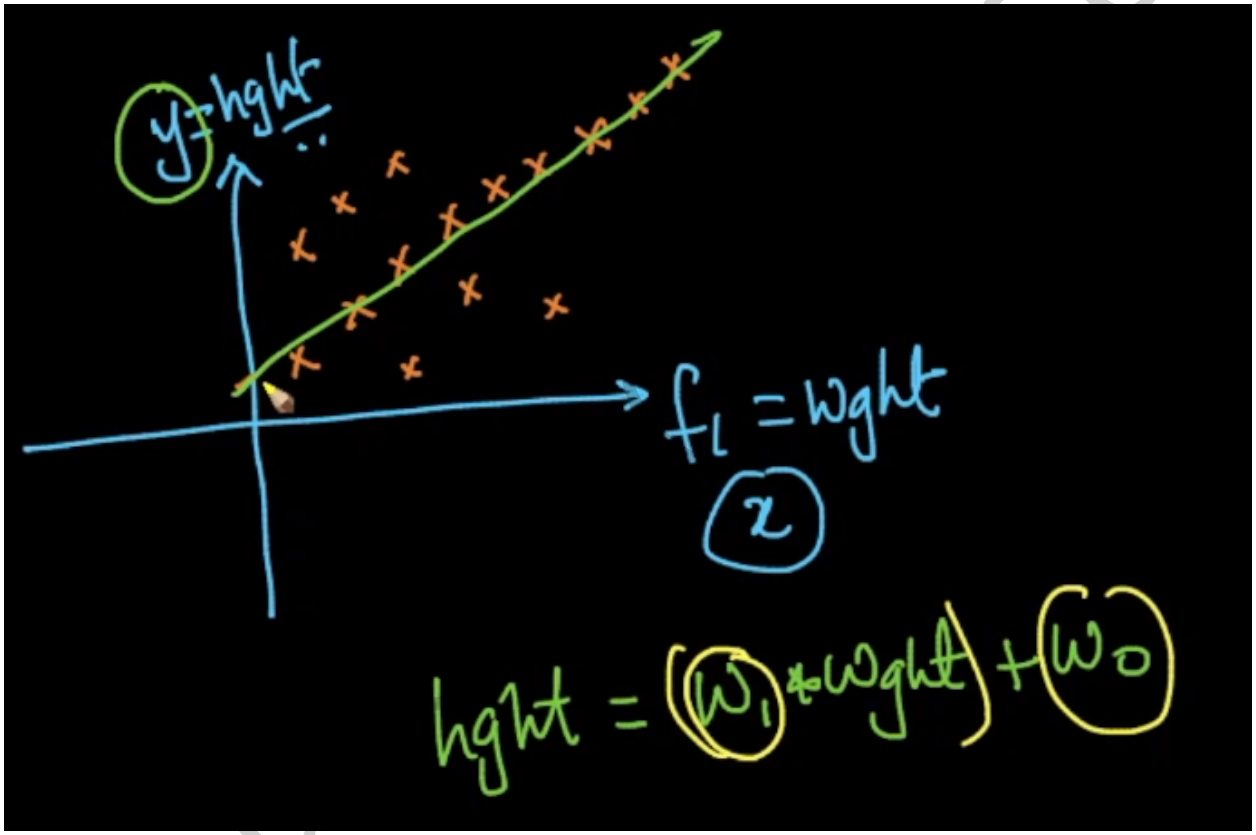
34.1 Geometric Intuition of Linear Regression

Linear Regression is an actual regression technique. Given the dataset, $D = \{x_i, y_i\}_{i=1}^{n}$ where $x_i \in R^d$ and $y_i \in R$.

Geometry behind Linear Regression

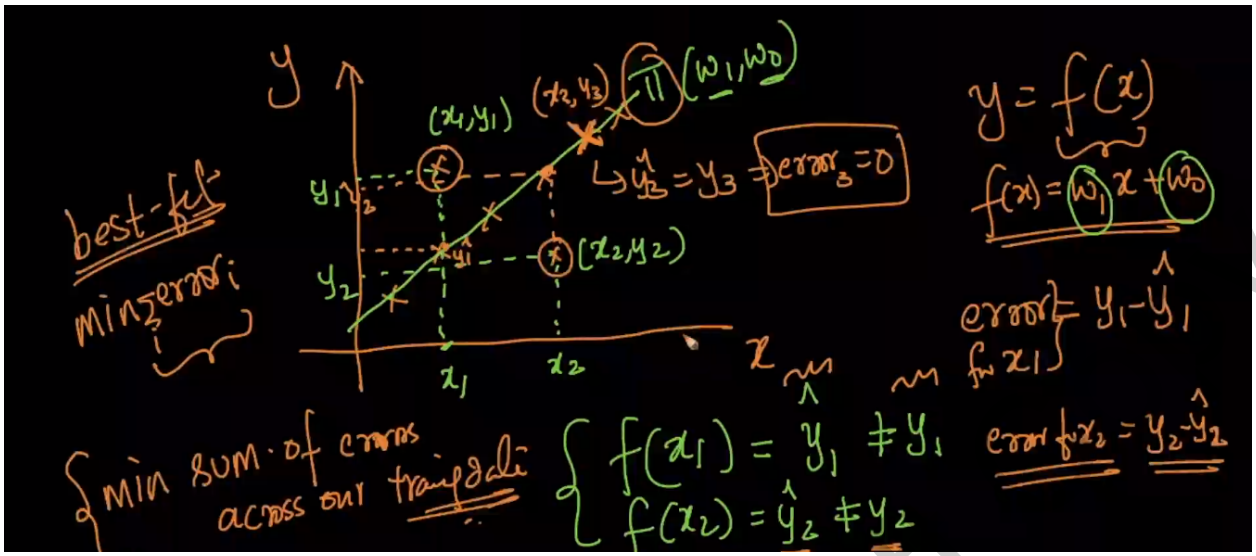
Let us consider the problem of predicting the height given weight, gender, ethnicity, hair color, etc.



The main objective in linear regression is to find a line/plane that fits the given data. In 2-D space, **height = (w_1*weight)+w_0**.

If we want to take another feature 'hair-color' also into consideration, then the formulation becomes **height = (w_1*weight) + (w_2*hair-color) + w_0**.

Linear Regression works on the assumption that most of the points lie on a linear surface. We have to find a line/plane that best fits the data points.



If any point (x_i, y_i) doesn't lie on the plane $\pi(w_1, w_0)$, then there occurs an error with that point. (ie., $y_i - y_i\hat{}$).

In the figure, we see the points (x_1, y_1) and (x_2, y_2) are not present on the plane. So

For $(x_1, y_1) \rightarrow$ **$f(x_1) = y_1\hat{}$** (where $y_1\hat{} \neq y_1$)

For $(x_2, y_2) \rightarrow$ **$f(x_2) = y_2\hat{}$** (where $y_2\hat{} \neq y_2$)

Error for $x_1 \rightarrow$ **$e_1 = y_1 - y_1\hat{}$**

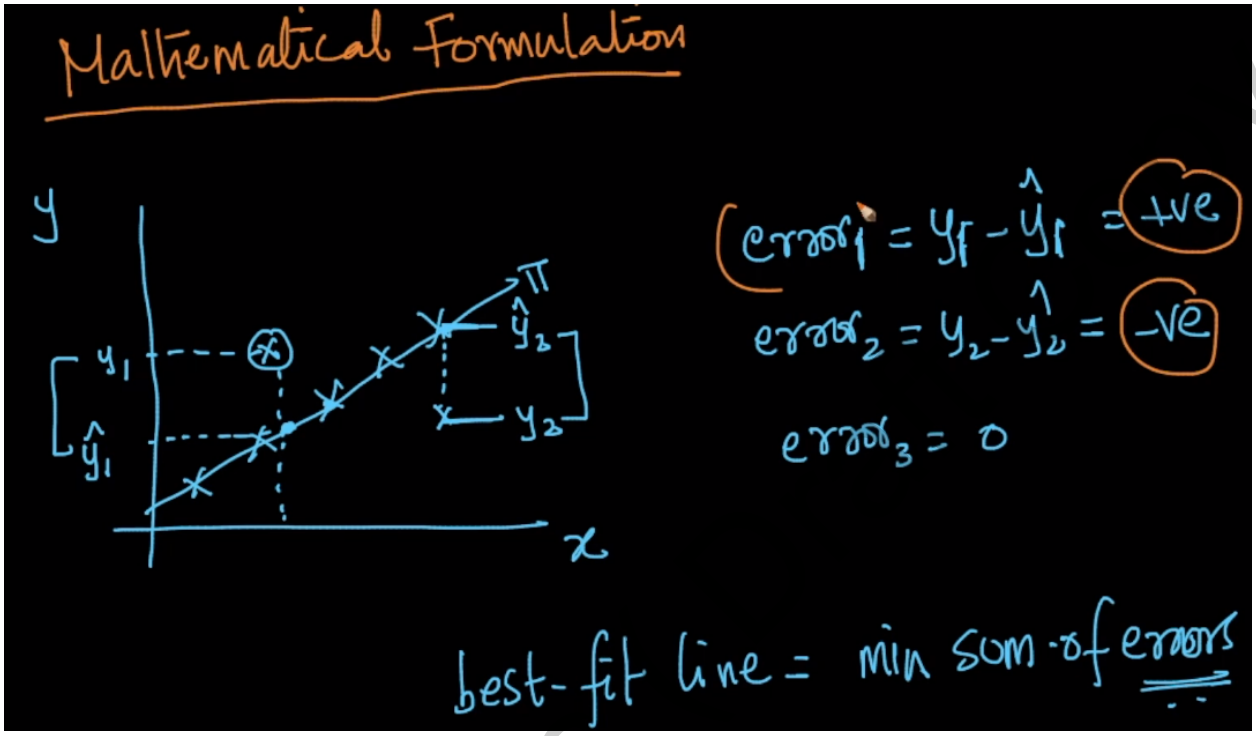
Error for $x_2 \rightarrow$ **$e_2 = y_2 - y_2\hat{}$**

We have to best fit the plane. It means we have to minimize the sum of errors for each point across the training data. So the term that has to be minimized is **$\Sigma_{i=1}^{n} (y_i - y_i\hat{})$.**

Difference between Logistic Regression and Linear Regression

- While both Linear Regression and Logistic Regression are linear models, their objective functions are slightly different.
- Logistic Regression minimizes the Log-Loss whereas Linear Regression minimizes the Squared loss.
- In Logistic Regression, we are trying to find a plane that separates the data points belonging to different classes, whereas in Linear Regression, we find a plane that passes through most of the data points.

34.2 Mathematical Formulation



For a point (x_i, y_i), error$(e_i) = y_i - y_i\hat{}$

In the above figure, we can see

For the point $(x_1, y_1) \rightarrow$ error(e_1) = +ve

For the point $(x_2, y_2) \rightarrow$ error(e_2) = -ve

If we have positive and negative values for the errors, they might get cancelled out and may not give the proper interpretation. Hence we take squares of the errors.

So the mathematical formulation for Linear Regression is we have to find the optimal (w, w_0) that maximizes the sum of squares of errors.

$$\mathbf{(w^*, w_0^*) = \text{arg-min}_{w,\, w0}\ \Sigma_{i=1}^{n}\ (y_i - y_i\hat{})^2}$$

Where $y_i\hat{} = f(x_i) = w^T x_i + x_0$

So now the optimization problem becomes

$$\mathbf{w^*,\ w_0^* = \text{arg-min}_{w,\, w0}\ \ \Sigma_{i=1}^{n}\ [y_i - (w^T x_i + x_0)]^2}$$

Linear Regression is also called **Ordinary Least Squares** (or) **Linear Least Squares**.

Regularization

The optimization problem with regularization becomes

$(w^*, w_0^*) = \text{arg-min}_{w, w0} \Sigma_{i=1}^{n} [y_i-(w^Tx_i+x_0)]^2 + \lambda||w||_2^2$

Same like in Logistic Regression, we can apply L1 regularization (or) L2 regularization (or) ElasticNet regularization.

Note

In the probabilistic interpretation of Linear Regression,

If $P(y_i|x_i) = N(\mu, \sigma^2)$, then using the basic tools of probability, we can easily derive the Linear Regression.

All the three interpretations of Linear Regression(ie., Geometric, Probabilistic, and Loss Minimization) leads us to the same point.

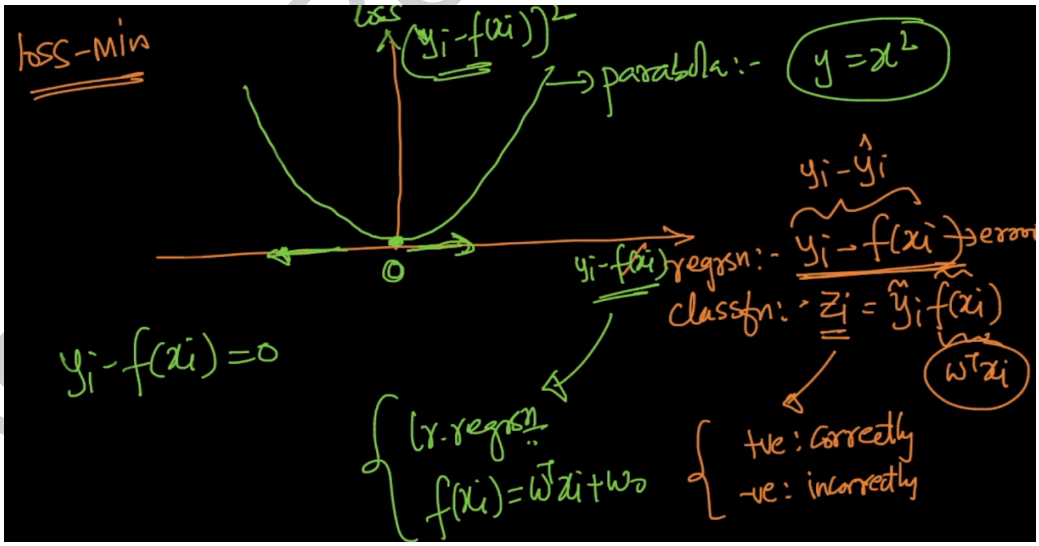
Loss Minimization Interpretation

In classification using Logistic Regression, we have seen

If $z_i = y_i*(w^Tx_i)>0$, then 'x_i' is correctly classified.

If $z_i = y_i*(w^Tx_i)<0$, then 'x_i' is misclassified.

In the case of Linear Regression, we see a plot with the error on the 'X' axis and loss on the 'Y' axis. Here we do not have right or wrong, as we have for classification in Logistic Regression. We only have the error. The plot looks like a parabola as shown below.



Using the loss minimization approach for regression, if we assume square loss, what we get is the Linear Regression.

The loss is penalized equally for both positive and negative points. The loss is always positive for both positive and negative errors.

The derivative of a squared function is easier when compared to the absolute value. Hence we take the square of the errors in the optimization problem.

Need for Regularization

Video Lecture Link: https://www.youtube.com/watch?v=iUm2Z1SKuGk

In Logistic Regression, we have seen that if $w \rightarrow \infty$, then the whole optimization fails. Hence we have introduced regularization to overcome this problem.

But in Linear Regression, the optimization problem is given as

$$\mathbf{w^* = arg\text{-}min_w \Sigma_{i=1}^{n} (y_i - w^T x_i)^2 + \lambda ||w||_2^2}$$

(We also can use L1 regularization in place of L2)

Let us consider an example. The relation we have in Linear Regression is

$$\mathbf{y_i = w^T x_i}$$

Let us assume we have 3 features in our problem.

$x_i \in R^3$, $w \in R^3$, $y_i \in R$

So now the relationship becomes

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

Let us consider now a simulated dataset 'D' such that

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

This is the underlying function and the output for every data point is computed using this function. If this dataset is given, and we are asked to fit a Linear Regression model, the ideal solution would be

w = [2,3,0]

But if we want to simulate the real-world datasets, then there will be minor errors associated with each feature. So for the real world simulated datasets, the relationship becomes

$$\mathbf{y_i = 2(x_{i1} + \in_1) + 3(x_{i2} + \in_2) + 0(x_{i3} + \in_3)}$$

The original data points in the training dataset are slightly corrupted by the noise. So now let this new simulated dataset be denoted as 'D^l'.

If we are not informed about the underlying function, then

Case 1: Applying Linear Regression without Regularization

Here as our dataset was corrupted with noise, and as we aren't applying the regularization, we'll have only the loss term in the optimization. As there is no regularization, the model tries to fit the training data as best as possible. So instead of [2,3,0], we get the weight vector as [2.1, 3.4, 0.2].

Case 2: Applying Linear Regression with Regularization

Here we get the weight vector somewhere like [2.01, 3.003, 0.003], but not perfectly as [2,3,0]. Here the regularization is trying to control the vector 'w' from reaching '∞', and the whole optimization problem is to reduce the squared error.

The reason why we do not get perfect values for the weight vector 'w' is, the main aim of 'w' here is to minimize the squared loss and the regularizer forces the vector 'w' to be as small as possible. The moment we introduce the regularizer, it pulls down the vector 'w' just to create a balance, as the squared loss tried to decrease, the regularizer tries to increase the loss (ie., the domination of loss is more than 'w').

Also in case 1, when there is no control on 'w', it tries to overfit the data. Hence in order to put some control on 'w', regularization is used.

These types of weights that are close to the underlying weights tend to generalize well on the unseen data. This is the reason why regularization occupies the first place.

34.3 Real-world cases

1) **Imbalanced Data**

As this is a regression problem, we do not have the problem of the dataset being imbalanced, as all the output values are the real values.

2) **Feature Importance and Model Interpretability**

If the features are not collinear, we can easily use $|w_j|$ to find out the feature importance.

3) **Multi-class**

As the output values are the real numbers, there are no chances for the existence of multiple classes in the dataset.

4) **Feature Engineering/Feature Transform**

Whatever Feature engineering techniques are applicable in Logistic Regression, the same can be applied here as well.

5) **Outliers**

In Logistic Regression, the sigmoid function has reduced the impact of the outliers on the model. But in Linear Regression, the procedure to remove the outliers is

a) Find w^*, w_0^*

b) Find the points very far away from the hyperplane 'π' based on the error values(ie., y_i-y_i^).

c) Remove these points as the outliers.

(ie., $D_{Train'} = D_{Train}$ - Outliers). Go to step 'a' and repeat this process.

Note: Regarding the video lecture 34.4, we aren't giving any notes, as it is purely a discussion on the code samples. We are providing you with the link to download the ipython notebook, and if you still have any queries, please feel free to post them in the comments section.

Link:https://drive.google.com/file/d/16yIAGm-A61vEgcnmmNY5rWCWFCDQ6Y-k/view