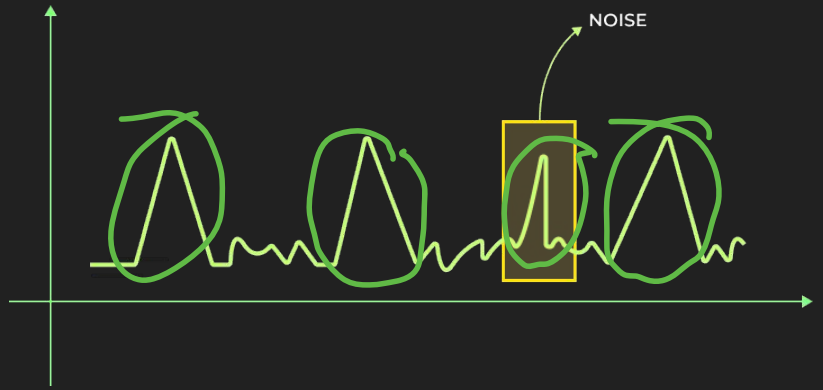
- Classes = 20 - Multiclass classification

  - Use label encoder



- Handling Noise ( noise due to equipment calibration etc. )

- Using moving average ( take average for an interval )

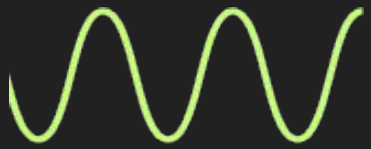


# Feature extraction

- Rectification ( **domain specific** )

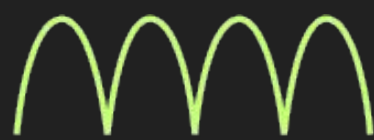
✓

Converts signal to positive value



Original signal

Half - wave Rectification



Full - wave Rectification

- Using full wave rectification to avoid data loss.