8

Linear Regression

True values → 2, 3, 4 (say)

Predicted values → 1.8, 2.5, 3.5 (say)

Coefficient of Determination:

Score → $1 - \frac{u}{v}$

$u = \sum \left(y_i^{True} - y_i^{Pred.}\right)^2$

$v = \sum \left(y_i^{True} - y_{Mean}^{True}\right)^2$

$\therefore u = \sum \left(y_i^{True} - y_i^{Pred}\right)^2$

$= \left[(2-1.8)^2 + (3-2.5)^2 + (4-3.5)^2\right]$

$= (0.2)^2 + (0.5)^2 + (0.5)^2 = 0.54$

$v = (2-3)^2 + (3-3)^2 + (4-3)^2$

$= 1 + 0 + 1$

$= 2$

$\therefore$ Score $= 1 - \frac{0.54}{2} = 1 - 0.27 = 0.73$

Error Minimising

Let's suppose we have n features and an output.

$\rightarrow x_i^1, x_i^2, x_i^3, \ldots, x_i^n$

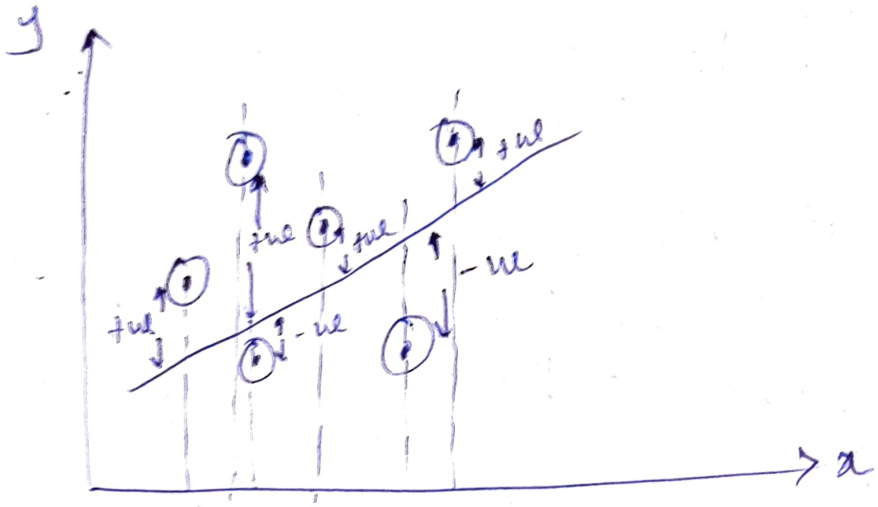
We plot a linear function $y = m_1 x_i^1 + m_2 x_i^2 + m_3 x_i^3 + \ldots + m_n x_i^n + c$

By linear regression, we find the $[m_1, m_2, \ldots m_n]$ and we get the line of best fit.

for 'ith' pt $\Rightarrow$,

$\text{Error}_i = y_i - (m_1 x_i^1 + m_2 x_i^2 + \ldots + m_n x_i^n)$

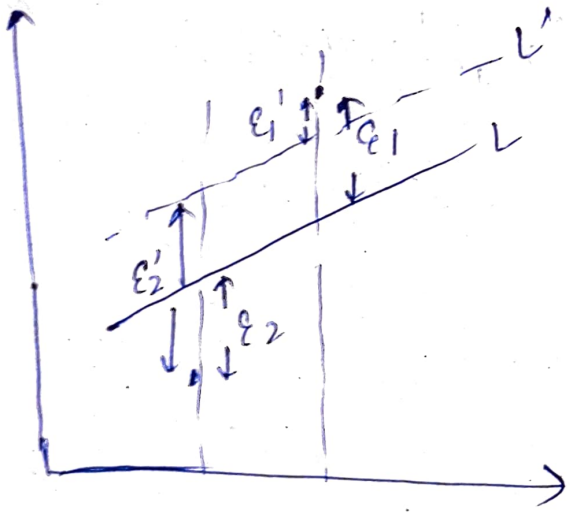
Total error $= \sum \text{Error}_i$



For normal scenario, the +ve and the -ve errors could cancel each other. But, we only want the absolute diff/error.

$\therefore$ | Error | = absolute error.

Let's consider another scenario.



By the above formula, both l & l' will give same error.

But, as we know that l is a better line than l'

$\therefore$ **Error ultimate** $=$ $(\text{Error})^2$ **

$\therefore (\varepsilon_1' + \varepsilon_2')^2 > (\varepsilon_1 + \varepsilon_2)^2$

$\therefore$ for n factors

$$\text{Error} = \left(y_i - \left(m_1 x_i^1 + m_2 x_i^2 + \cdots + m_n x_i^n\right)\right)^2$$

Eg: (1 dim.)

$$\text{Error} = \sum (y_i - (mx_i + c))^2$$

Line $= [y = mx + c]$

$$\text{Cost}(m, c) = \sum_i (y_i - (mx_i + c))^2$$

$$\frac{\partial(\text{Cost})}{\partial m} = 0 \qquad \frac{\partial(\text{Cost})}{\partial m} = 0.$$

$$\frac{\partial(co)}{\partial m} = \sum_i \frac{\partial}{\partial m}(y_i - (mx_i + c))^2$$

$$= \sum 2(y_i - (mx_i + c)) \frac{\partial[y_i - (mx_i + c)]}{\partial m}$$

$$\Rightarrow \sum 2(y_i - (mx_i + c))(-x_i) = 0$$

$$\Rightarrow \sum x_i y_i - m\sum x_i^2 - c\sum x_i = 0$$

$x*y.mean() \longrightarrow \frac{\sum x_i y_i}{N}$; $\frac{m\sum x_i^2}{N} \longrightarrow x*x.mean()$; $\frac{\sum x_i}{N} \longrightarrow x.mean()$

Similiarly we do $\frac{\partial(co)}{\partial c}$

$$\sum y_i - m\sum x_i = \sum c$$

$$y.mean() - m[x.mean()] = c$$

$\therefore$ (i) $[(x * y).mean()] - m[(x^2).mean()]$

$- c[x.mean()] = 0.$

(ii) $\rightarrow c = y.mean() - m * [x.mean()]$

from (i) & (ii)

$(x*y).mean() - m(x^2).mean() - x.mean()$
$* y.mean()$
$+ [x.mean()]^2 * m = 0.$

$\Rightarrow m = \dfrac{(x*y).mean() - x.mean() * y.mean()}{[x.mean()]^2 - x^2.mean()}$

$c = y.mean() - m * x.mean()$

Actually writing the code for 'fit', 'predict', 'score function'

ALGO

1) fit:

```
def fit(x, y):
    calc(m), calc(c)
    return f(m, c)
```

2) predict:

```
def predict(x, m, c):
    y_pred = mx + c
    return y_pred.
```

3) Coeficient of determination:

```
def COD(y_pred, y_true)
x = 1 - u/v
return x
```

Cost:

4) def cost (x, y, m, c)

return $\Sigma(y_i - (mx_i + c))^2$

CODE:

```
import numpy as np.
data = np.loadtxt("data.csv", delimeter=",")
x = data[:,0]
y = data[:,1]
from sklearn import model_selection.
x_train, x_test, y_train, y_test = model_selection
                                   .train_test_split(x, y)

// FIT FUNCTION
def fit (x_train, y_train):
    numerator = (x_train * y_train).mean()
                - (x_train).mean() * (y_train).mean()
    denominator = (x_train ** 2).mean()
                  - (x_train.mean()) ** 2
```

বিদ্যালয়

m = numerator / denominator

c = y_train.mean() - m * x_train.mean()

return m, c

// Let's call it.

// m, c = fit (x_train, y_train)

//PREDICT FUNCTION

def predict (x, m, c):

    return m * x + c

//SCORE FUNCTION

def score (y_truth, y_pred):

    u = ((y_truth - y_pred) ** 2).sum()

    v = (((y_truth - y_truth.mean()) ** 2).sum()

    return 1 - u/v

```
// COST FUNCTION
def cost(x, y, m, c):
    x=((y-(m*x+c))**2).sum()
    return x
// or ... mean()
```

The above algorithm was for 1 parameter.