Loss & Cost Function & Gradient Descent Algorithm

28 July 2024 21:22

What is a cost function ? How is it different from loss function ?

Loss function

* A loss function, also known as error function, measures the difference between predicted value and actual value.

— Loss function is used when we refer the error for a single training example (a single row)

— Instance level – calculated for a single training example.

Regression: MSE : Mean Squared error:

$$L(y_i, \hat{y}_i) = (y_i - \hat{y}_i)^2$$

for some i=10

Classification: Cross-Entropy loss

Predicted Probabilities from logistic regression.

$$y_i \cdot log(p(y_i)) + (1 - y_i) \cdot log(1 - p(y_i))$$

actual

for some i=10

Cost Function

A cost function aggregates the losses over the entire training dataset or batch of $\underbrace{\text{training examples}}_{\text{training datapoints}}$

— Dataset level: summarizes the performance of the model over the entire training dataset

$h_\theta(x^{(i)})$: Predicted value : $\hat{y}$

$y^{(i)}$: Actual value : y

For Regression:

Mean Squared Error (MSE) Formula: $J(\theta) = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$

Mean Absolute Formula: $J(\theta) = \frac{1}{m} \sum_{i=1}^{m} |h_\theta(x^{(i)}) - y^{(i)}|$

cost functions for the entire dataset

Mean Absolute Error (MAE) Formula: $J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \left|h_\theta(x^{(i)}) - y^{(i)}\right|$

cost functions for the entire dataset

$|x|$: modulus function

$|0| \rightarrow 0$

For classification

$$y_i \cdot log(p(y_i)) + (1 - y_i) \cdot log(1 - p(y_i))$$

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left\{y^{(i)} \log\left(h_\theta(x^{(i)})\right) + \left(1-y^{(i)}\right)\log\left(1-h_\theta(x^{(i)})\right)\right\}$$: Binary Classification

Task why $\ominus$ve sign in the above equation?

Gradient Descent Algorithm (GDA)

slope fall down

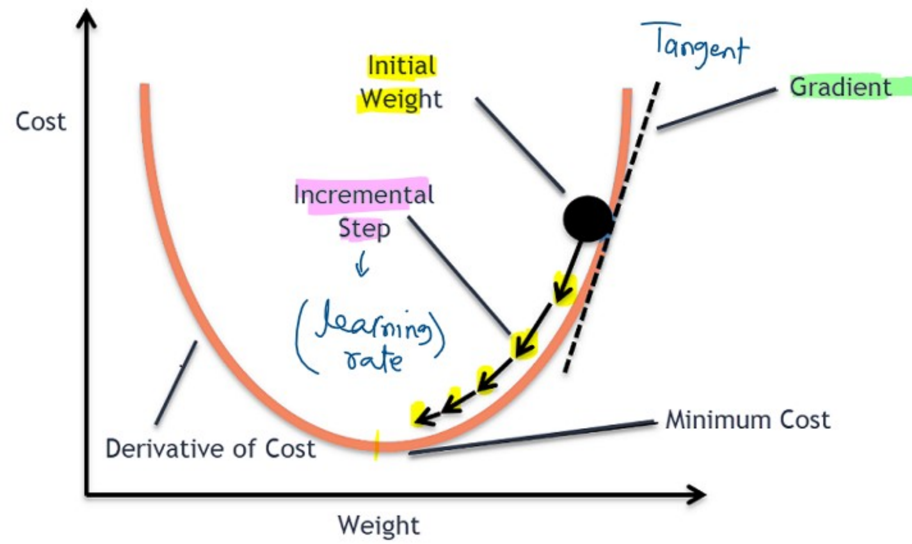
©abc

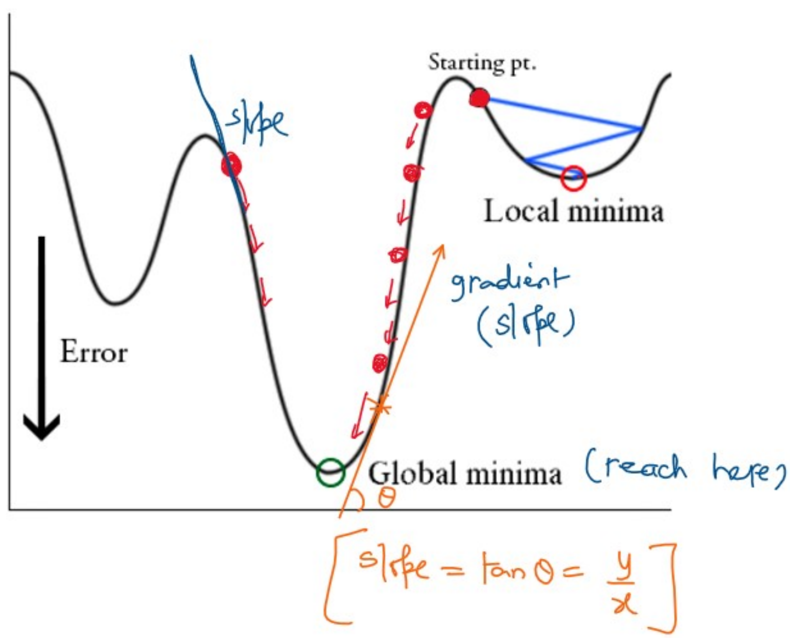
Cost Function

Follow the path of the steepest descent, taking steps in the direction that

slope (tangent)

in the direction that slope (tangent)
decreases the slope and brings Gradient
you closer to the land | plains.





GDA helps the model to find the optional set of parameters (weights and biases) by iteratively adjusting them in the opposite direction of the gradient.

GDA

repeat until convergence {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} \underbrace{\left[\overbrace{J(\theta_0, \theta_1)}^{\text{cost Function}} \right]}_{\text{gradient / slope for } (j=0 \text{ and } j=1)}$$

θ_j: initial value

α: learning rate

}

For $j=0$

$$\theta_0 := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} \left[J(\theta_0, \theta_1) \right]$$

$\theta_0 = \boxed{10}$

$\theta_1 = 9$

$\theta_2 = 8$

$\theta_3 = 7.5$

$\vdots$

θ_{100} 3

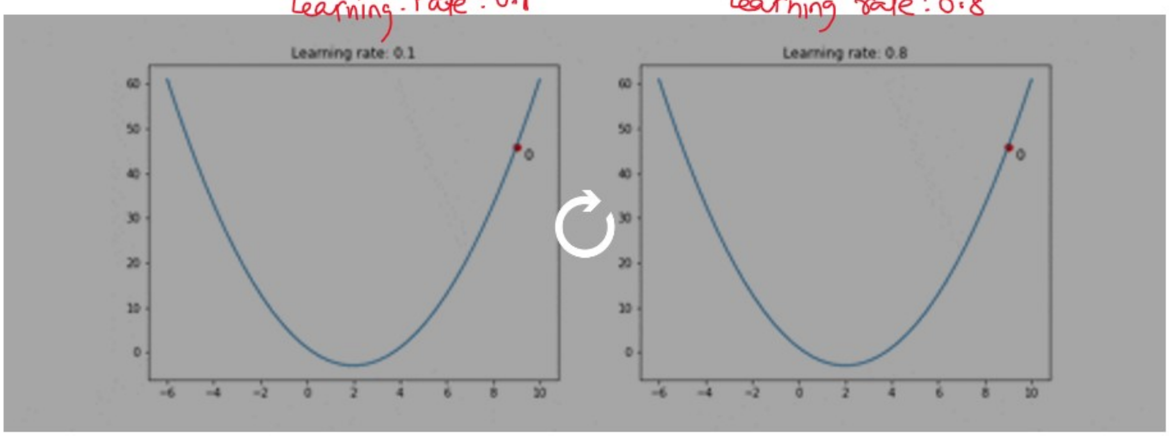
3

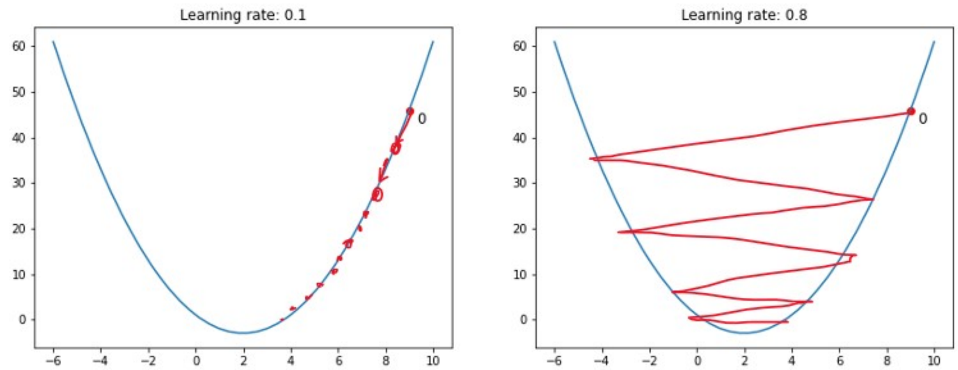
3

2

[Animated GIF - Find & Share on GIPHY]

Learning rate : 0.1 Learning rate : 0.8





$\theta_0)_{updated} \triangleq \theta_0)_{last} \longrightarrow$ GDA Converged

Note: If the learning rate is too slow ($\alpha = 0.001$), model takes quite a lot of time to converge (training time is unrealistically high.)

If the learning rate is too high (say $\alpha = 0.8$), model might overshoot the minima and keep bouncing without reaching the minima

$\alpha = 0.1$ to 0.2 : as per the best practices