

From Linear to Logistic Regression

Exploring the Gradient Descent Approach

Ibrahim Radwan

LINEAR REGRESSION

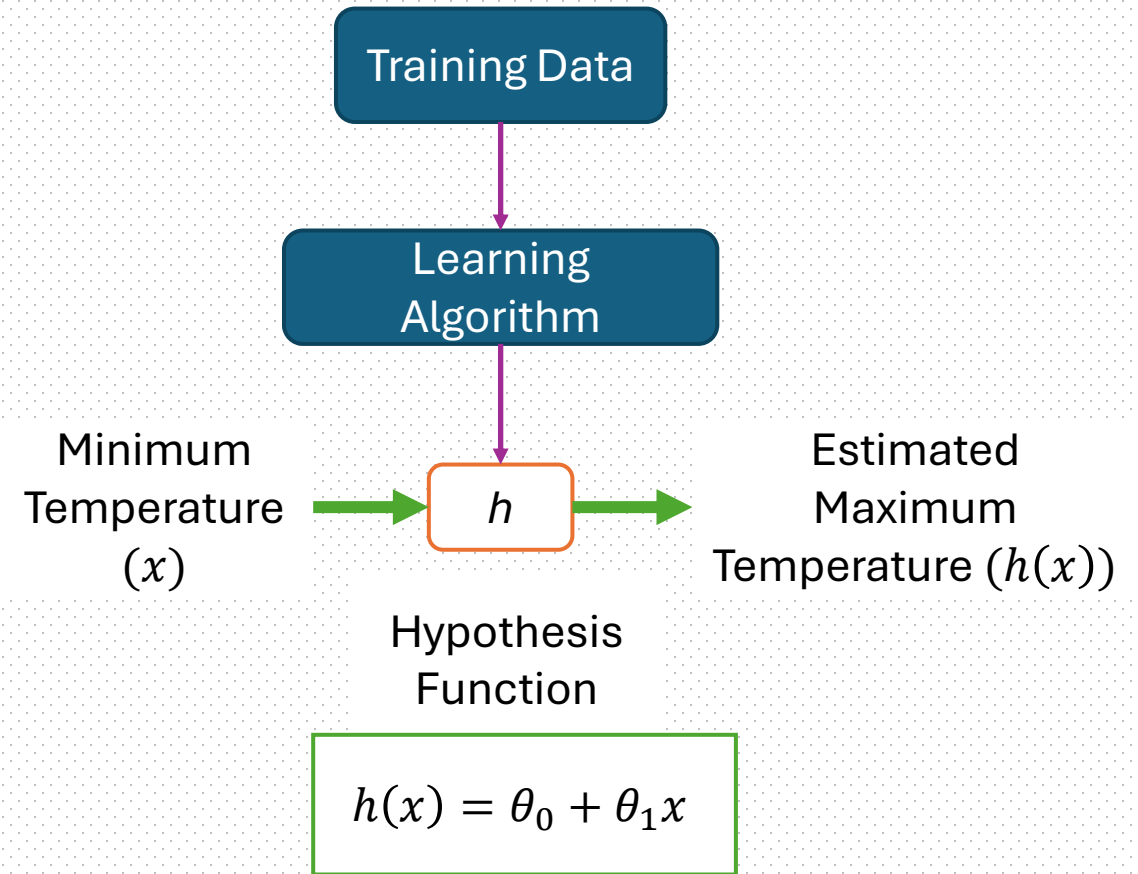
Notations:

m = Number of training examples.

$x^{(i)}$ = input variable/feature of i^{th} example

$y^{(i)}$ = output variable/target of i^{th} example

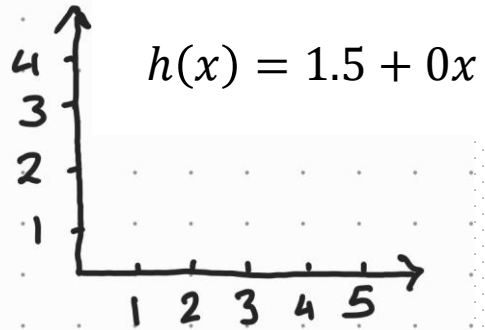
How to choose parameters
 θ_0 and θ_1



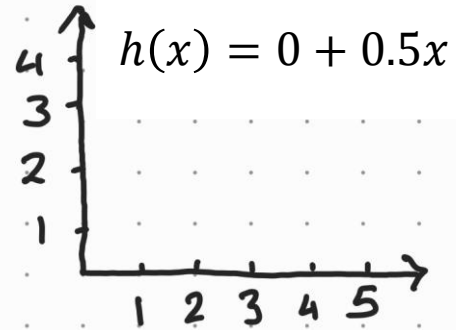
COST FUNCTION

$$h(x) = \theta_0 + \theta_1 x$$

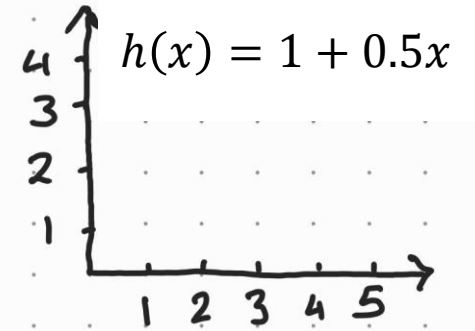
Hypothesis
Function



$$\begin{aligned}\theta_0 &= 1.5 \\ \theta_1 &= 0\end{aligned}$$



$$\begin{aligned}\theta_0 &= 0 \\ \theta_1 &= 0.5\end{aligned}$$

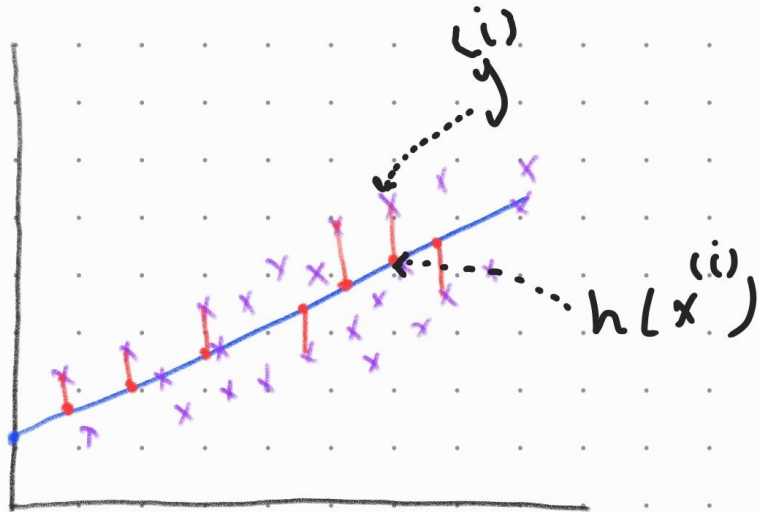


$$\begin{aligned}\theta_0 &= 1 \\ \theta_1 &= 0.5\end{aligned}$$

Idea: Choose θ_0, θ_1 so that $h(x)$ is close to y for the training pairs $(x^{(i)}, y^{(i)})$

COST FUNCTION

How and which cost function?



Minimize θ_0, θ_1 $\frac{1}{2m} \sum_{i=1}^m (h(x^{(i)}) - y^{(i)})^2$

where: $h(x^{(i)}) = \theta_0 + \theta_1 x^{(i)}$

Squared Error Function

or

Loss Function $J(\theta_0, \theta_1)$

LINEAR REGRESSION

Hypothesis function:

$$h(x) = \theta_0 + \theta_1 x$$

Parameters:

$$\theta_0, \theta_1$$

Cost Function:

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^m (h(x^{(i)}) - y^{(i)})^2$$

How to obtain best parameters?

$$\underset{\theta_0, \theta_1}{\text{minimize}} J(\theta_0, \theta_1)$$

Simplified Version

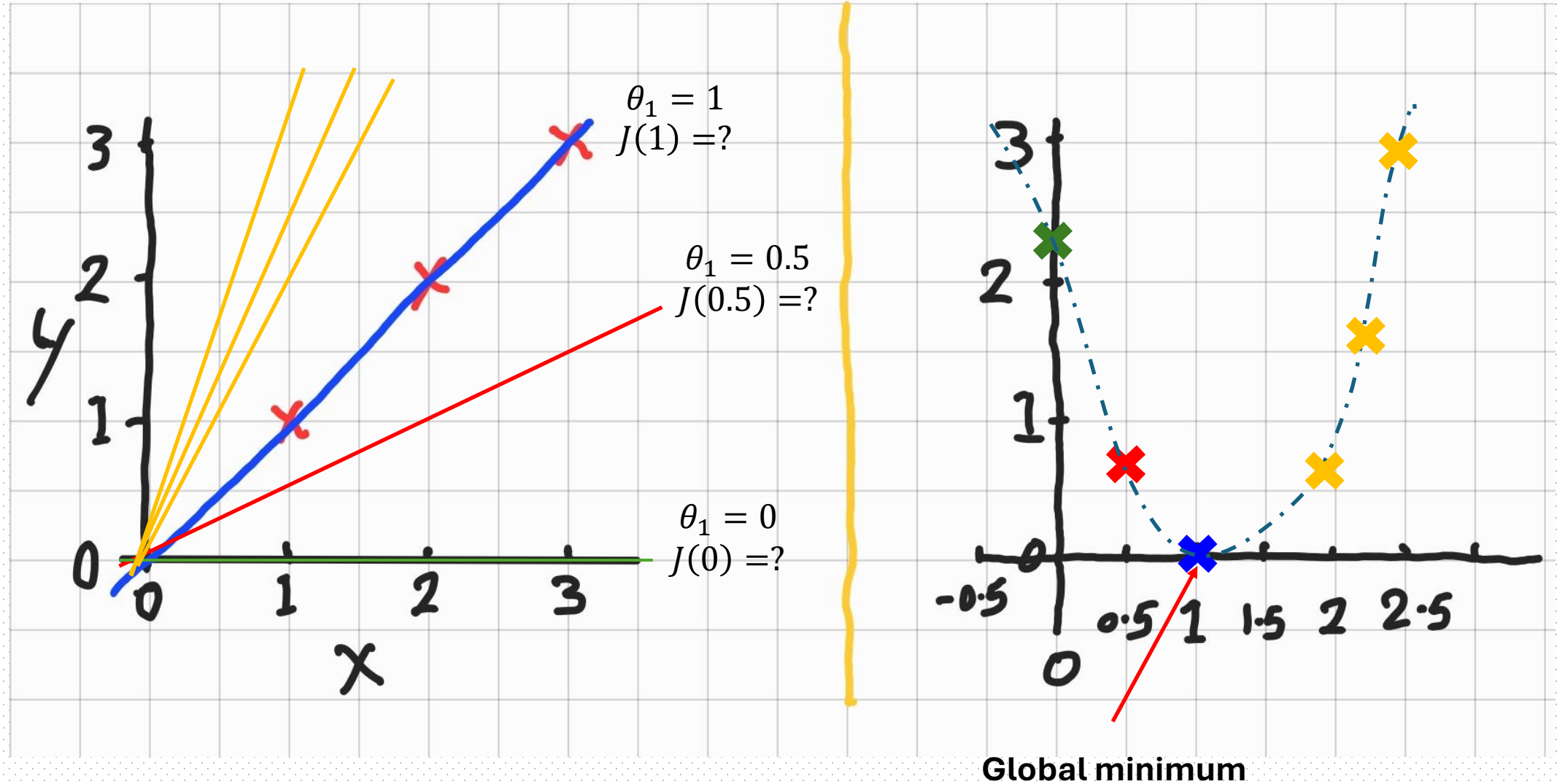
$$h(x) = \theta_1 x$$

$$\theta_1$$

$$J(\theta_1) = \frac{1}{2m} \sum_{i=1}^m (h(x^{(i)}) - y^{(i)})^2$$

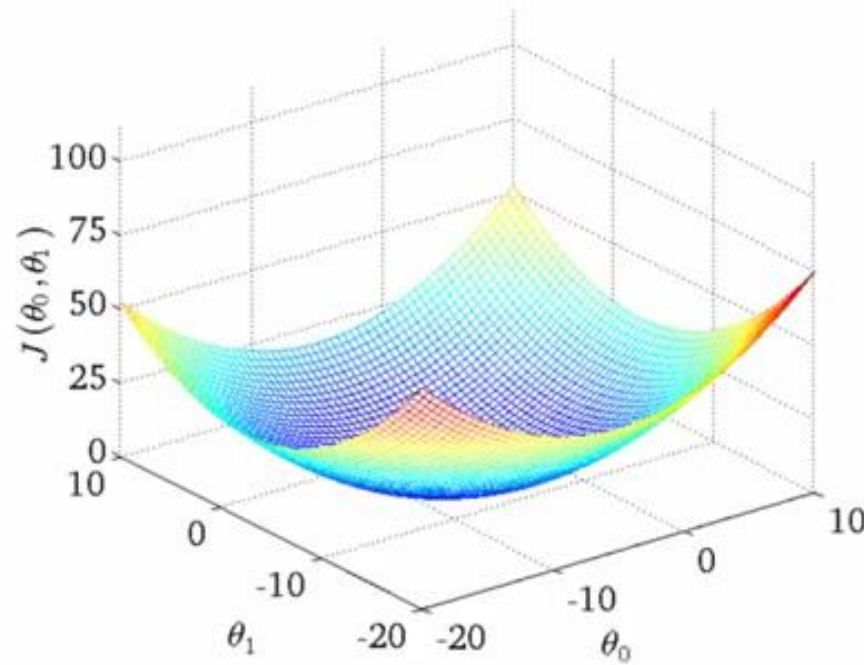
$$\underset{\theta_1}{\text{minimize}} J(\theta_1)$$

COST FUNCTION



COST FUNCTION

How about if we plot it for the two parameters?



Bowl-shaped Function

Convex Function

Image Source : Coursera

GRADIENT DESCENT

Imagine that you are standing at the top of a mountain. You have been asked to reach the valley down the mountain, which you do not know where it is. What would you do? Since you are unsure of where and in which direction you need to move to reach that valley, you would probably take little baby steps in the direction down and try to figure out if that path leads to your destination. You would repeat this process until you reach the ground. This is exactly how Gradient Descent algorithm works.

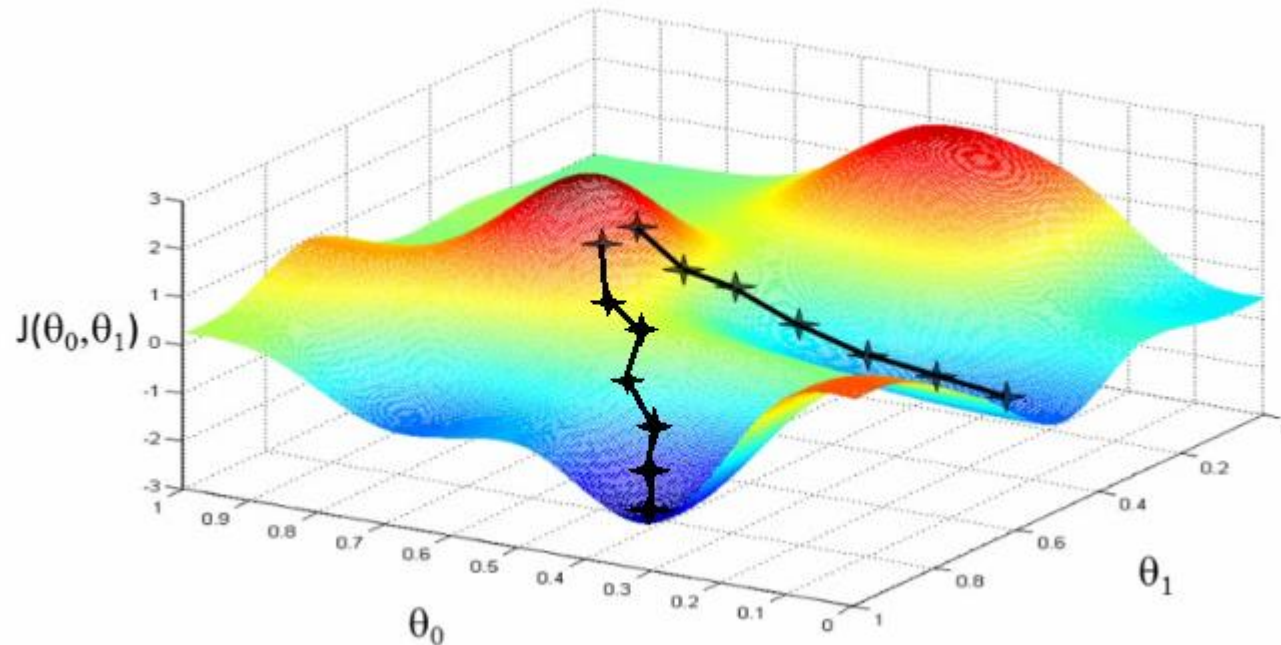


Image Source : Coursera

GRADIENT DESCENT

Gradient descent algorithm:

repeat until convergence {
$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$
(for $j = 1$ and $j = 0$)
}

Linear regression

Hypothesis function:

$$h(x) = \theta_0 + \theta_1 x$$

Parameters:

$$\theta_0, \theta_1$$

Cost Function:

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^m (h(x^{(i)}) - y^{(i)})^2$$

How to obtain best parameters?

$$\underset{\theta_0, \theta_1}{\text{minimize}} J(\theta_0, \theta_1)$$

GRADIENT DESCENT + LINEAR REG.

Gradient descent algorithm becomes:

repeat until convergence {

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^m ((h(x^{(i)}) - y^{(i)})$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^m ((h(x^{(i)}) - y^{(i)}) \cdot x^{(i)}$$

}

α : Learning rate, which controls the speed of the training process.

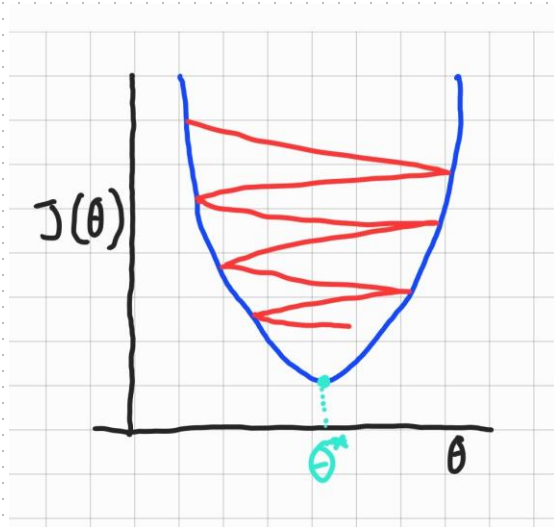
θ_0, θ_1 are required to update simultaneously.

Start the training with some initial weights (θ_0, θ_1) and the gradient descent will keep updating these weights until it finds the optimal value for these weights which result in minimizing the cost function.

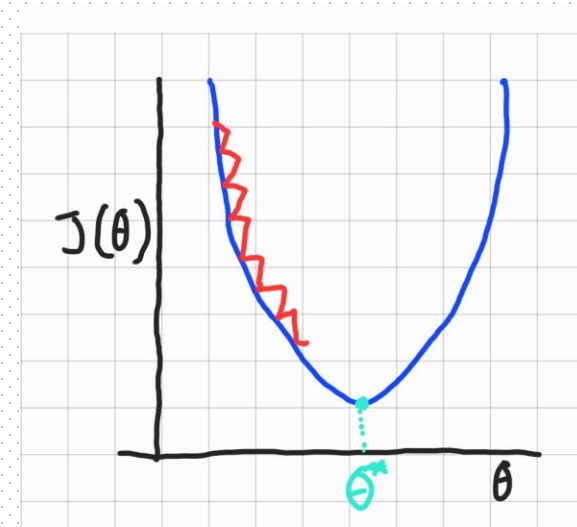
With the Linear Regression, the cost function is always a convex function (bowl-shaped), which means it will always converge to a global minimum.

GRADIENT DESCENT + LINEAR REG.

But how about the learning rate?



Too big: This results in overshooting the minima and most likely to miss the global minimum

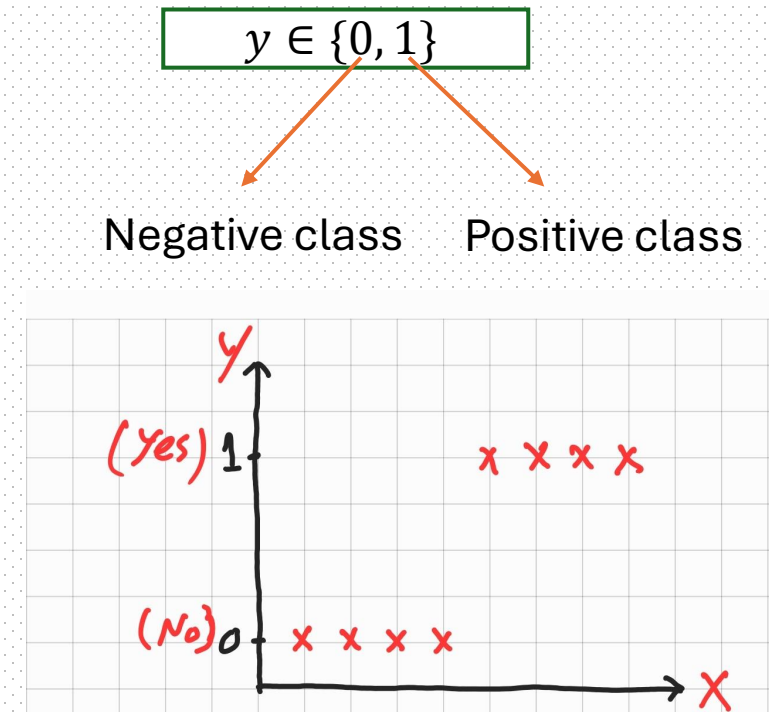


Too small: This results in a slow convergence

Implementing gradient descent for linear regression

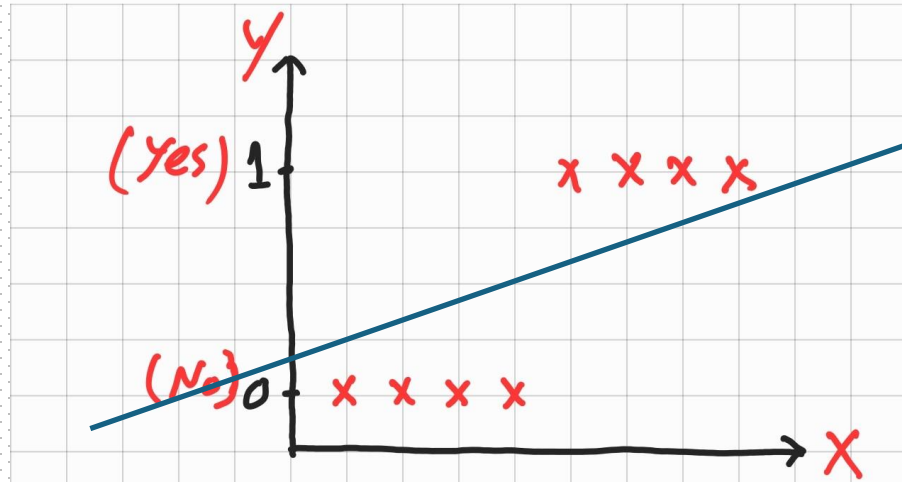
CLASSIFICATION

- Email: Spam / Not Spam?
- Online Transactions: Fraudulent (Yes / No)?
- Tumor: Malignant / Benign?
- Product rate: Like / Dislike?
- Product quality: High / Low?
- ...



CLASSIFICATION

- Can we use the linear regression for classification?



Idea 1: Thresholding the predicted values against a specific threshold. For example, any value below 0.5 is zero, otherwise is one.

Issues:

- This solution is inefficient as it violates the fact that for the binary classification the values of y is either 0 or 1.
- $h(x)$ can be any real-value between $-\infty$ and ∞ for the linear regression.

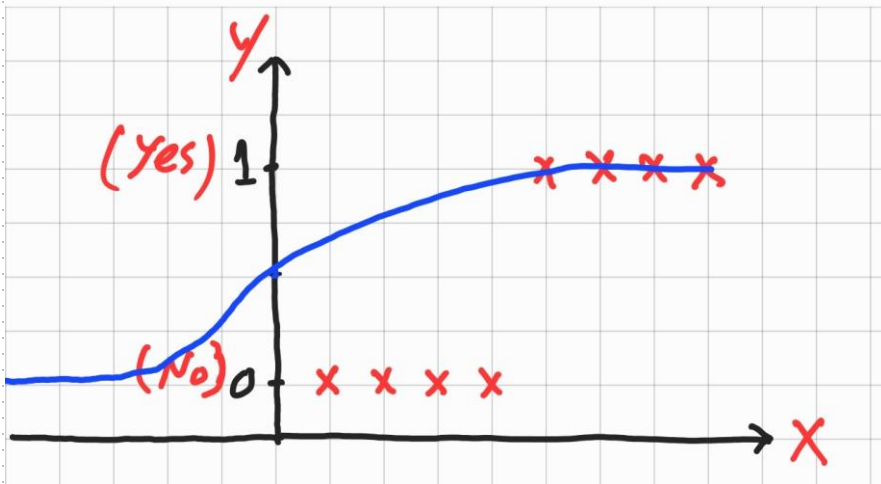
Idea 2: How about we predict $h(x)$ and apply another function g to put the values into the $[0, 1]$ range.

So, what is this function g ?

LOGISTIC REGRESSION

- g is the logistic function or sometimes can be referred as the Sigmoid function, which turns the values of the hypothesis $h(x)$ to be between 0 and 1.

Sigmoid/Logistic function $g(z) = \frac{1}{1 + e^{-z}}$



Logistic regression

Hypothesis function:

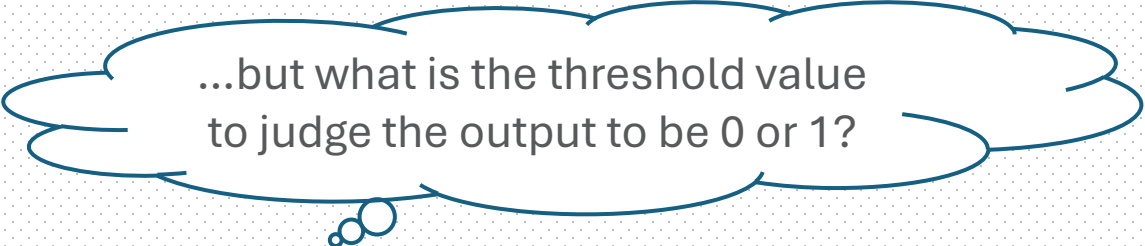
$$\begin{aligned} h(x) &= g(\theta_0 + \theta_1 x) \\ &= g(\theta^T x) \\ &= \frac{1}{1 + e^{-\theta^T x}} \end{aligned}$$

Parameters:

$$\theta^T$$

LOGISTIC REGRESSION

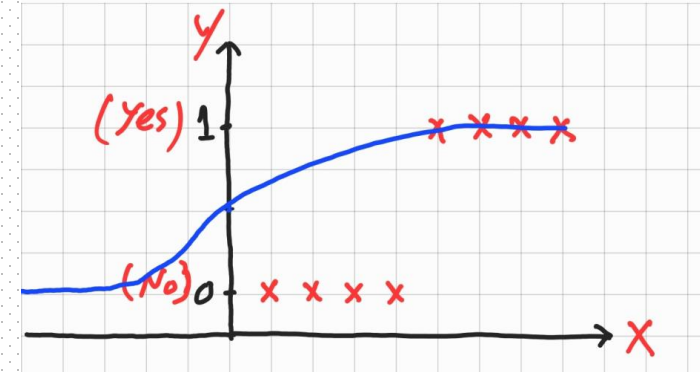
- In the context of the binary classification (aka, two classes problem), this means $h(x)$ will give a value between 0 and 1, which represents the probability that this class is either 0 or 1
- Basically, the two class probabilities are complementary to each other
- This means,
 - $h(x) = P(y = 1 | x; \theta) = 1 - P(y = 0 | x; \theta)$
 - $P(y = 1 | x; \theta) + P(y = 0 | x; \theta) = 1$



...but what is the threshold value to judge the output to be 0 or 1?

DECISION BOUNDARY

- This threshold defines the decision boundary
- The way our logistic function g behaves is that when its input is greater than or equal to zero, its output is greater than or equal to 0.5



$$g(z) > 0.5, \\ \text{when } z > 0$$

Remember:

$$\begin{aligned} z = 0, e^0 = 1 &\Rightarrow g(z) = 1/2 = 0.5 \\ z \rightarrow \infty, e^\infty \rightarrow \infty &\Rightarrow g(z) = 1 \\ z \rightarrow -\infty, e^\infty \rightarrow \infty &\Rightarrow g(z) = 0 \end{aligned}$$

That means:
 $g(\theta^T x) > 0.5$
when $\theta^T x > 0$

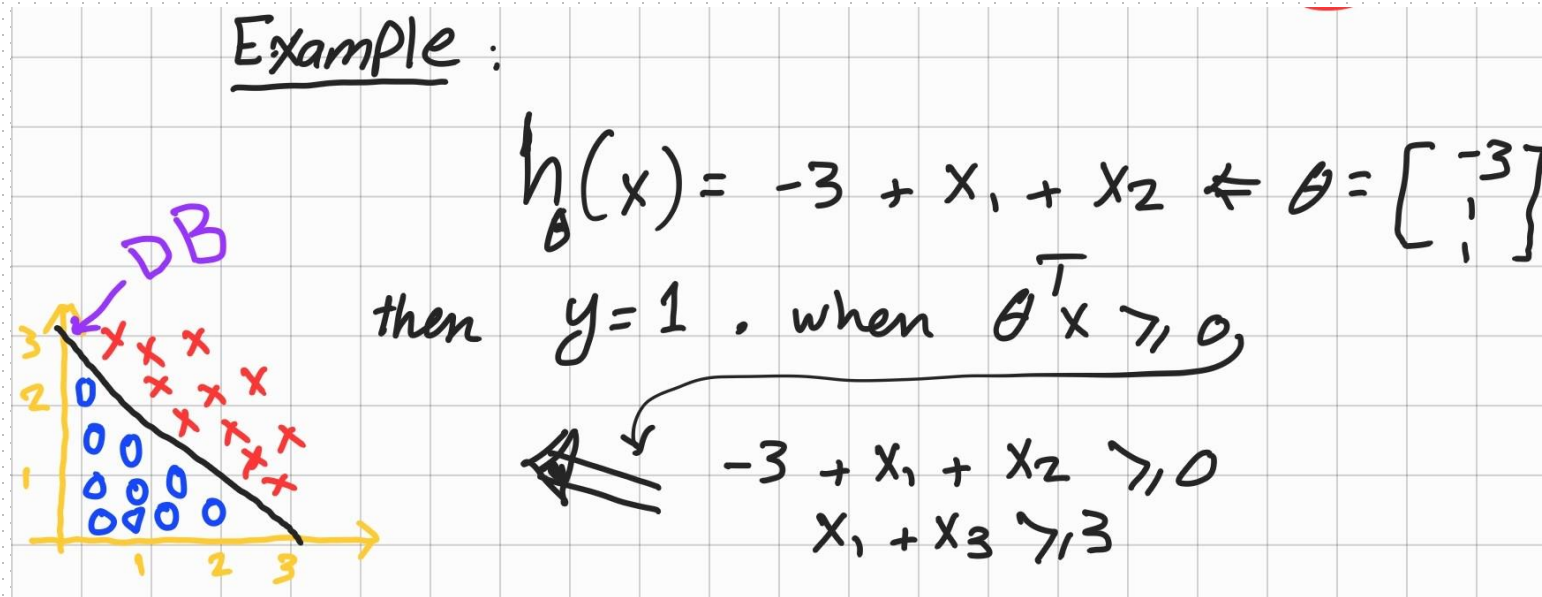


This leads to:

$$\left. \begin{aligned} \theta^T x > 0 &\Rightarrow y = 1 \\ \theta^T x < 0 &\Rightarrow y = 0 \end{aligned} \right\} \Leftarrow$$

The decision boundary
is created by
the hypothesis
function

DECISION BOUNDARY



However, the input to the logistic function doesn't need to be linear; it could be non-linear, such as a function that describes a circle or any shape that fits the data.

COST FUNCTION

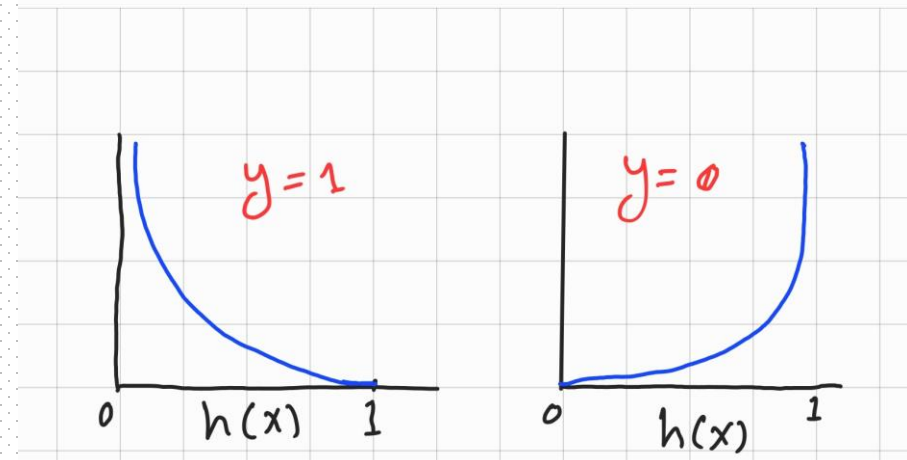
Analogy

- Given training examples X and y , the difference between the $h(x^{(i)})$ and $y^{(i)}$ represents the cost function.
- Can not we use the sum squared differences as our cost function as in the linear regression?
- Using the sum squared will lead to a non-convex function as the $h(x)$ is non-convex.
- This leads to the risk of local-minima
- To avoid this, we need to think of a new cost function that fits this problem and guarantees the global minimum existence.

Cost Function

$$\frac{1}{m} \sum_{i=1}^m \text{cost}(h(x^{(i)}), y^{(i)})$$

$$\text{cost}(h(x), y) = \begin{cases} -\log(h(x)), & y = 1 \\ -\log(1 - h(x)), & y = 0 \end{cases}$$



LOGISTIC REGRESSION

Hypothesis function:

$$h(x) = \frac{1}{1 + e^{-\theta^T x}}$$

Parameters:

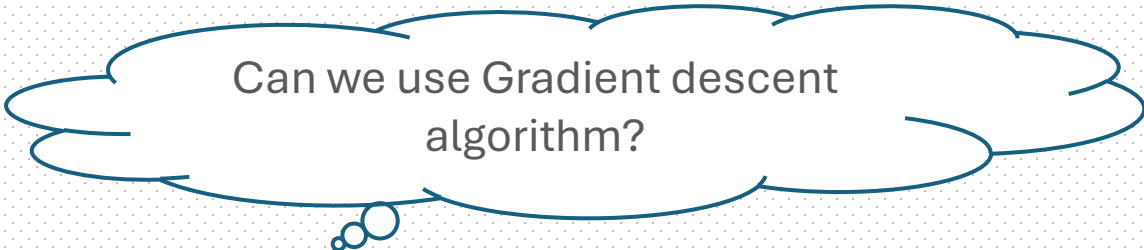
$$\theta^T$$

Cost Function:

$$J(\theta^T) = -\frac{1}{m} \sum_{i=1}^m [y^{(i)} \log(h(x^{(i)})) + (1 - y^{(i)}) \log(1 - h(x^{(i)}))]$$

How to obtain best parameters?

$$\underset{\theta_0, \theta_1}{\text{minimize}} J(\theta^T)$$



Can we use Gradient descent algorithm?

GRADIENT DESCENT + LOGISTIC REG.

Gradient descent algorithm becomes:

$$\begin{aligned} & \text{repeat until convergence} \{ \\ & \quad \theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta) \\ & \quad := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^m ((h(x^{(i)}) - y^{(i)}) \cdot x^{(i)}) \\ & \} \end{aligned}$$

α : Learning rate, which controls the speed of the training process.

θ^T are required to update simultaneously.

Start the training with some initial weights ($\theta_0, \theta_1, \theta_2, \dots$) and the gradient descent will keep updating these weights until it finds the optimal values for these weights which result in minimizing the cost function.

With the logistic Regression, the logistic cost function is always a convex function (bowl-shaped), which means it will always converge to a global minimum.

Implementing gradient descent for logistic regression

CASE STUDY

Import the data

Explore the data

Build a predictive model

Evaluate the model on the testing data

Interpret the results

DATA

- In this example we will be working on the following dataset:
 - <https://rdrr.io/cran/rattle.data/man/weatherAUS.html>
- First, we need to read the description of the data and understand the column names and formats

FULL CODE

- The full code of this use case can be downloaded from:
 - [Link](#)