Linear Regression

Linear Regression is the simplest regression model. Linear regression can be used only in **continous** variable.

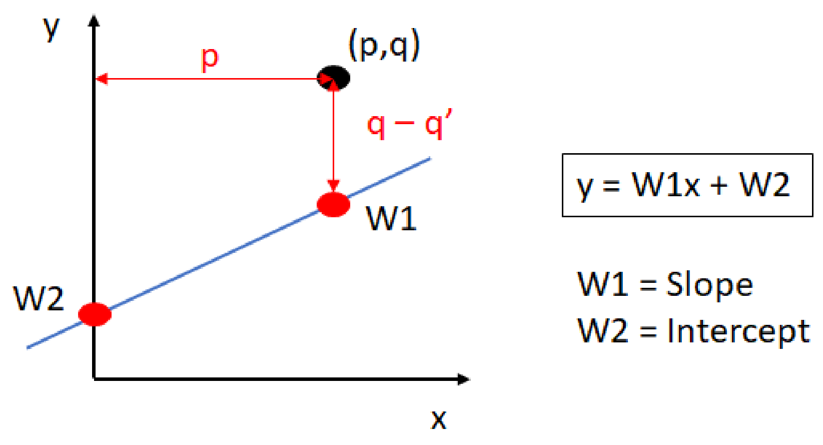
regression formula : $y = w1x + w2$

Here slop is $w1$ and y-intercept is $w2$.

```
In [39]:
```

```python
from IPython.display import Image
Image(filename='../img/linear_reg.png')
```

```
Out[39]:
```



Absolute Trick : when we got a point (p, q) and a line y the goal is to move the line closer to the point. To ensure that the line does not go to far from the point we use a parameter called **learning rate** $(\boldsymbol{a})$. We will multiply the learning rate to the line to get closer to the point and also ensure that it does not go further from the point.

$y = (w1 + p * a)x + w2 + a$ (a is the learning rate and (p, q) is a point)

Square Trick : If the point (p, q) is closer to the line then the line will move little if the point is far the line will move larger amount. Square trick add a **vertical** distance to the line.

lets say the point $A(p, q)$ that we want to move our line closer and another point $B(p, q^{'})$ over the line. the vertical distance between two point is $(q - q^{'})$ the equation would be

$y = (w1 + p(q - q^{'}) * a)x + w2 + a * (q - q^{'})$

Gradient Decent : Gradient decent is a process of minimizing error of the line by calculating the minimum distance. we keep iterating till getting the minimum error/cost.

Loss Function: A loss function is a mathematical function that takes in predicted values that we predicted using our model parameter and a set of actual values and tells us how well our model performs on the entire input data. Loss functions output a single value known as loss, which tells us how much loss or error we are getting when using a certain model parameter. Since loss functions output loss, the smaller the loss value, the better the model performance.

The most common loss function are:

1. Mean absolute error
2. Mean squared error

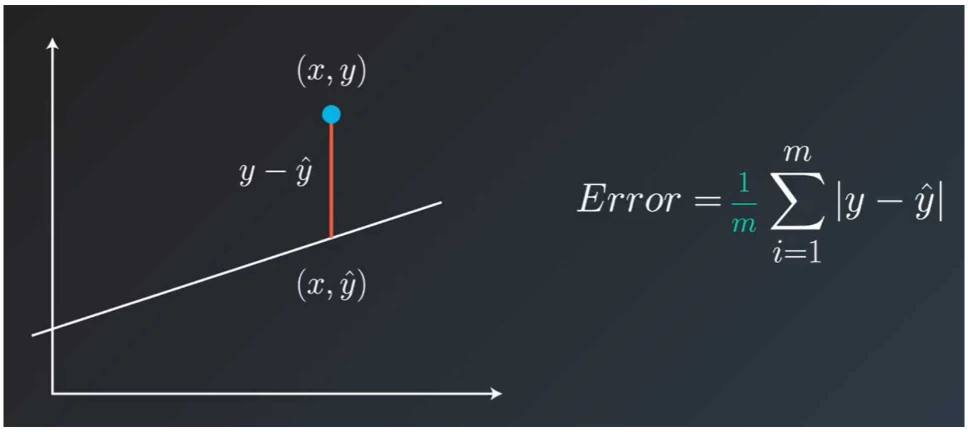
Mean Absolute Error: Mean absolute error is the sum of all the vertical distance between the points and line and divided by the total number os points.

```
In [1]:
```

In [1]:

```
from IPython.display import Image
Image(filename='../img/mean-absolute-error.png', width=600, height=500)
```

Out[1]:

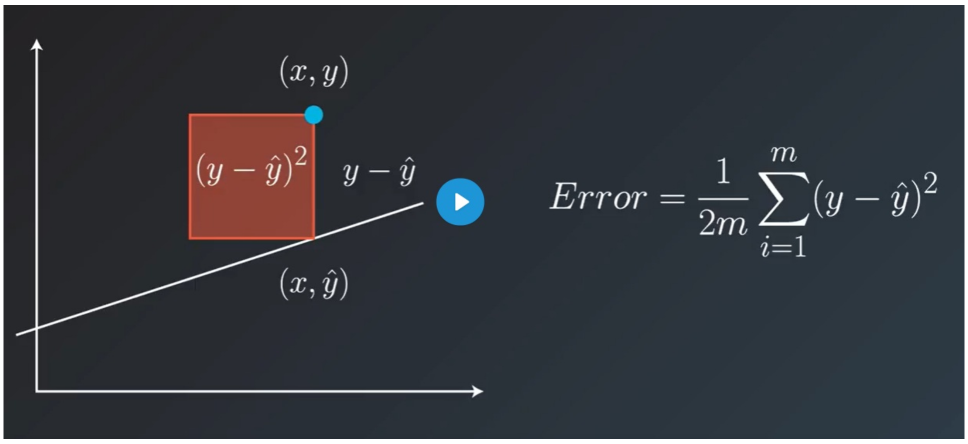


Mean Squared Error: Mean squared error is very similar to the mean absolute error, it is sum of all vertical distance between points and line, squared and divide by total number of points multiple 2.

In [2]:

```
from IPython.display import Image
Image(filename='../img/mean-squared-error.png', width=600, height=500)
```

Out[2]:



Regularization

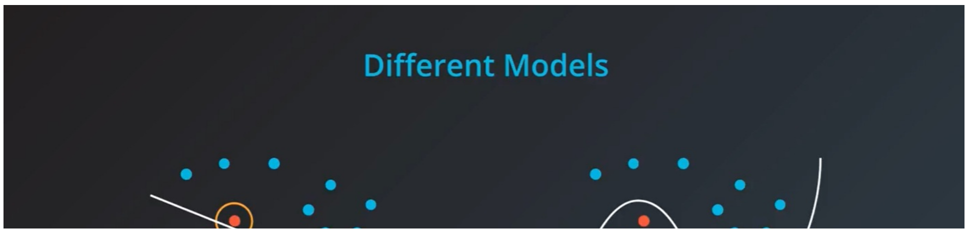
Regularized regression is the way of preventing model's overfitting or underfitting which may result from simple linear or compplex regression model.

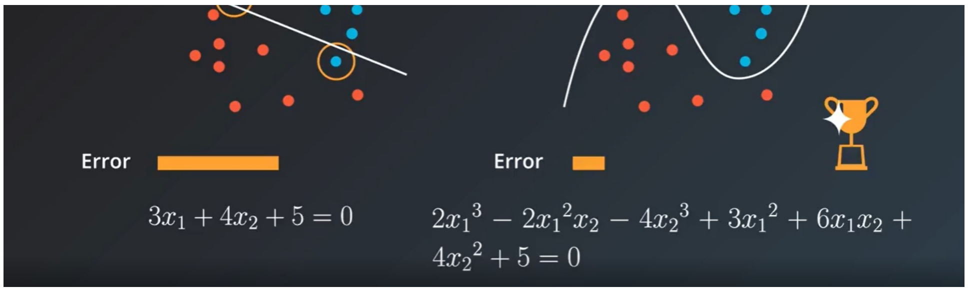
Let say we have two model 1st one is simple model but large error, next one is complex model but samller error.

In [3]:

```
from IPython.display import Image
Image(filename='../img/regularization1.png', width=600, height=500)
```

Out[3]:



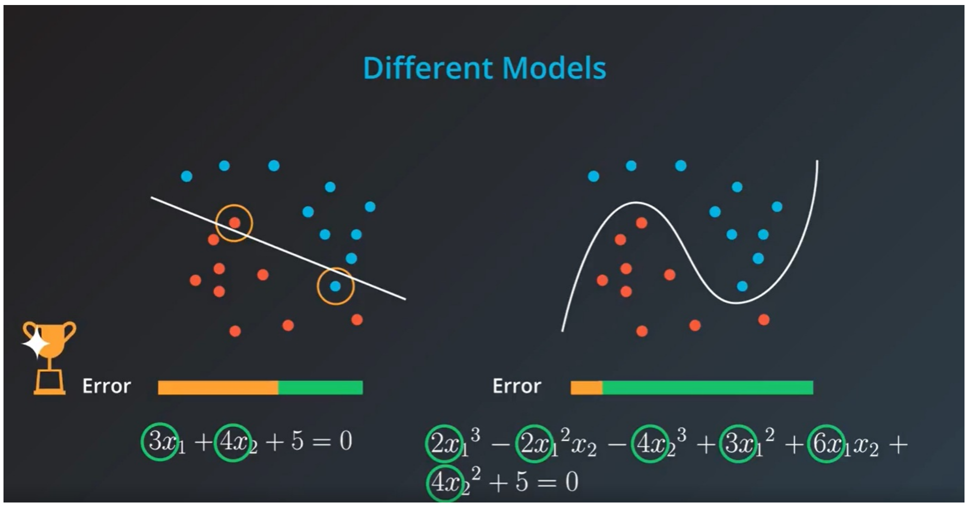

Error
Error
$3x_1 + 4x_2 + 5 = 0$
$2x_1^3 - 2x_1^2x_2 - 4x_2^3 + 3x_1^2 + 6x_1x_2 + 4x_2^2 + 5 = 0$

if we take all the cooefficient of the model and add them, we will see that simpler model with larger error has lesser error than the complex model of smaller error.

In [4]:

```
from IPython.display import Image
Image(filename='../img/regularization2.png', width=600, height=500)
```

Out[4]:


Different Models
Error
Error
$3x_1 + 4x_2 + 5 = 0$
$2x_1^3 - 2x_1^2x_2 - 4x_2^3 + 3x_1^2 + 6x_1x_2 + 4x_2^2 + 5 = 0$

There are two types of regularization:

1. L1 or Lasso regularization
2. L2 or Ridge regularization

L1/Lasso regularization: Takes the absolute value of the cooefficient and add

L2/Ridge regularization: Takes the squared value of the cooefficient and add

Notes :

- Linear Regression Works Best When the Data is Linear
- Linear Regression is Sensitive to Outliers