Gradient Boosting:-

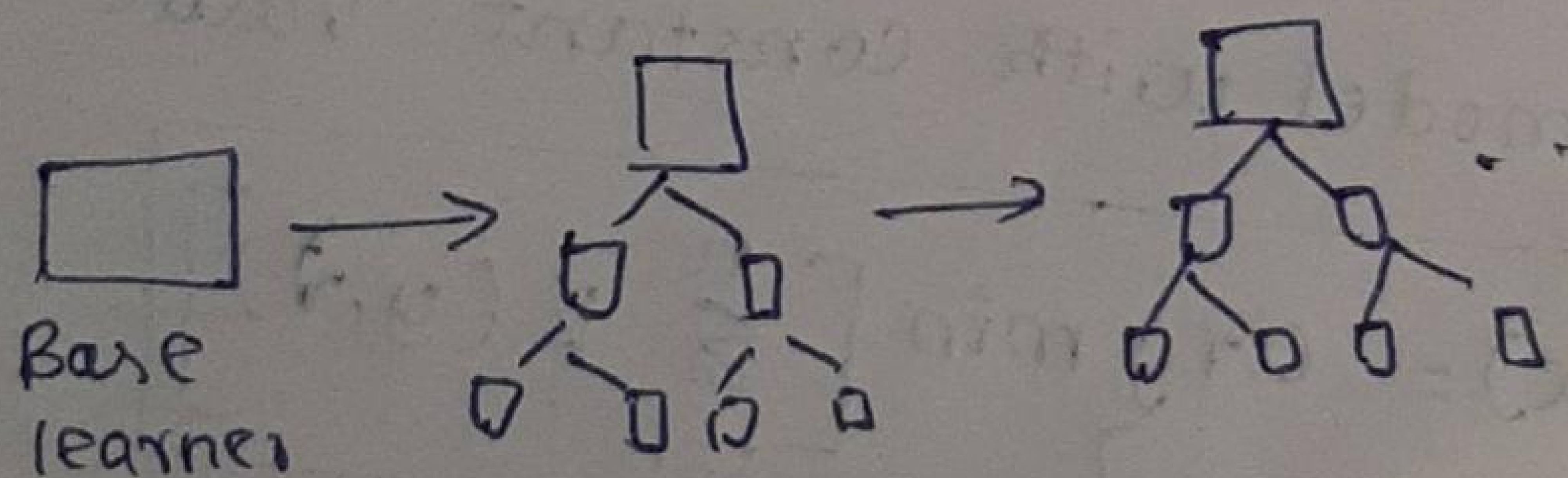
* It is the numerical optimisation problem where the objective is to minimise the loss function of the model by adding weak learners using gradient descent.

* Gradient descent is a first order iterative optimisation algorithm for ~~findin~~ finding local minimum of differentiable function.

* Gradient boosting is based on minimising a loss function.

* The basic idea is it will first set a basic learner and then the decision trees will get added as per the ~~key~~ value of hyperparameter.

* This is workedout by taking the previous model's residuals as the i/p to the present model.



* The main difference with adaboost is adaboost focuses on misclassified observation where as gradient boost trains learners based upon minimising the loss function of learners that is training ~~as~~ on the residuals of the model.

Scanned by TapScanner

Dataset :- ~~I/p~~ Indep O/P

Exp	degree	Salary (in K)
2	BE	50K
3	PHD	70 K
4	BE	60K

Input to gradient boosting :-

① $\{x_i, y_i\}$

$x_i \Rightarrow$ independent features

$y_i \Rightarrow$ dependent features

② $d(y, F(x))$

↳ Loss function like mse, Rmse.

(Loss function should be differentiable)

③ No. of trees needed.

Pseudo Algorithm :-

① Initialize model with constant Value

$$\boxed{F_0(x) = \arg\min_{\hat{y}} \left[\sum_{i=1}^{n} L(y, \hat{y}) \right]}$$

where $L(y, \hat{y})$ is Loss function

y is data pts

$\hat{y}$ is predicted values

Now, defining a loss function

$$\boxed{\underset{L(y,\hat{y})}{Loss} = \sum_{i=1}^{n} \frac{1}{2}(y-\hat{y})^2}$$

* we want to find $\hat{y}$ in such a way that the loss function should be reduced.

* Substituting values from dataset in loss function

$$= \frac{1}{2}(50-\hat{y})^2 + \frac{1}{2}(70-\hat{y})^2 + \frac{1}{2}(60-\hat{y})^2$$

* To find the $\hat{y}$ in order to minimise loss function we want to find first order derivative. ~~this step~~ actually this step uses gradient descent.

$$0 = \frac{\partial}{\partial \hat{y}}\left[\frac{1}{2}(50-\hat{y})^2 + \frac{1}{2}(70-\hat{y})^2 + \frac{1}{2}(60-\hat{y})^2\right]$$

critical Pt. so it is 0

$$0 = \frac{2}{2}(50-\hat{y})(-1) + \frac{2}{2}(70-\hat{y})(-1) + \frac{2}{2}(60-\hat{y})(-1)$$

$$0 = -50+\hat{y}-70+\hat{y}-60+\hat{y}$$

$$0 = -180+3\hat{y}$$

$$3\hat{y} = 180$$

$$\boxed{\hat{y} = 60}$$

* This is $\hat{y}$ for the Base learner.

$$\boxed{\text{Constant value} = 60}$$

updating dataset with $\hat{y}$

exp	degree	salary (in K)	$\hat{y}$
2	BE	50K	60
3	PHD	70K	60
4	BE	60K	60

② Iterate $m = 1$ to M (no. of trees)

i) compute pseudo residuals (Pseudo error)

$$r_{im} = -\left[\frac{\partial h(y, F(x_i))}{\partial F(x_i)}\right].$$

This is nothing but

$$\text{loss} = \frac{1}{2}(y-\hat{y})^2$$

$$= \frac{\partial}{\partial \hat{y}}\left(\frac{1}{2}(y-\hat{y})^2\right)$$

$$\frac{\partial h}{\partial \hat{y}} = \frac{\cancel{2}}{\cancel{2}}(y-\hat{y})(-1)$$

$$\frac{\partial h}{\partial \hat{y}} = -(y-\hat{y})$$

$$\boxed{-\frac{\partial h}{\partial \hat{y}} = y-\hat{y}}$$

So,

$$r_{im} = -\left[\frac{\partial h(y, F(x_i))}{\partial F(x_i)}\right] = -\frac{\partial h}{\partial \hat{y}} = y-\hat{y}$$

* Basically r_{im} is O/P feature − residual($\hat{y}$) of Base learner. (r_{im} means residual of ~~1st~~ Base model)

updating dataset with r_{im}

exp	degree	salary (in K)	$\hat{y}$	r_{im} $(y-\hat{y})$
2	BE	50K	60	-10
3	PHD	70K	60	10
4	BE	60K	60	0

ii) Fit a base learner $h_m(x)$ with $\{x_i, r_{im}\}$

(Training decision tree)

where x_i = Independent features

r_{im} (residuals) = dependent feature

* This is a decision tree regressor

iii) Finding ν_m in order to minimize the loss function

$$\boxed{\nu_m = \underset{\nu}{\operatorname{argmin}} \sum_{i=1}^{n} L(y_i, F_{m-1}(x_i) + \nu)}$$

This term clearly states its using output of previous iteration

* This ν_m same as $F_0(x)$.

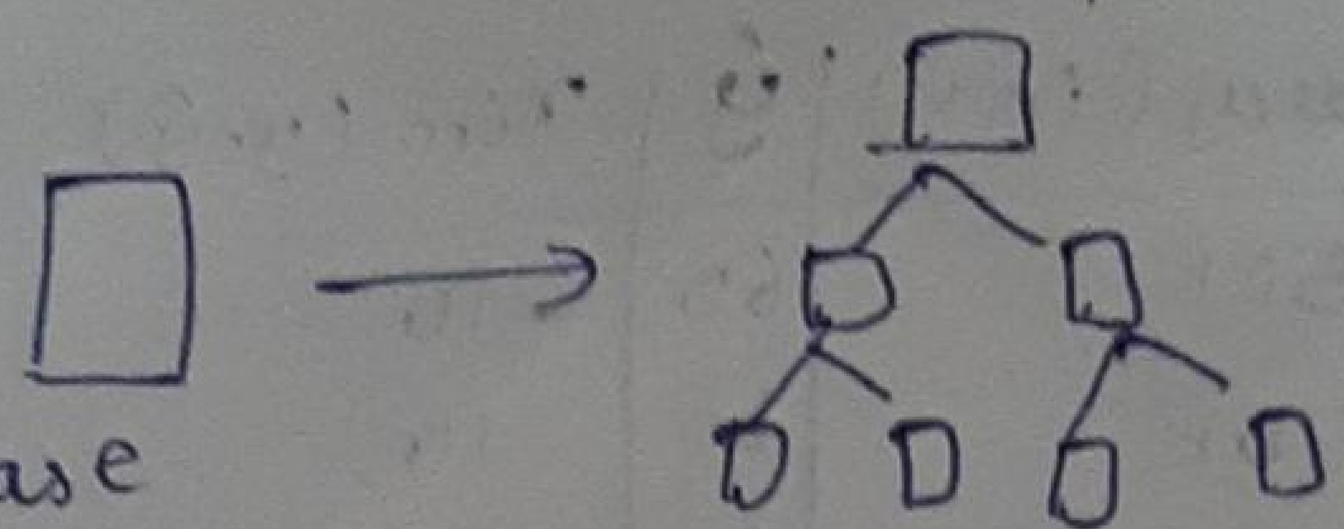
Previous model value added

$$\nu_m = \sum_{i=1}^{n} \frac{1}{2} (y_i - (60 + \hat{y}))^2$$

iv) update model

$$\boxed{F_m(x) = F_{m-1}(x) + \alpha(h(x))}$$

m = 1



Base learner

decision tree regressor

α → learning rate usually b/w 0 to 1

$$F_1(x) = F_{(1-1)}(x) + \alpha(h(x)) \rightarrow r_{11}$$

$$= F_0(x) + \alpha(h(x))$$

First tree (with only ~~one~~ record)

$$= 60 + 0.1(-10)$$

$$= 60 - 1.0$$

$$= 59.0$$

$y_1 = 50K$, $F_1(x) = 59$ } huge diff So, goes to next tree

* It will calculate for all the records.

* and if the diff is huge, it will go to next tree.

* usually decision trees leads to overfitting, so using less no. of trees will be ideal or else we can use regularization techniques to ~~reduce~~ generalize.