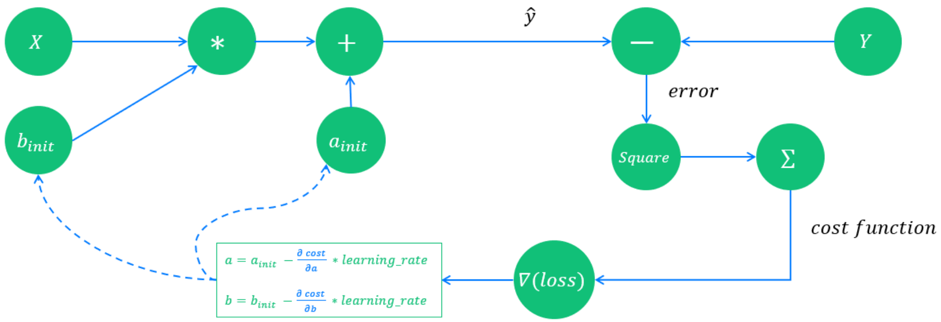
# Linear Model

**Srikanth Dakoju**
GitHub: https://github.com/srikanthdakoju/custom-regression-training-tensorflow2
Udemy: https://www.udemy.com/course/draft/3075150/?referralCode=B685C9F6A839572F40CA

# What you will learn




$$\hat{y} = b + W * X$$

**Linear Regression**

$$\nabla f(b, W)$$

**Gradient Descent Algorithm**

**Model Architecture (Perceptron)**

**Advanced TensorFlow**
**(Custom Training)**

**TensorFlow Keras**

TensorFlow 2.x

# Fitting
# Linear Model

DATA SCIENCE
ANYWHERE

# Consider Data

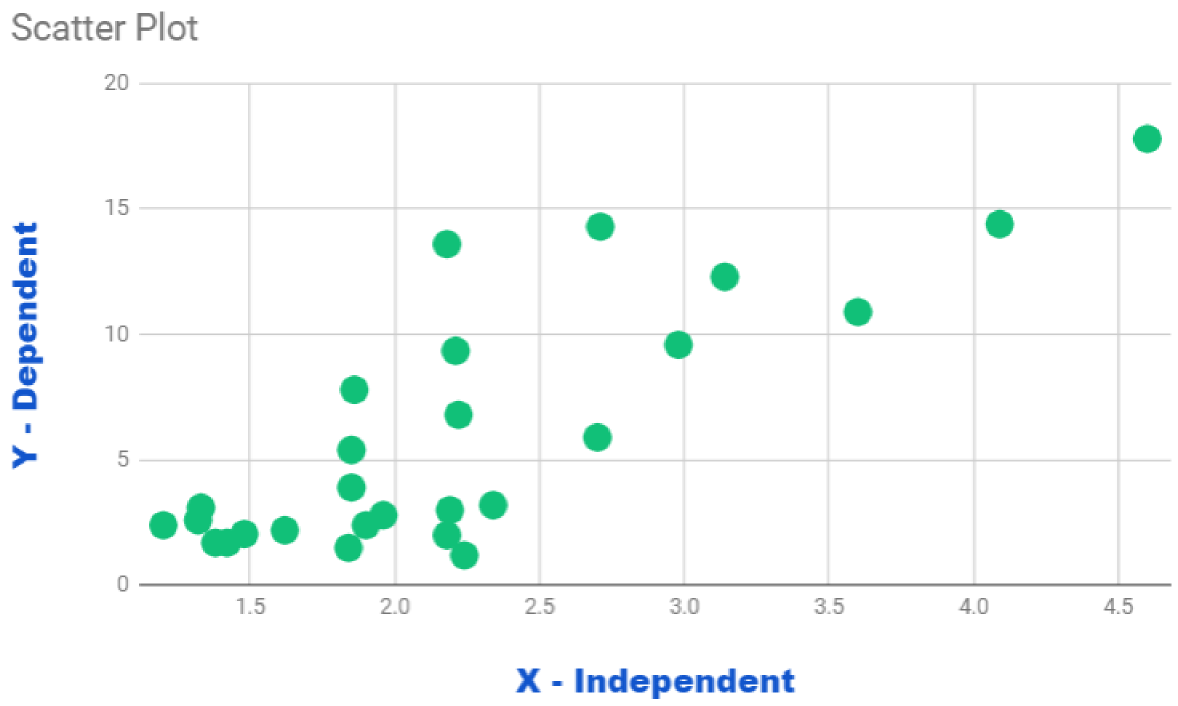
TensorFlow 2.x

| X | Y |
|---|---|
| 1.42 | 1.7 |
| 1.86 | 7.8 |
| 1.48 | 2.05 |
| 3.14 | 12.3 |
| 2.21 | 9.35 |
| 1.96 | 2.8 |
| 1.2 | 2.4 |
| 1.9 | 2.4 |
| 4.09 | 14.4 |
| 2.98 | 9.59 |
| 2.19 | 3 |
| 1.84 | 1.5 |
| 2.18 | 13.6 |
| 1.33 | 3.1 |
| 2.18 | 2 |
| 2.22 | 6.8 |
| 2.24 | 1.2 |
| 1.62 | 2.2 |
| 1.32 | 2.6 |
| 1.85 | 5.4 |
| 1.85 | 3.9 |
| 2.7 | 5.9 |
| 3.6 | 10.9 |
| 4.6 | 17.8 |

# Consider Data

TensorFlow 2.x

| X | Y |
|---|---|
| 1.42 | 1.7 |
| 1.86 | 7.8 |
| 1.48 | 2.05 |
| 3.14 | 12.3 |
| 2.21 | 9.35 |
| 1.96 | 2.8 |
| 1.2 | 2.4 |
| 1.9 | 2.4 |
| 4.09 | 14.4 |
| 2.98 | 9.59 |
| 2.19 | 3 |
| 1.84 | 1.5 |
| 2.18 | 13.6 |
| 1.33 | 3.1 |
| 2.18 | 2 |
| 2.22 | 6.8 |
| 2.24 | 1.2 |
| 1.62 | 2.2 |
| 1.32 | 2.6 |
| 1.85 | 5.4 |
| 1.85 | 3.9 |
| 2.7 | 5.9 |
| 3.6 | 10.9 |
| 4.6 | 17.8 |

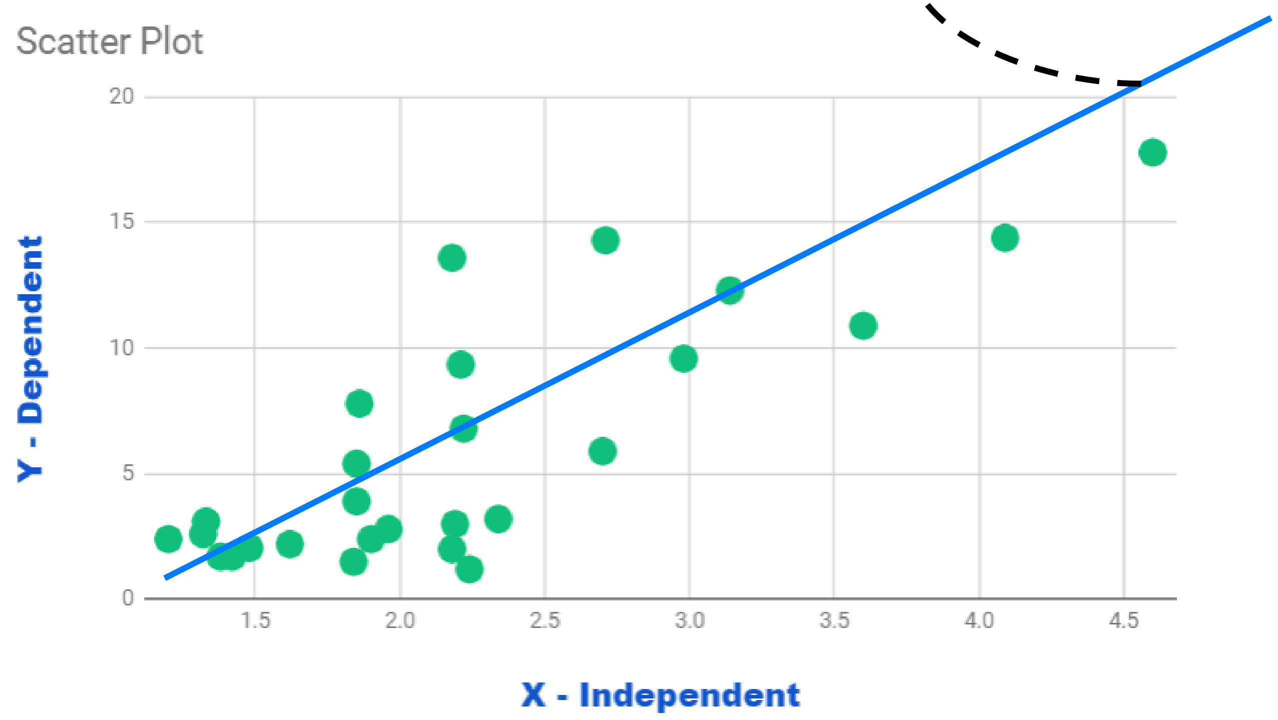


# Linear Regression

TensorFlow 2.x

| X | Y |
|---|---|
| 1.42 | 1.7 |
| 1.86 | 7.8 |
| 1.48 | 2.05 |
| 3.14 | 12.3 |
| 2.21 | 9.35 |
| 1.96 | 2.8 |
| 1.2 | 2.4 |
| 1.9 | 2.4 |
| 4.09 | 14.4 |
| 2.98 | 9.59 |
| 2.19 | 3 |
| 1.84 | 1.5 |
| 2.18 | 13.6 |
| 1.33 | 3.1 |
| 2.18 | 2 |
| 2.22 | 6.8 |
| 2.24 | 1.2 |
| 1.62 | 2.2 |
| 1.32 | 2.6 |
| 1.85 | 5.4 |
| 1.85 | 3.9 |
| 2.7 | 5.9 |
| 3.6 | 10.9 |
| 4.6 | 17.8 |

$$\hat{y} = a + b * X$$

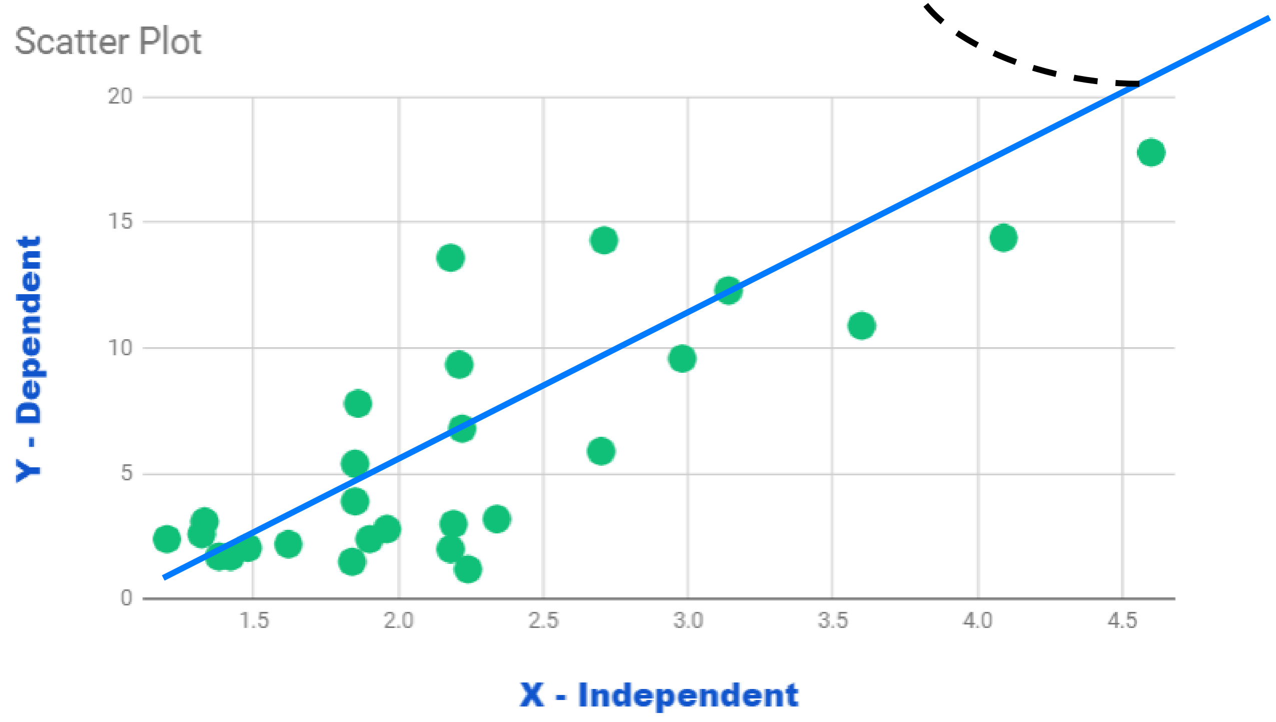



$$a = ?$$
$$b = ?$$

# Linear Regression



$$\hat{y} = a + b \ * X$$

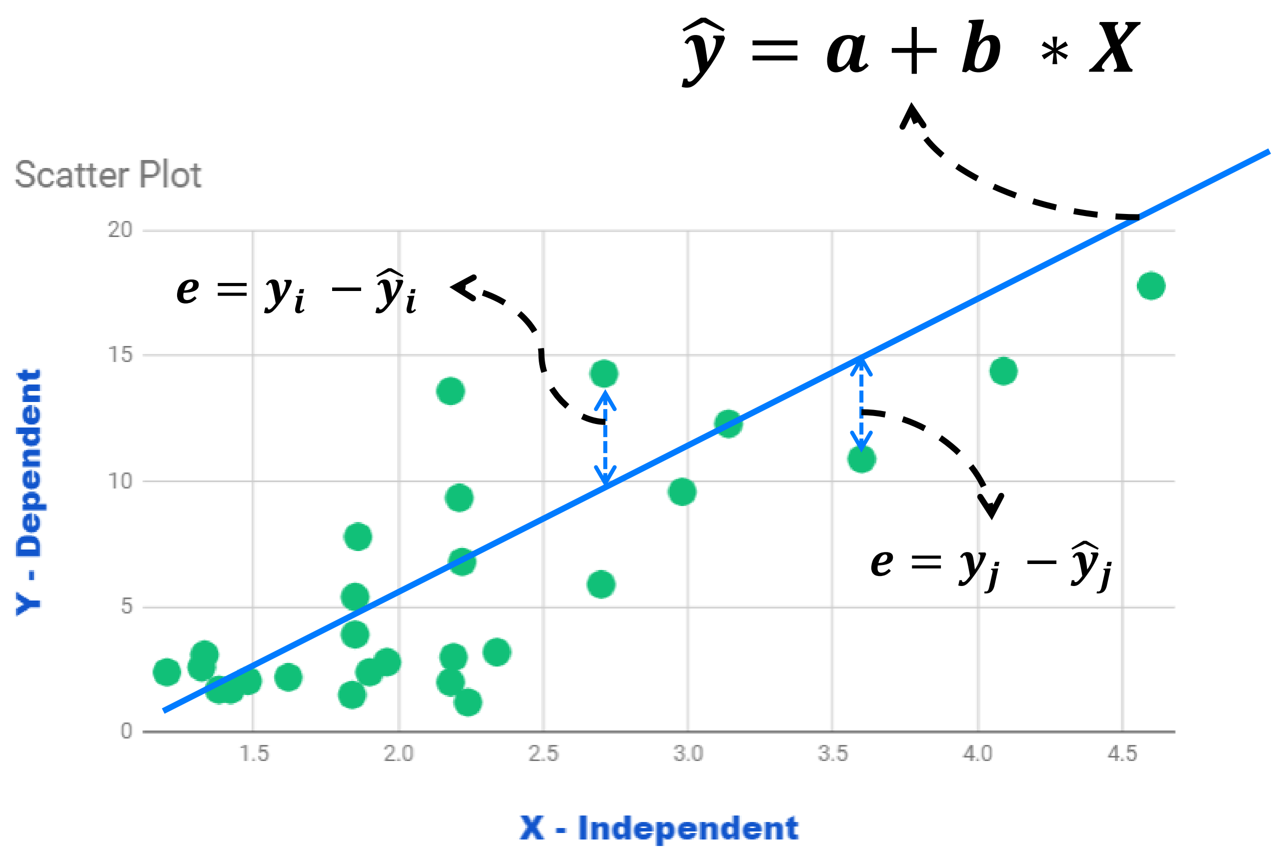



Linear Regression is a line, which will identify the relation between independent and dependent variable.

Such a way that line show has minimum **sum of squared error** or mean squared error

# Linear Regression

TensorFlow 2.x

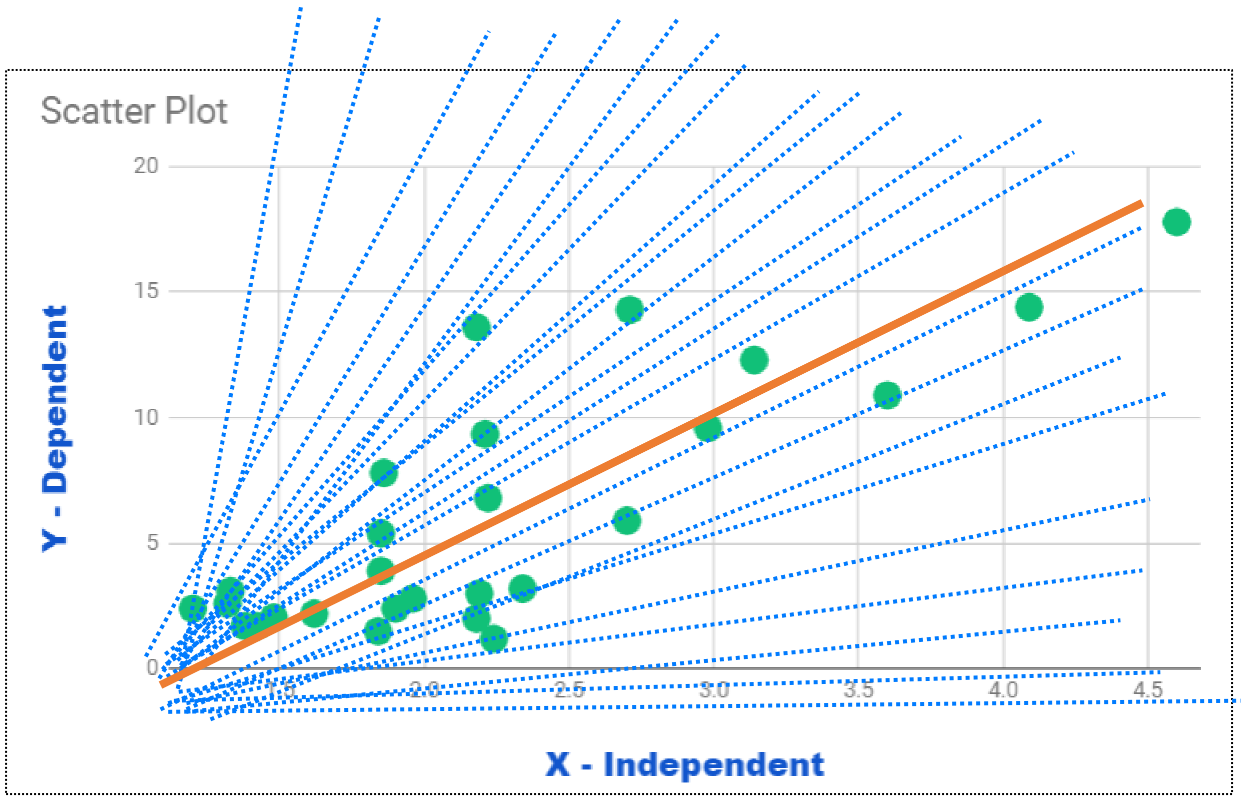



$$e = y_i - \hat{y}_i$$

$$SSE = \sum_i (y_i - \hat{y}_i)^2$$

DATA SCIENCE ANYWHERE

# Linear Regression




$$\hat{y} = a + b * X$$

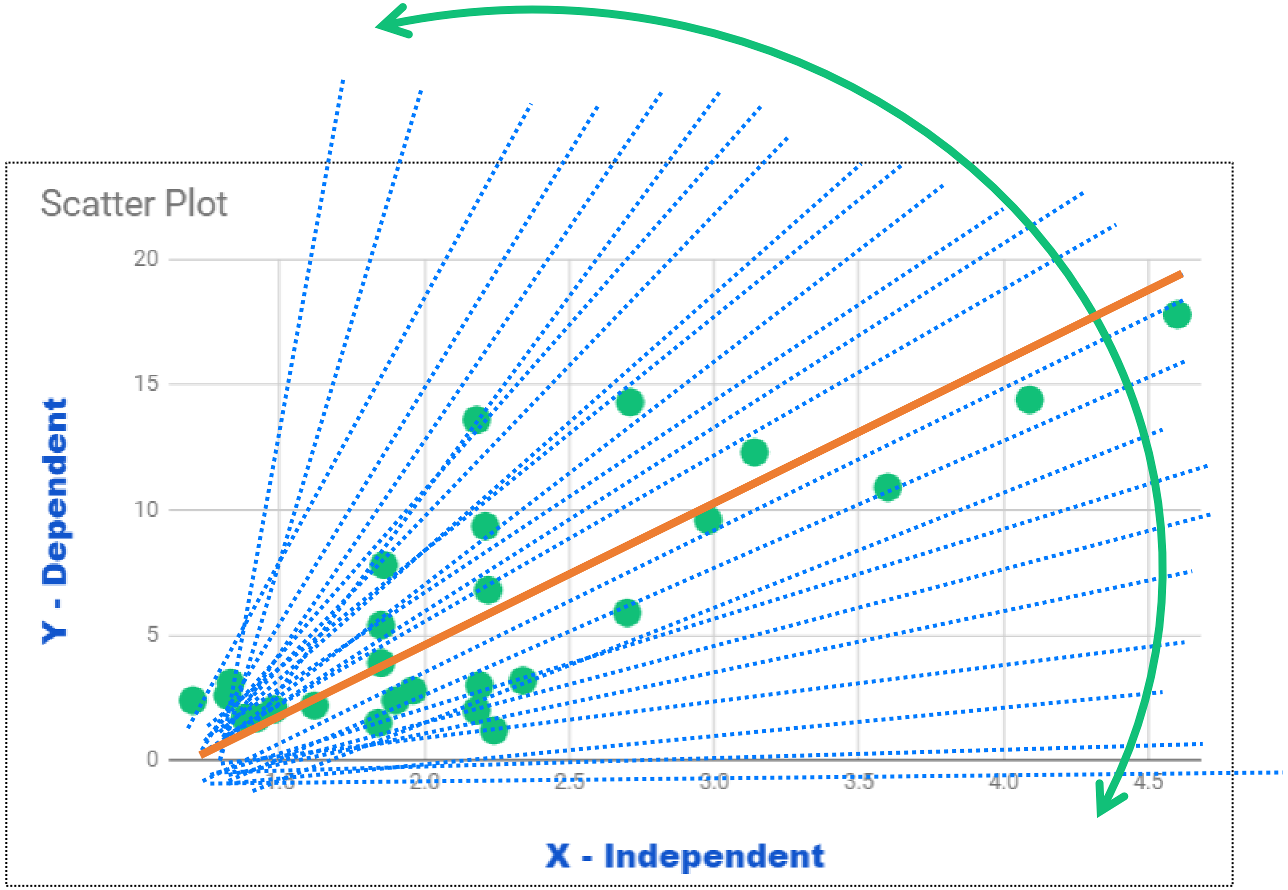
Select the line with minimum SSE

# Gradient Descent

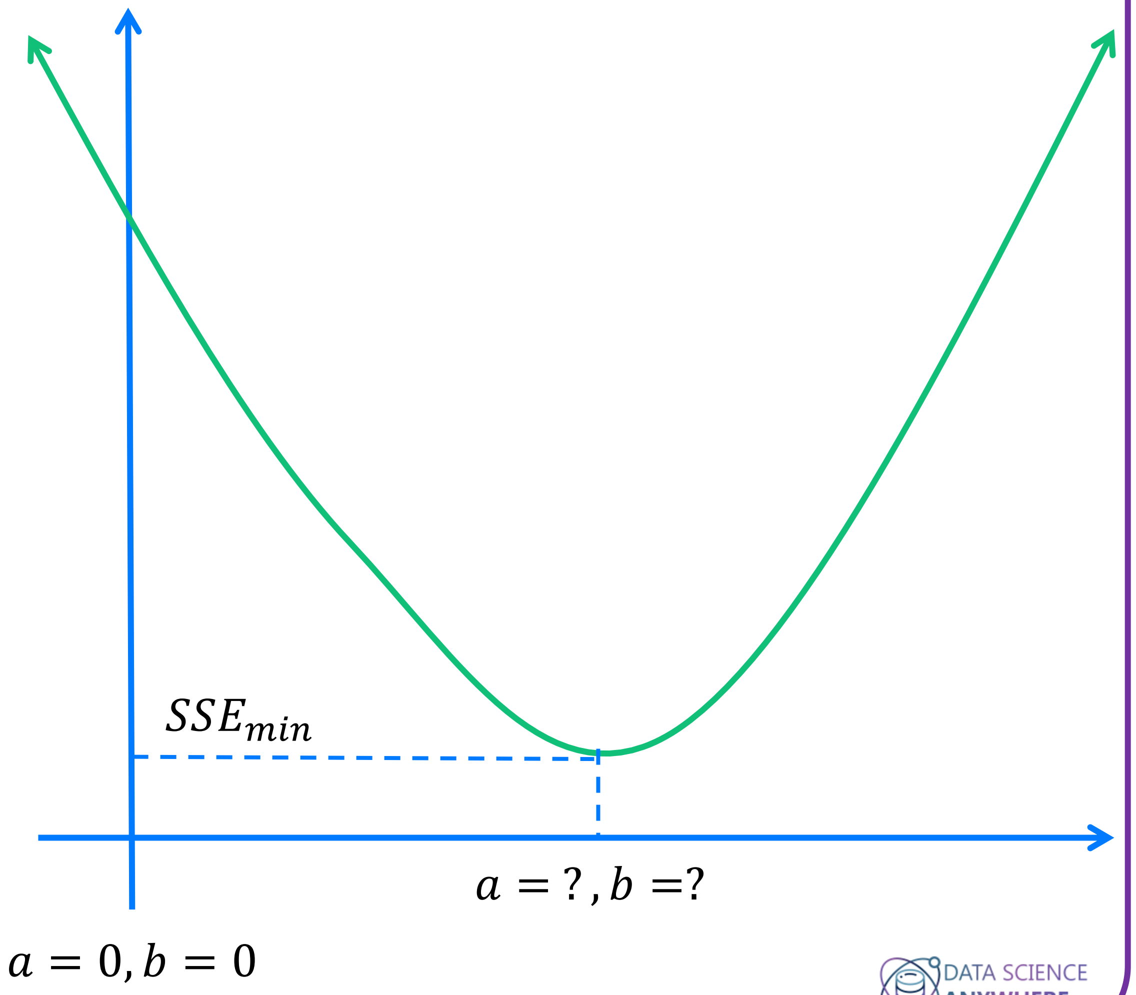
# Linear Regression

TensorFlow 2.x

$$\hat{y} = a + b * X$$

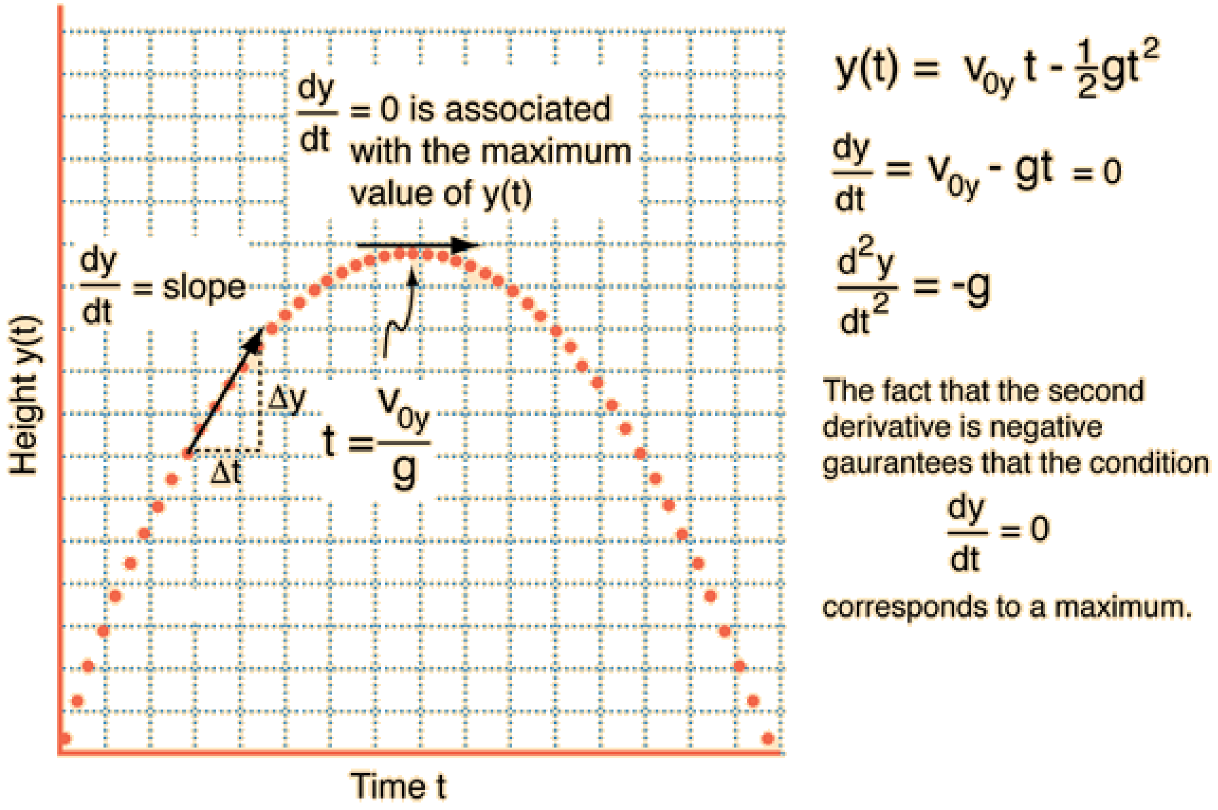



*Sum of Squared Error (SSE)*



# Calculus – Maxima and Minima

One of the great powers of calculus is in the determination of the maximum or minimum value of a function. Take f(x) to be a function of x. Then the value of x for which the derivative of f(x) with respect to x is equal to zero corresponds to a maximum, a minimum or an inflexion point of the function f(x).

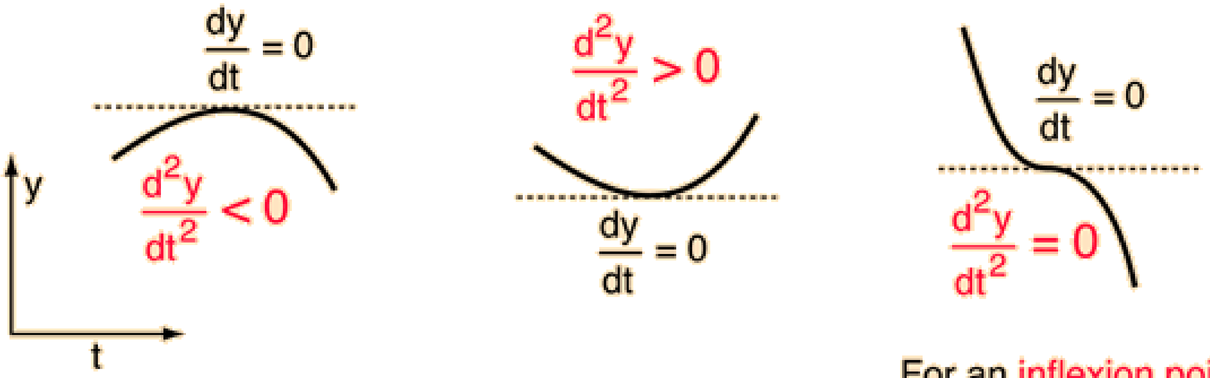


TensorFlow 2.x

# Calculus – Maxima and Minima

The derivative of a function can be geometrically interpreted as the slope of the curve of the mathematical function y(t) plotted as a function of t. The derivative is positive when a function is increasing toward a maximum, zero (horizontal) at the maximum, and negative just after the maximum. The second derivative is the rate of change of the derivative, and it is negative for the process described above since the first derivative (slope) is always getting smaller. The second derivative is always negative for a "hump" in the function, corresponding to a maximum.

The second derivative demonstrates whether a point with zero first derivative is a maximum, a minimum, or an inflexion point.




For a maximum, the second derivative is negative. The slope of the curve ( first derivative) is at first positive, then goes through zero to become negative.

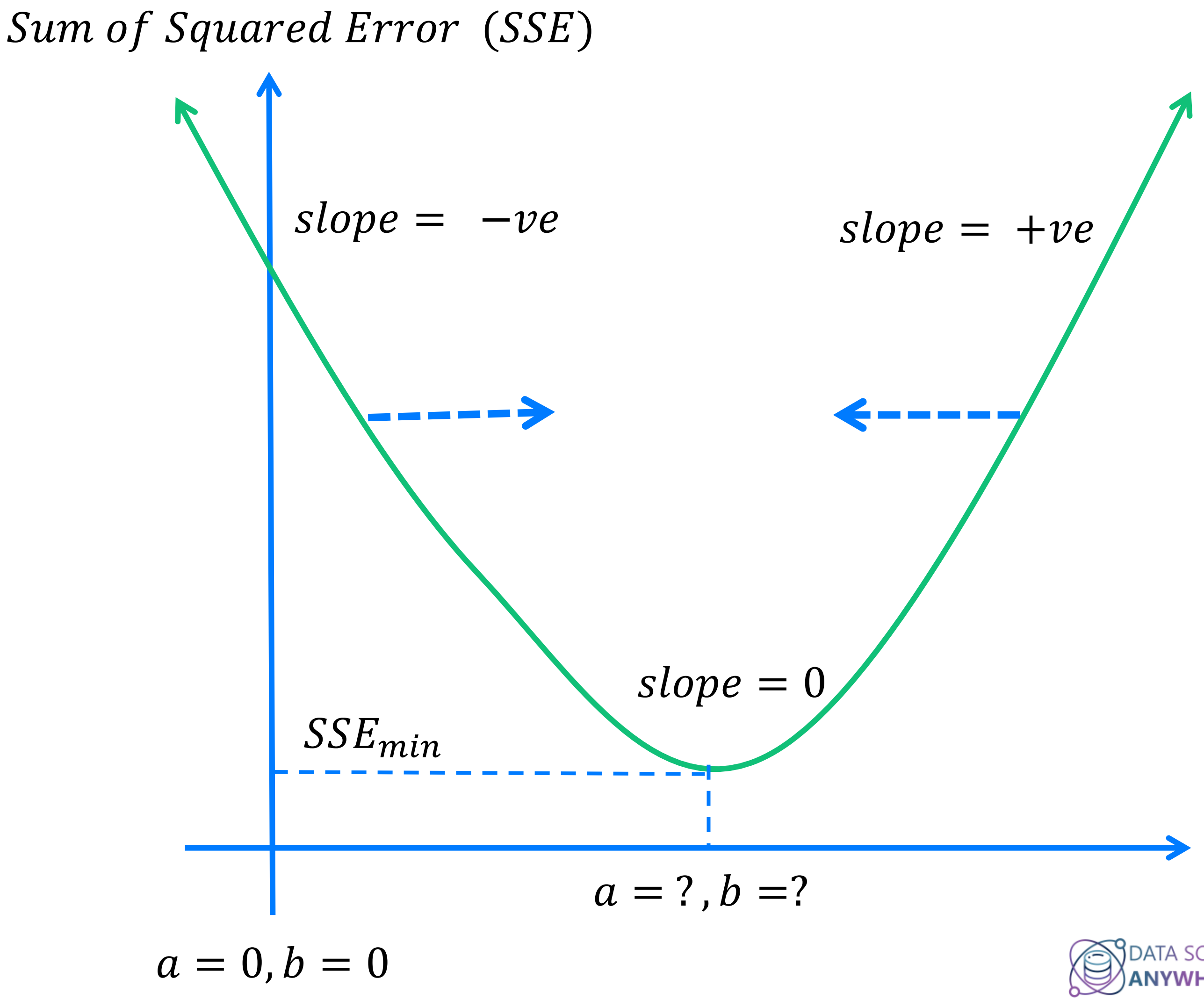
For a minimum, the second derivative is positive. The slope of the curve = first derivative is at first negative, then goes through zero to become positive.

For an inflexion point, the second derivative is zero at the same time the first derivative is zero. It represents a point where the curvature is changing its sense. Inflexion points are relatively rare in nature.

# Gradient Descent

TensorFlow 2.x

$$\hat{y} = a + b * X$$




DATA SCIENCE ANYWHERE

# Gradient Descent

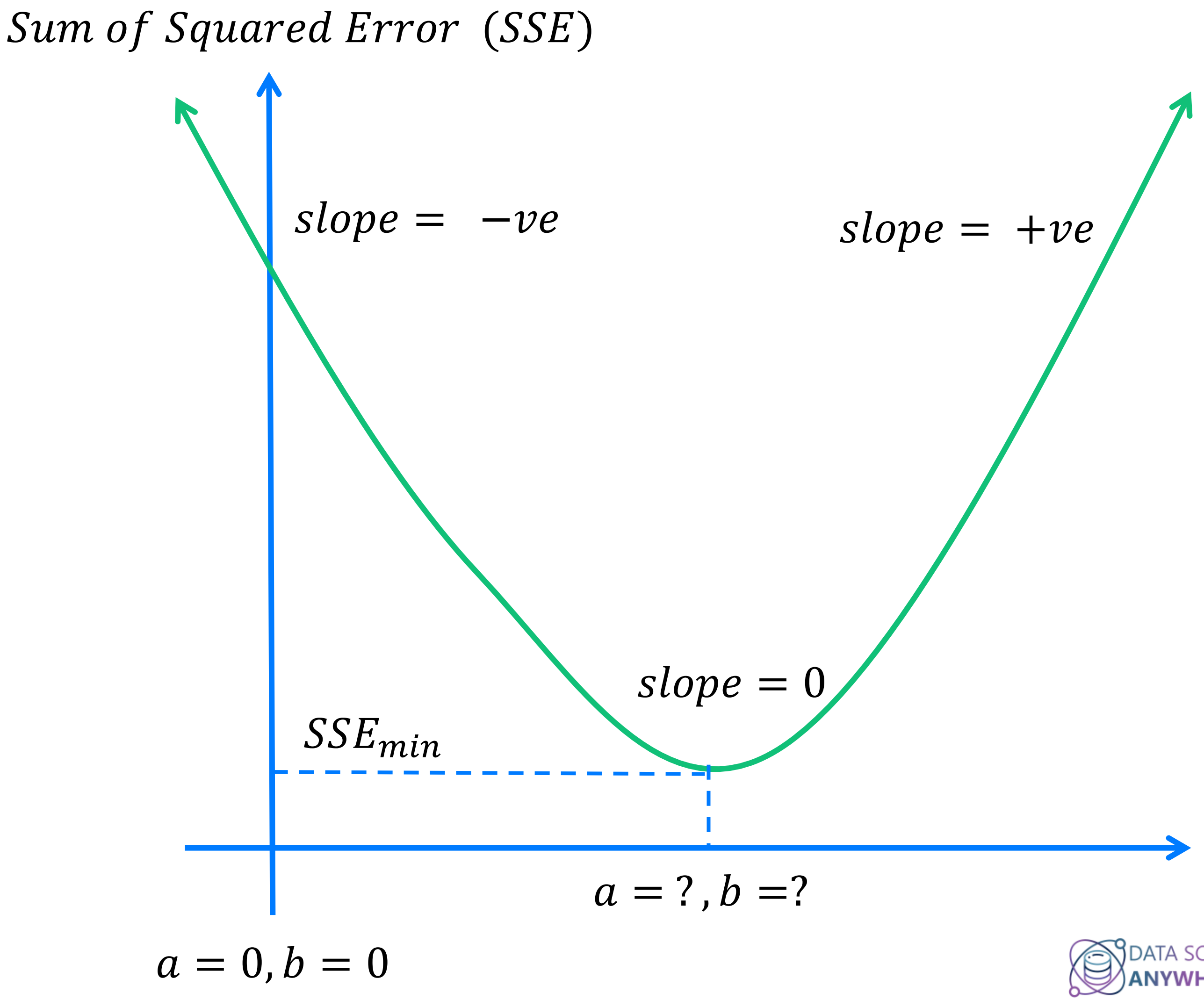
TensorFlow 2.x

$$\boldsymbol{SSE} = \sum_i (y_i - \hat{y}_i)^2$$

$$= \sum_i (y_i - (a + b * X_i))^2$$

$$f(a, b) = \sum_i (y_i - (a + b * X_i))^2$$

$$\hat{y} = a + b * X$$




DATA SCIENCE ANYWHERE

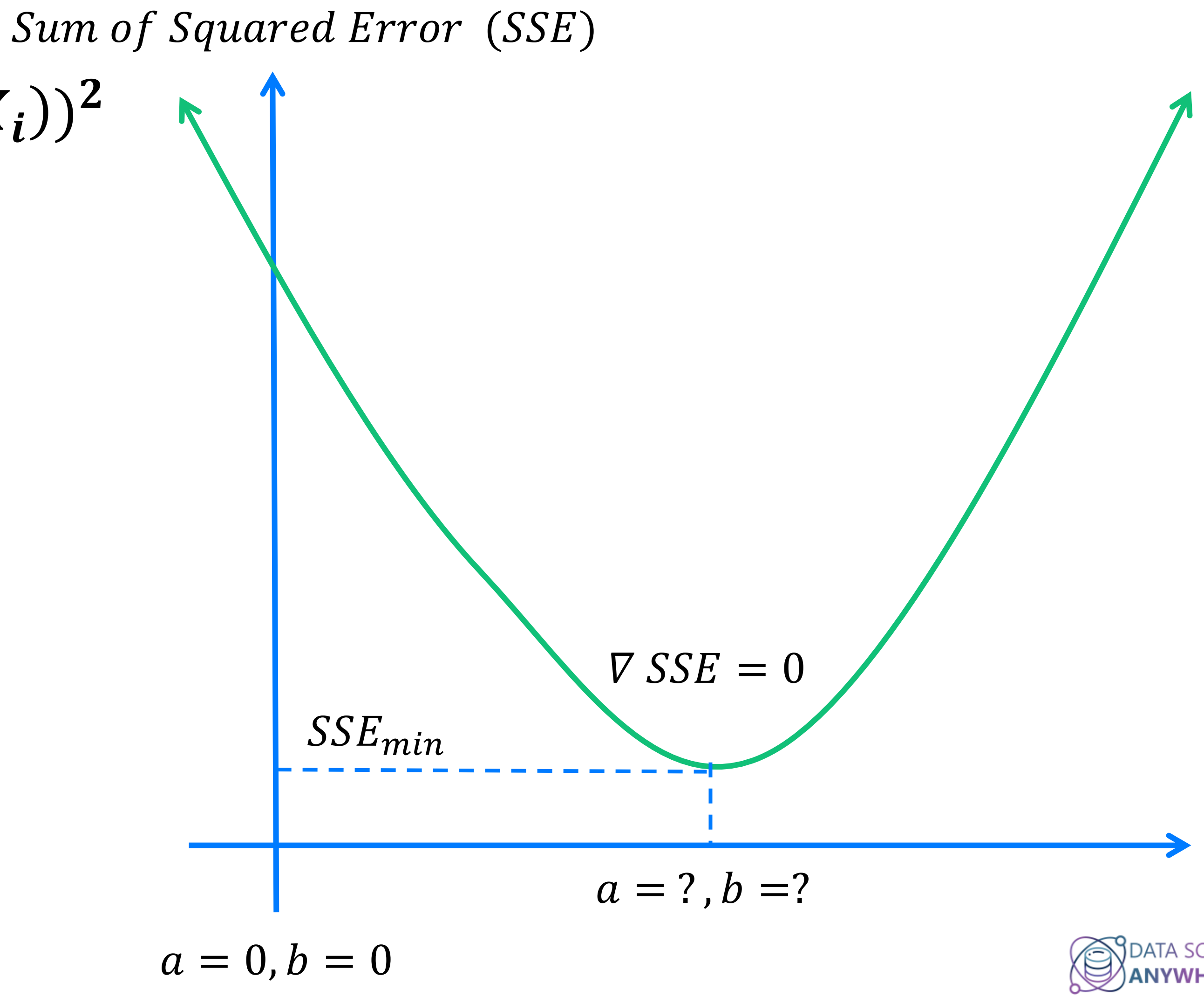
# Gradient Descent

TensorFlow 2.x

$$\boldsymbol{f(a,b) = cost = \sum_i (y_i - (a + b * X_i))^2}$$

$$\nabla f(a,b) = \begin{pmatrix} \dfrac{\partial\, cost}{\partial a} \\ \\ \dfrac{\partial\, cost}{\partial b} \end{pmatrix}$$

Gradient



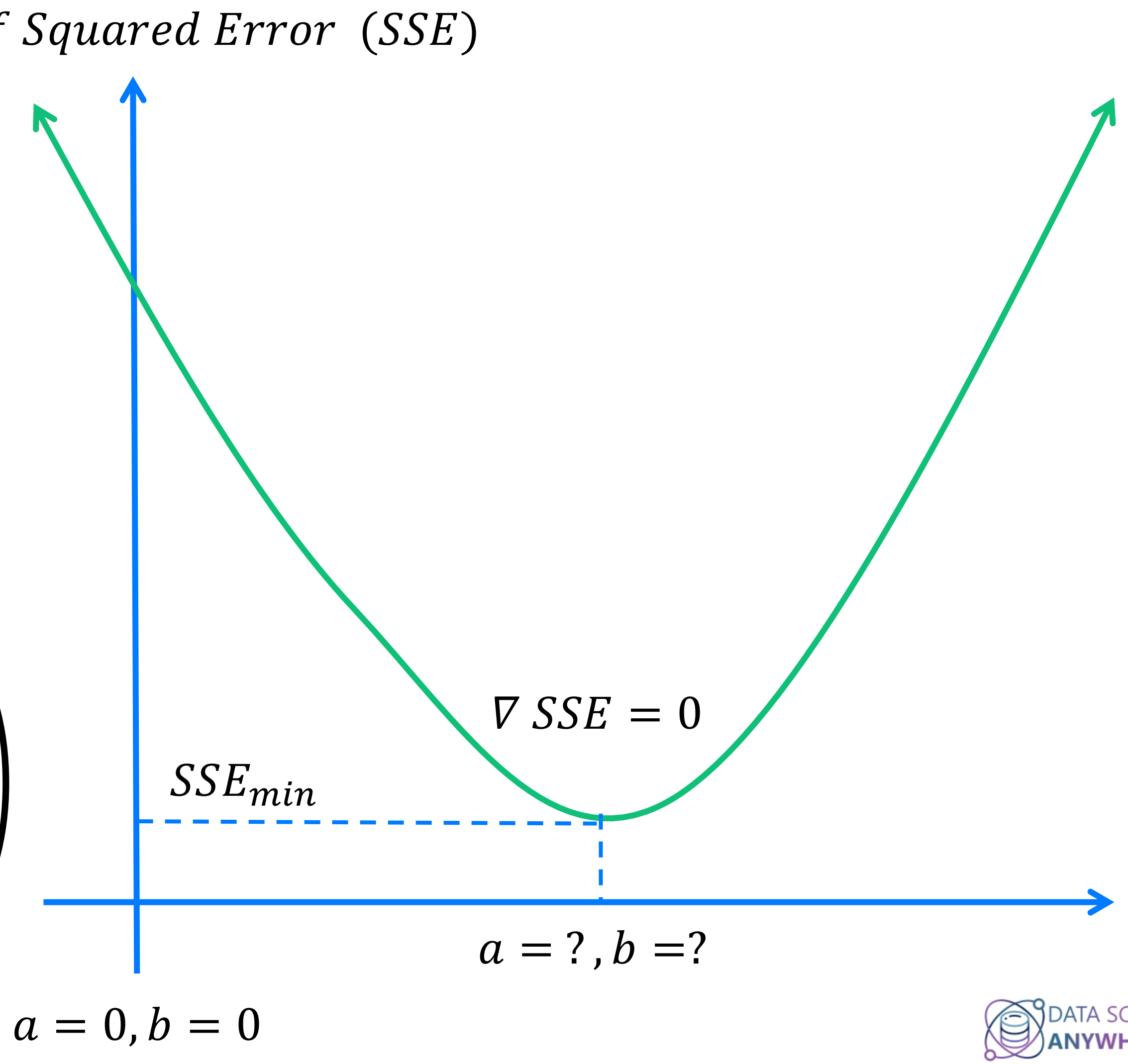
# Gradient Descent

TensorFlow 2.x

$$\boldsymbol{f(a,b) = cost = \sum_i (y_i - (a + b * X_i))^2}$$

$$\nabla f(a,b) = \begin{pmatrix} \frac{\partial\, cost}{\partial a} \\ \\ \frac{\partial\, cost}{\partial b} \end{pmatrix}$$

$$= \begin{pmatrix} \sum_i -2X_i\,(y_i - a - b * X_i) \\ \\ \sum_i -2\,(y_i - a - b * X_i \end{pmatrix}$$




DATA SCIENCE ANYWHERE

# Gradient Descent

$$\nabla\ f(a,b) = \begin{pmatrix} \dfrac{\partial\ cost}{\partial a} \\ \\ \dfrac{\partial\ cost}{\partial b} \end{pmatrix} \qquad = \begin{pmatrix} \sum_i -2X_i\ (y_i\ - a\ - b * X_i) \\ \\ \sum_i -2\ (y_i\ - a\ - b * X_i \end{pmatrix}$$

TensorFlow 2.x

# Gradient Descent

$$\nabla f(a,b) = \begin{pmatrix} \frac{\partial\, cost}{\partial b} \\ \\ \frac{\partial\, cost}{\partial a} \end{pmatrix} = \begin{pmatrix} \sum_i -2X_i\,(y_i - a - b * X_i) \\ \\ \sum_i -2\,(y_i - a - b * X_i \end{pmatrix}$$

$$a = a_{init} - \frac{\partial\, cost}{\partial a} * learning\_rate$$

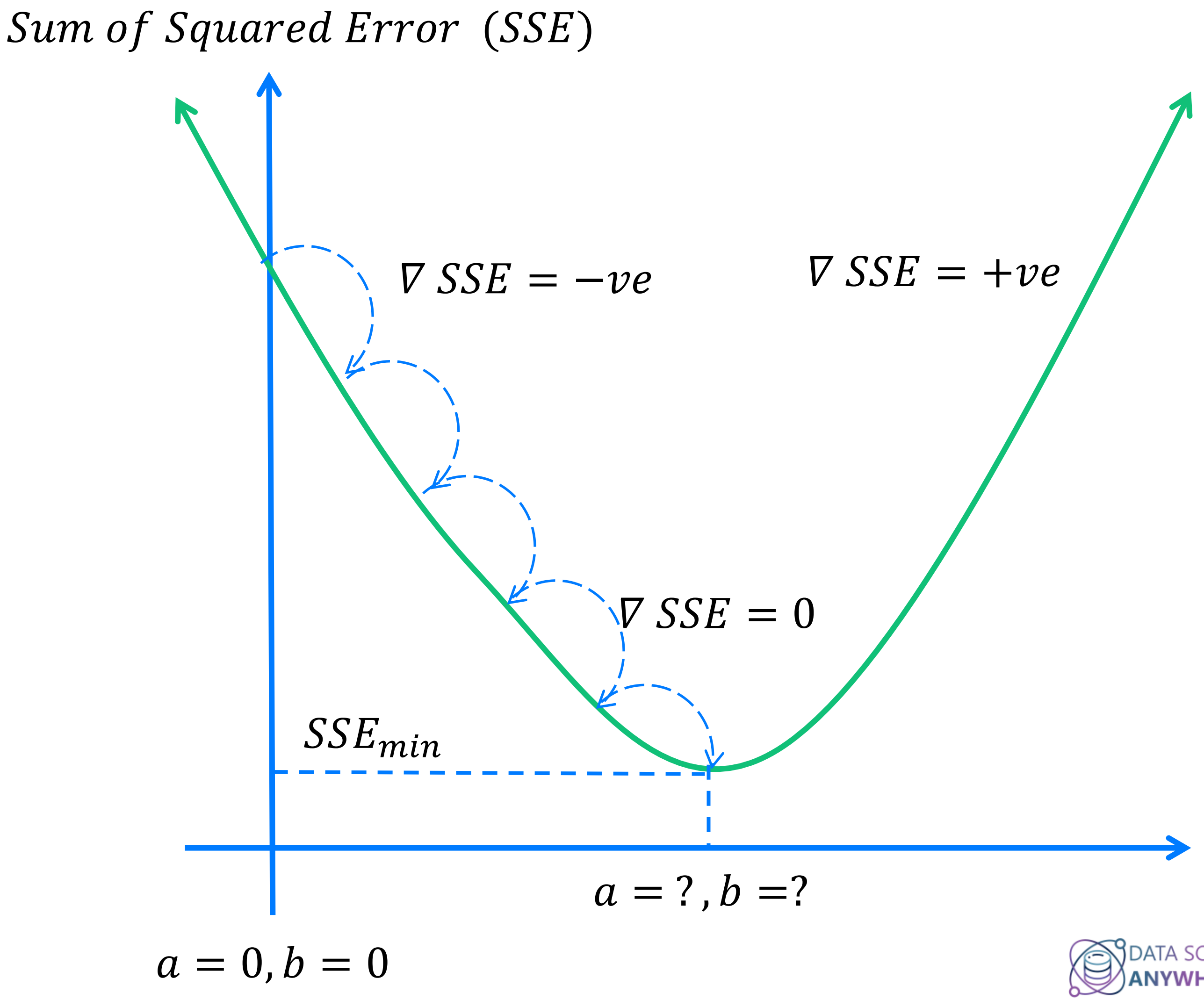
$$b = b_{init} - \frac{\partial\, cost}{\partial b} * learning\_rate$$

# Gradient Descent

TensorFlow 2.x

$$a = a_{init} - \frac{\partial\, cost}{\partial a} * learning\_rate$$

$$b = b_{init} - \frac{\partial\, cost}{\partial b} * learning\_rate$$




DATA SCIENCE ANYWHERE

# Learning Rate

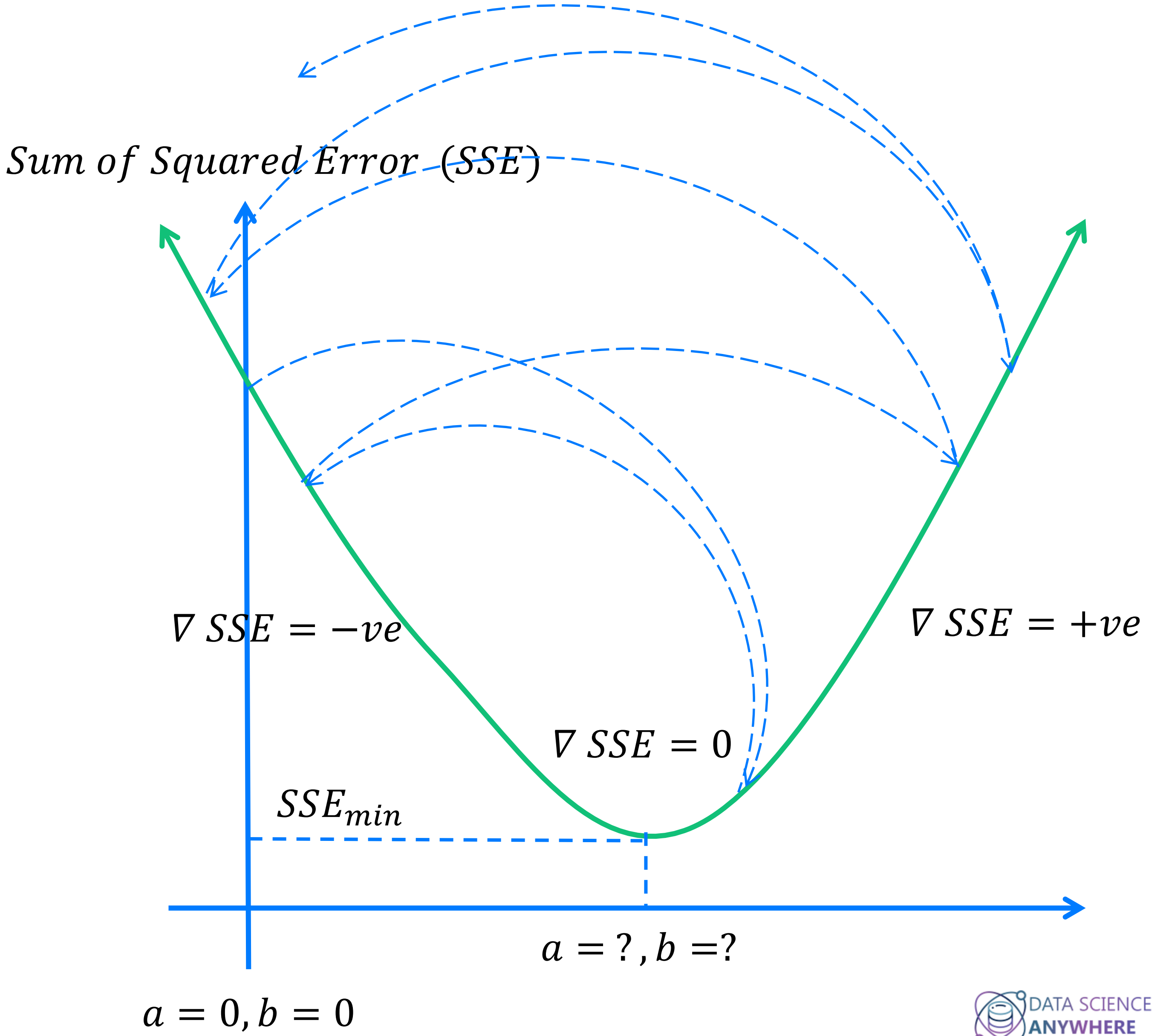


$$Learning\ Rate\ = (0\ -\ 1)$$

$$a = a_{init} - \frac{\partial\ cost}{\partial a}$$

$$b = b_{init} - \frac{\partial\ cost}{\partial b}$$

Without learning rate or for large learning rate, model will become **unstable.**



# Gradient Descent

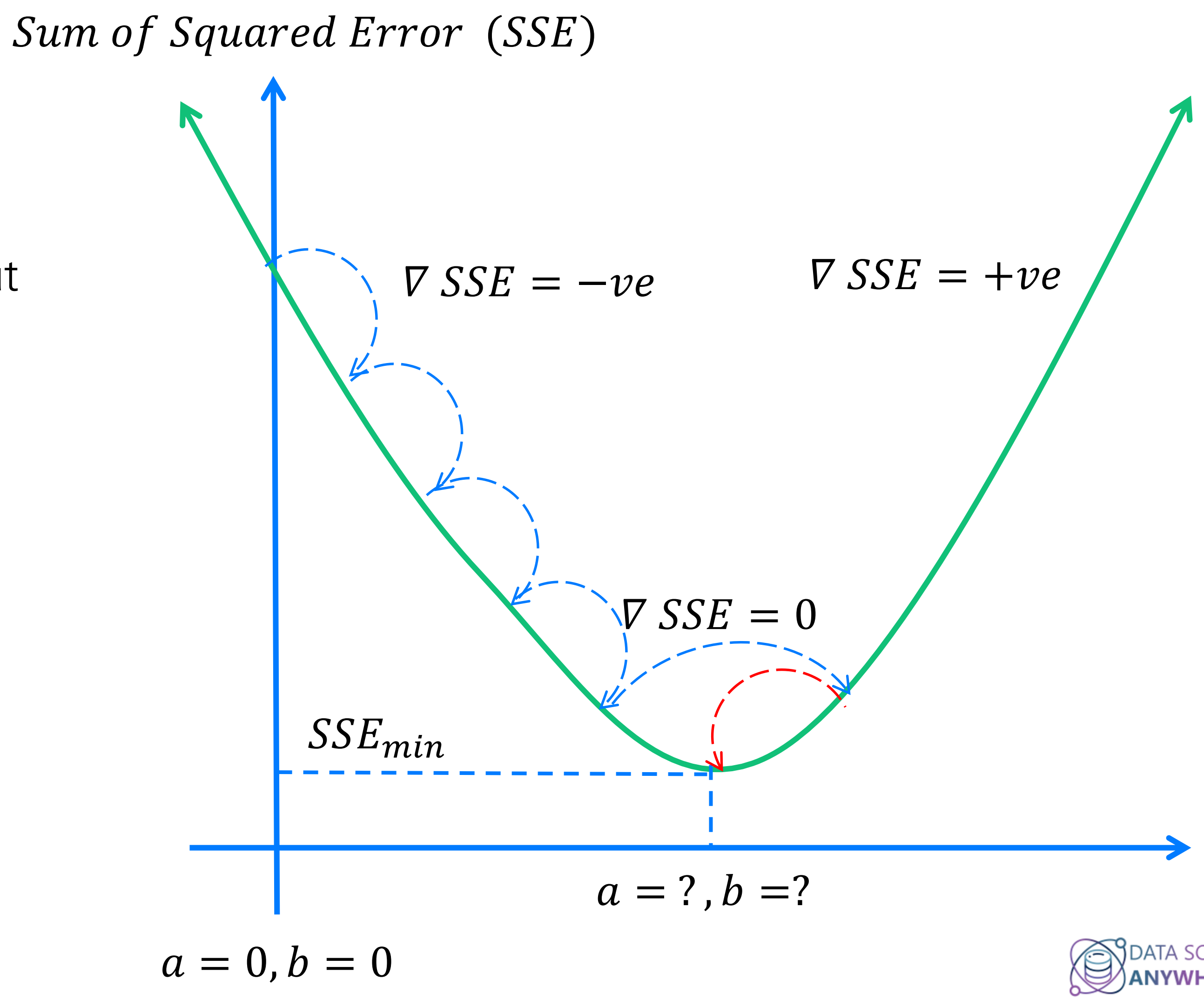
TensorFlow 2.x

$Learning\ Rate\ = (0\ -\ 1)$

- Small learning rate takes longer steps and converges grantee
- Large learning some time takes small steps but may diverge

$$a = a_{init} - \frac{\partial\ cost}{\partial a} * learning\_rate$$

$$b = b_{init} - \frac{\partial\ cost}{\partial b} * learning\_rate$$



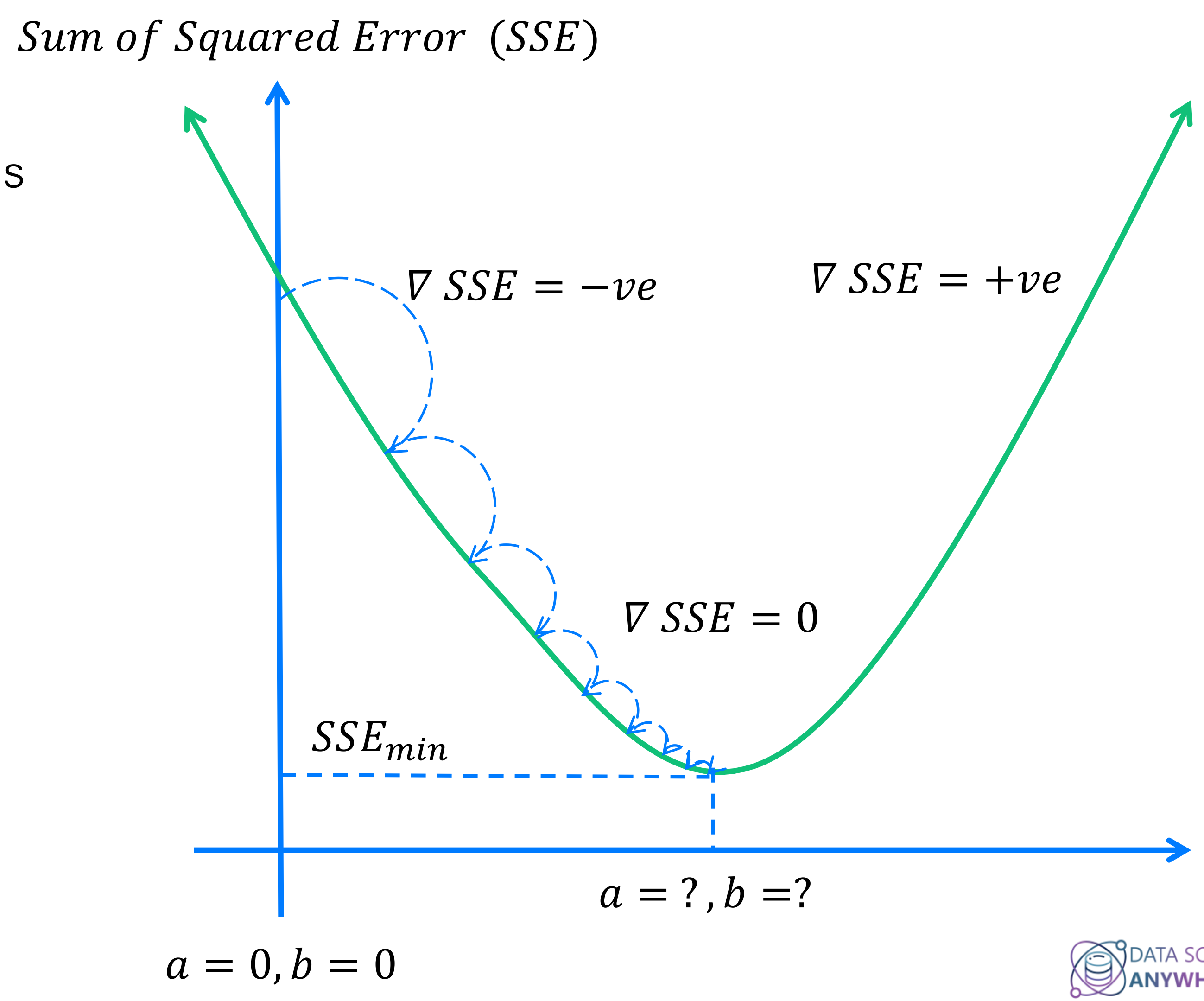

DATA SCIENCE ANYWHERE

# Adam Optimizer

TensorFlow 2.x

$Learning\ Rate\ = (0\ -\ 1)$

- Initially starts with large learning rate as epochs increase learning rate become smaller and smaller.
- Get optimal value with less epochs



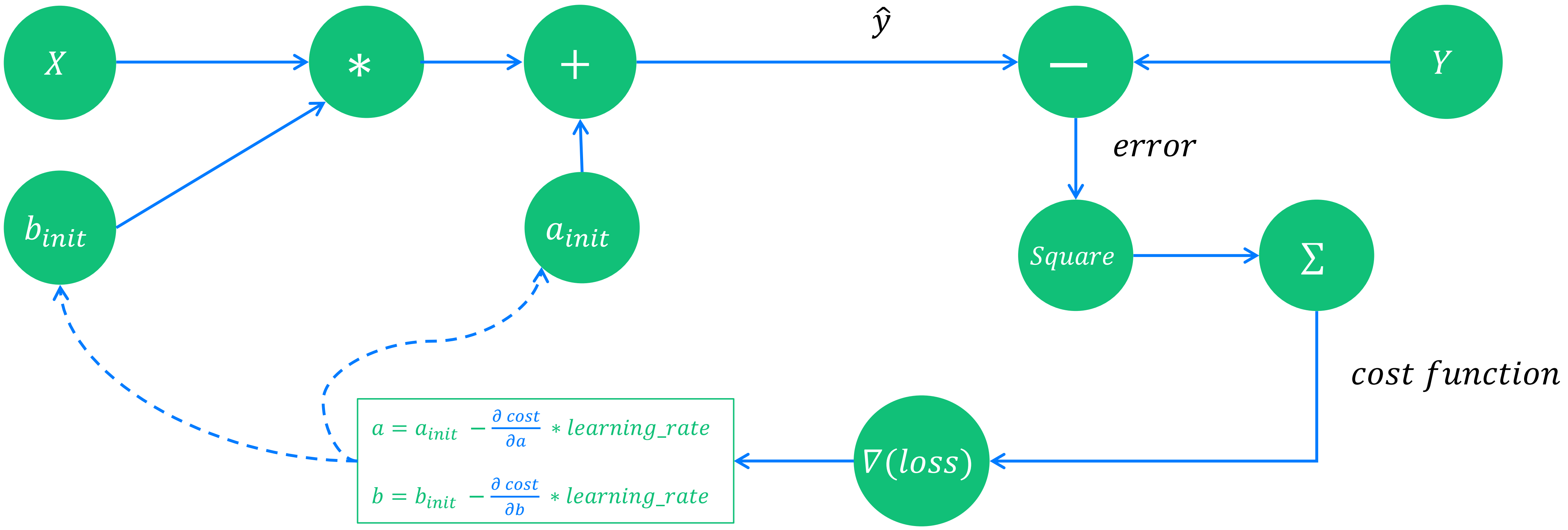
# Model Architecture

Reference: https://www.tensorflow.org/tutorials/customization/custom_training

# Linear Model Architecture (Neuron)

TensorFlow 2.x

$$\hat{y} = a + b * X$$

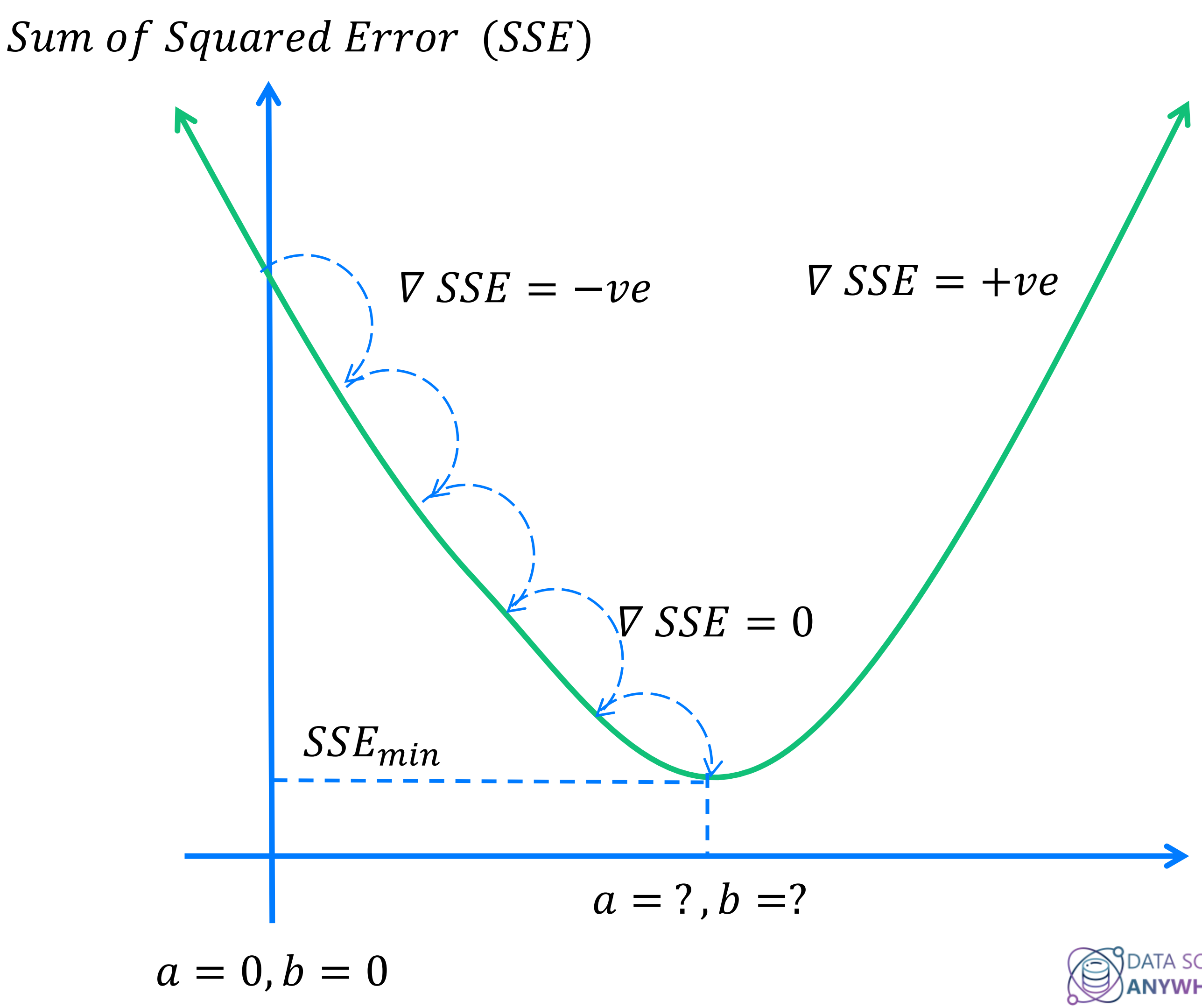


# Gradient Descent

TensorFlow 2.x

$$a = a_{init} - \frac{\partial\, cost}{\partial a} * learning\_rate$$

$$b = b_{init} - \frac{\partial\, cost}{\partial b} * learning\_rate$$




DATA SCIENCE ANYWHERE

**Srikanth**
GitHub: https://github.com/srikanthdakoju/custom-regression-training-tensorflow2
**Jupyter Notebook:** https://colab.research.google.com/drive/1i0nm4swR64QFBP2hPPl4Gn8GsPZR3f17?usp=sharing
website: http://www.datascienceanywhere.com