281

34.1 # Linear Regression.

Even though Logistic Regression has Regression term; its actually a classification technique.

Linear Regression → Actual Regression

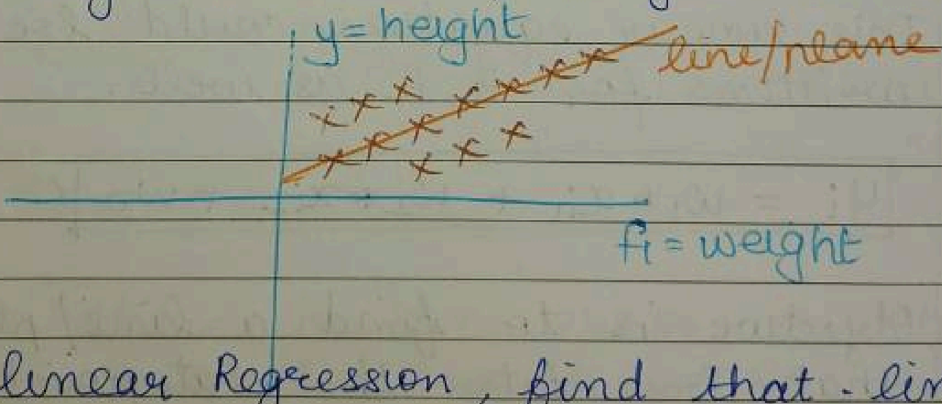
$D = \langle x_i, y_i \rangle_{i=1}^{n} \quad y_i \in \mathbb{R}$

Geometry behind Linear Reg

Eg: Predict the height of a person given weight, gender, ethinicity, hair Colour

Lets say I consider f_1 = weight and y ~~f_2~~ = height.

In general, lets say we observe that as height increases, weight also increases



In linear Regression, find that line ~~or~~ a plane that fits the given data

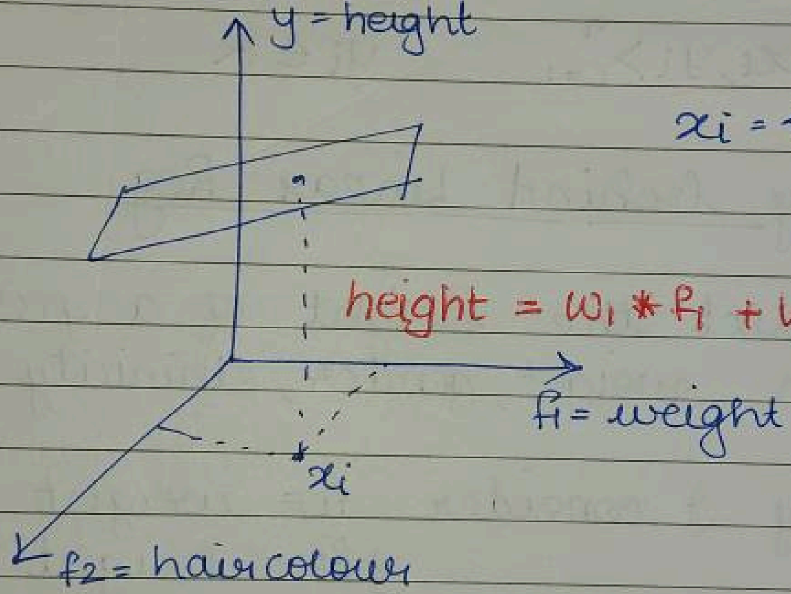
We can say that

$$\text{height} = (\underbrace{W_1}_{\text{slope}} * \text{weight}) + \underbrace{W_0}_{\text{y-intercept}}$$

So,

So, our whole objective is to find w_0 and w_1

Lets say my f_1 = weight and
f_2 = hair colour
y = height



$x_i = \langle x_{i1}, x_{i2} \rangle$
↓ wt, ↓ hair colour

height = $w_1 * f_1 + w_2 * f_2 + w_0$

$$y_i = w^T x_i + w_0$$

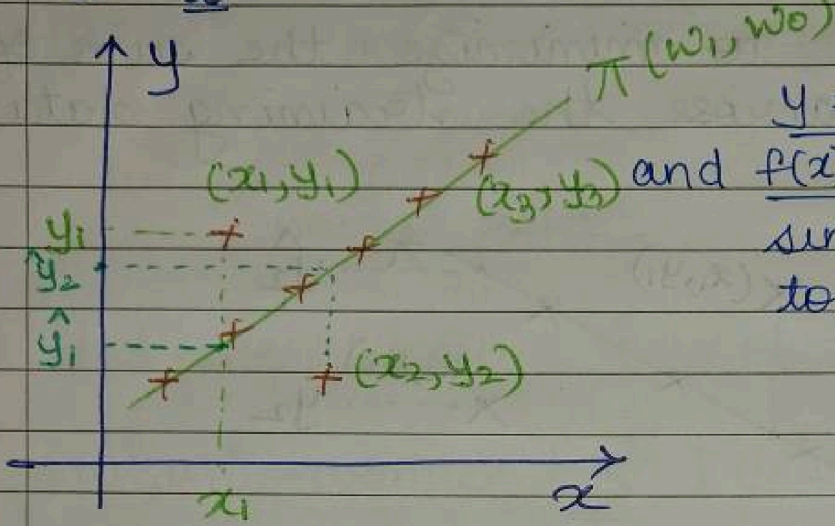
This linear equation could be written for f_1, f_2 as well.

$$\boxed{y_i = w_1 * x_{i1} + w_2 * x_{i2} + w_0}$$

Objective is to find a line/plane that best fits the datapoint

What does best fit means ?

Lets try to understand with 1 feature ~~x~~ 'x'



$y = f(x)$
and $f(x) = w_1 x + w_0$
since I want to fit a line

Consider the series of points as mentioned. The plane $\pi(w_1, w_0)$ best fits as it considers most of the points

$$f(x) = w_1 x + w_0$$

Now, consider the point (x_1, y_1) and apply $f(x)$ on that point.

$\therefore f(x_1) = \hat{y}_1 \neq y_1$ because (x_1, y_1) doesnt lie on the plane

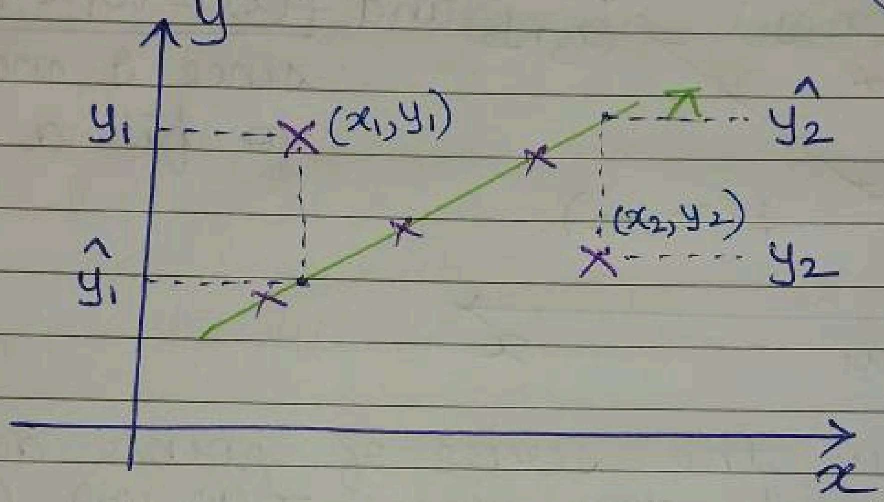
Illy $f(x_2) = \hat{y}_2 \neq y_2$

Hence, there is some error associated with those two points which are not exactly on the plane

Error in Model for
$x_1 = y_1 - \hat{y}_1$
$x_2 = y_2 - \hat{y}_2$
$x_3 = \underline{\underline{0}}$ because it lies on π

34.2 Mathematical Formulation

We want to minimize the sum of errors across the training data.



We want the best fit line that minimizes the sum of errors

For $(x_1) \longrightarrow$ error $= y_1 - \hat{y}_1 = +ve$

$(x_2) \longrightarrow$ error $= y_2 - \hat{y}_2 = -ve$

We want to minimize both +ve and -ve errors. hence, we can compute $(\text{error})^2$

~~Linear Regression:~~

$$\cancel{(w^*, w_0^*) = \underset{w^*, w_0}{\text{argMin}}}$$

Linear Regression

$$w^*, w_0^* = \underset{w, w_0}{\arg\min} \left[\sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \right]$$

w → Vector, w_0 → Scalar

And we know $\hat{y}_i = f(x_i) = w^T x_i + x_0$

$$\therefore w^*, w_0^* = \underset{w, w_0}{\arg\min} \left[\sum_{i=1}^{n} \underbrace{\{ y_i - (w^T x_i + x_0) \}^2}_{\longleftarrow \text{LOSS TERM} \longrightarrow} \right]$$

This is our optimization problem
The loss term is also called as
Square Loss.
Since we are computing the square
in a linear equation, the linear
Regression is also called as

→ Ordinary Least Square (OLS)
→ Linear Least Square (LLS)

We also apply Regularization here

$$(w^*, w_0^*) = \underset{w, w_0}{\arg\min} \sum_{i=1}^{n} \{ y_i - (w^T x_i + w_0) \}^2 + \underbrace{\lambda \|w\|_2^2}_{\text{L2-Regularizaⁿ}}$$

Similar to Logistic Regression
we can apply

L1 Regularizaⁿ OR
L2 — " — OR
Elastic Net

Probabilistic Interpretation

$P(y_i|x_i) = N(\mu, \sigma^2)$

Just like Logistic Regression

Linear Regression can be explained in

→ Geometric Interpretation
→ Probabilistic —"—
→ Loss function (minimization)

Loss Function

We saw 0-1 Loss and Logistic Loss

In classification we had

$z_i = y_i f(x_i)$
$\quad = y_i w^T x_i$

↓

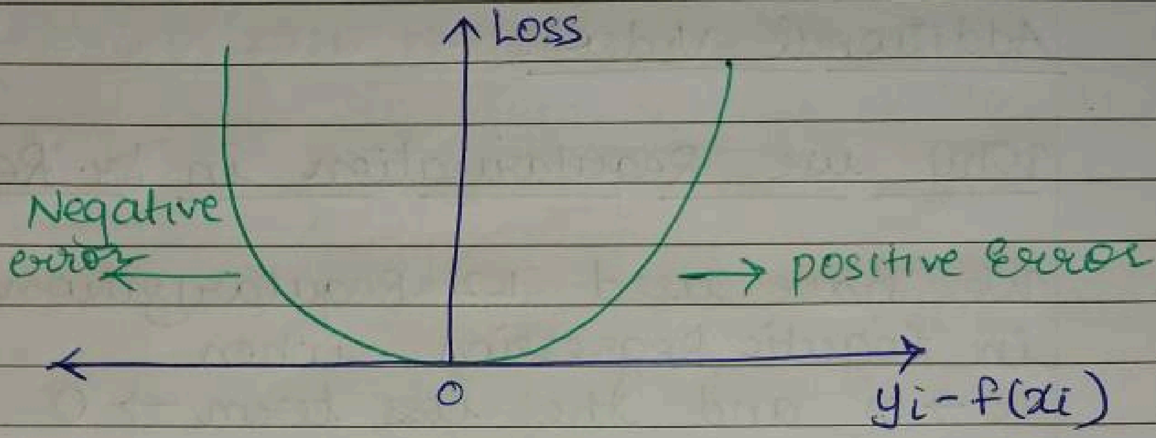
+ve : points are correctly classified
-ve —"— incorrectly —"—

In regression, we have

$z_i = y_i - f(x_i)$
$\quad = y_i - \hat{y}_i$

This represents error

Our x axis should represent error



Curve at 0 indicates $y_i - f(x_i) = 0$
Means error $= 0$
As the curve moves further towards +ve or -ve direction, the error (+ve/-ve) increases

The curve looks like parabola.
Equa^n of parabola: $y = x^2$
And our equation: $(y_i - f(x_i))^2$ which is our optimization problem.
If error is +ve/-ve, the loss is always +ve

0_1 Loss is only used for classification
But we cannot use that in Regression as loss cannot be 0 or 1.

Here loss is a squared function; hence we call it as squared loss

Using Loss-Min framework for regression
If we take square-Loss
we get Linear Regression OR
~~Ordg~~ Ordinary Least Square (ODL) OR
Linear Least Square (LLS)

Additional Video:

Why use Regularization in Lr. Regression

We have used L2 Regularization in Logistic Regression when $w \to \infty$ and the loss term $\to 0$ resulting in overfitting.

So, what problem occurs in Lr. Regression

Lr. Regression

$$w^*, w_0^* = \underset{w, w_0}{\text{Arg Min}} \overbrace{\sum_{i=1}^{n} (y_i - (w^T x_i + w_0))^2}^{\text{squared error}} + \underbrace{\lambda \|w\|_2^2}_{\text{L2 Regression}}.$$

Lets take one example.

Assume I have 3 features

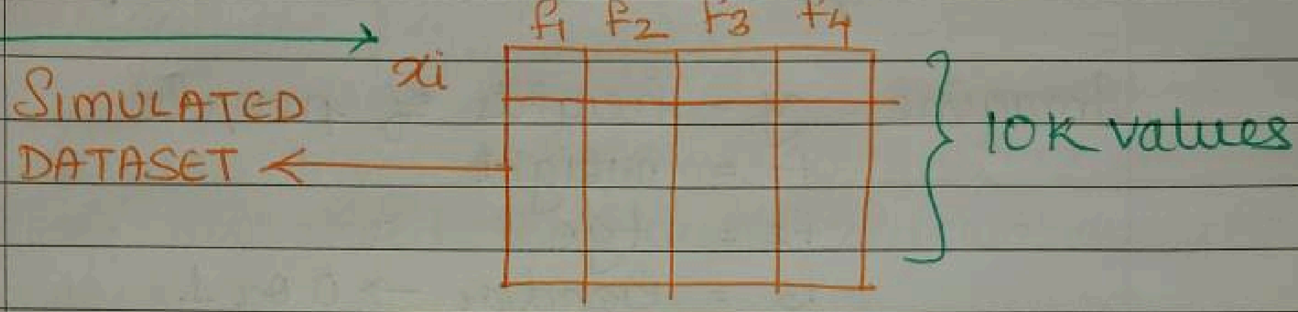
So $y_i = w_1 x_{i1} + w_2 x_{i2} + w_3 x_{i3}$

where $x_i \in \mathbb{R}^3$
$w \in \mathbb{R}^3$
$y_i \in \mathbb{R}^3$

Lets assume, I construct the data as follows:

$$\boxed{y_i = 2x_{i1} + 3x_{i2} + 0x_{i3}}$$

I generate random values of x_{i1}, x_{i2}, x_{i3}



This is artificially constructed dataset
Lets assume, I have 10K such points

Suppose we generate it as Dataset D
And we are asked to fit a Linear Regression Model.
Remember that we are given the dataset D and we dont know the underlying formula of those datapoints

So, the ideal solution should be

$$W = [2, 3, 0]$$

Such that

$$y_i = \underset{w_1}{2}\,x_{i1} + \underset{w_2}{3}\,x_{i2} + \underset{w_3}{0}\,x_{i3}$$

So, we are simulating a data using an underlying function and we are simulating the data, the ideal solution would be exactly the weights which were withheld for us.

Now, this is the simulation.

What happens in Real world ?

Imagine y = heights of people
f_1 = weight
f_2 = Age
f_3 = Gender → 0 or 1

And we know there is linear relationship between them.

$x_i \rightarrow w, a, g$

For one point we have x_i and y_i as follows:

$x_i = \{162.23, 30.24, 0\}$

So, we dont use the values as is
We convert it into some feasible value and save in dataset

Eg: $x_i = \{162, 30, 0\}$

Hence, we introduce some kind of error (Eg: Rounding off)

If we want to simulate real world data, then we need to add small error.

$$y_i = 2\left(x_{i1} + \epsilon_1\right) + 3\left(x_{i2} + \epsilon_2\right) + 0\left(x_{i3} + \epsilon_3\right)$$

And now we create dataset D' (10K data) for simulation

So, D → Perfectly simulated data

D' → Real world

Assume We are given D' and we are not told the underlying weight

Case 1:

Lets assume that I perfor Linear Regression without considering Regularization.

$$D' \xrightarrow{LR + No\ Reg}$$

Now since my data has ϵ_1, ϵ_2 and ϵ_3. It means the data is slightly corrupted by noise, we will only have squared error loss in optimization equa^n.

$$\begin{pmatrix} w^*, \\ w_0^* \end{pmatrix} = \underset{w, w_0}{\text{ArgMin}} \boxed{\sum_{i=1}^{n} \underbrace{(y_i - (w^T x_i + w_0))^2}_{\text{only term}}} + \underbrace{\lambda \|w\|_2^2}_{\text{wont be present}}$$

Hence the optimization term will try to minimize squared loss term as much as possible

Hence $D' \xrightarrow{LR + No\ Reg}$ final weights wont be exactly $(\underline{2, 3, 0})$

Final weights would look something like &. $W = [2.1, 3.06, 0.12]$.

And we get these deviated values because:
→ there is noise introduced in data
→ there is no regularizer used.

Case 2:

When we apply linear Regression with Regularization.

$$W = [2.01, 3.003, 0.003]$$

Means, we have values very much closer to original weights because
→ We have an additional term to take care of the noise
→ Regularizer is forcing Wis to be small
→ λ will control balance between Error and Regularizer.

Hence Overall the Optimizer is

$$\boxed{\text{Optimizer}} = \boxed{\begin{array}{c}\text{Trying to}\\ \text{Reduce the}\\ \text{Error}\end{array}} + \boxed{\begin{array}{c}\text{Ensuring that}\\ \text{Wis are}\\ \text{small}\end{array}}$$

The moment we ensure Regularizer, it will try to pull Wis down but wont make it 0 due to presence of square Loss.

Optimizer = $\underbrace{\text{Squared Loss}}_{\substack{\downarrow \\ \text{will reduce} \\ \text{Error}}}$ + $\underbrace{\text{Regularizaⁿ}}_{\substack{\downarrow \\ \text{bring down} \\ \text{Wis}}}$

Regularization is a way to control our weights from not overfitting

These type of weights which are closer to the underlying weights tend to generalize well for new unseen data.

34.3 Real world Cases for Linear Regression

Imbalanced data: Upsampling / Downsampling

For feature imp & interpretability
→ If features are not multicollinear, we can use feature weights
→ All things we learnt for Logistic Regression remains same.

No Multiclass

*** Feature Engineering / Feature Transformaⁿ
→ Same as Logistic Reg

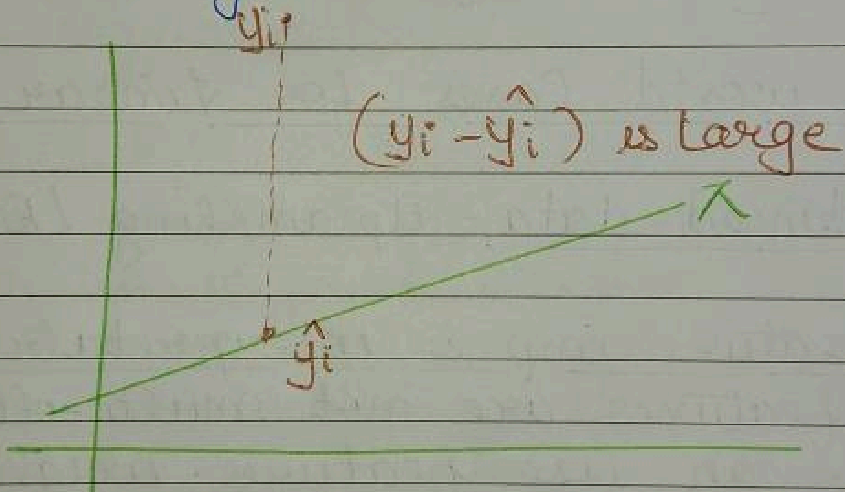
Regularizaⁿ
→ L1 Reg = Sparsity
As $\lambda \uparrow$; sparsity $\uparrow$ and the non imp feature weights will move to 0

One imp. difference between Logistic & linear Regression is

OUTLIERS

In Logistic Regression, we had $\sigma(x)$ which was limiting impact of outliers.

But linear Regression could be impacted by outliers



For an outlier, which is a very far away point, $(y_i - \hat{y}_i)$ is very large

And since Squared Loss $= \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$

It would be heavily impacted
One far away point might throw the entire model away.

So, what to do in such case?
We will apply same trick used in Logistic Regression

1. Using all of DTrain, we will find (w^*, w_0^*)
2. find all points very far away from Hyperplane (π) i.e. $(y_i - \hat{y}_i)$ is very High
3. Remove these points as outliers
4. Create a new dataset D'Train
5. Build your Linear Regression on D'Train

We can do these steps iteratively.
This technique is called as
<u>RANSAC</u> technique.