Applied Machine Learning

BITS Pilani
Pilani Campus

Dr. Harikrishnan N B
Computer Science and Information Systems

innovate achieve lead

BITS Pilani

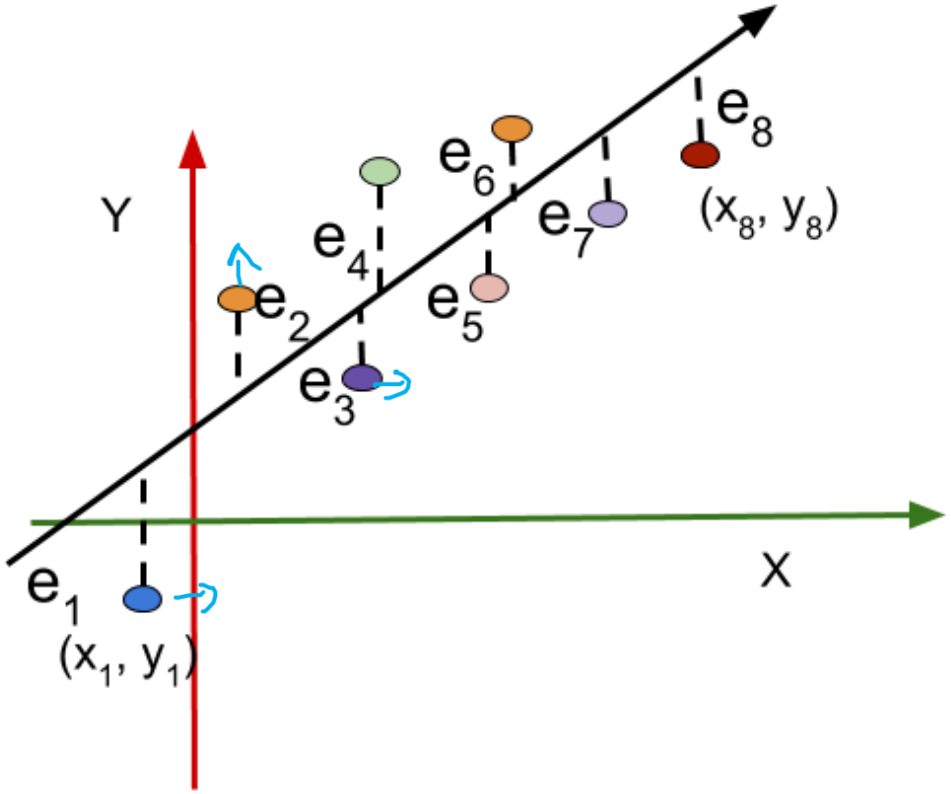
Pilani Campus

SE ZG568 / SS ZG568, Applied Machine Learning Lecture No. 10

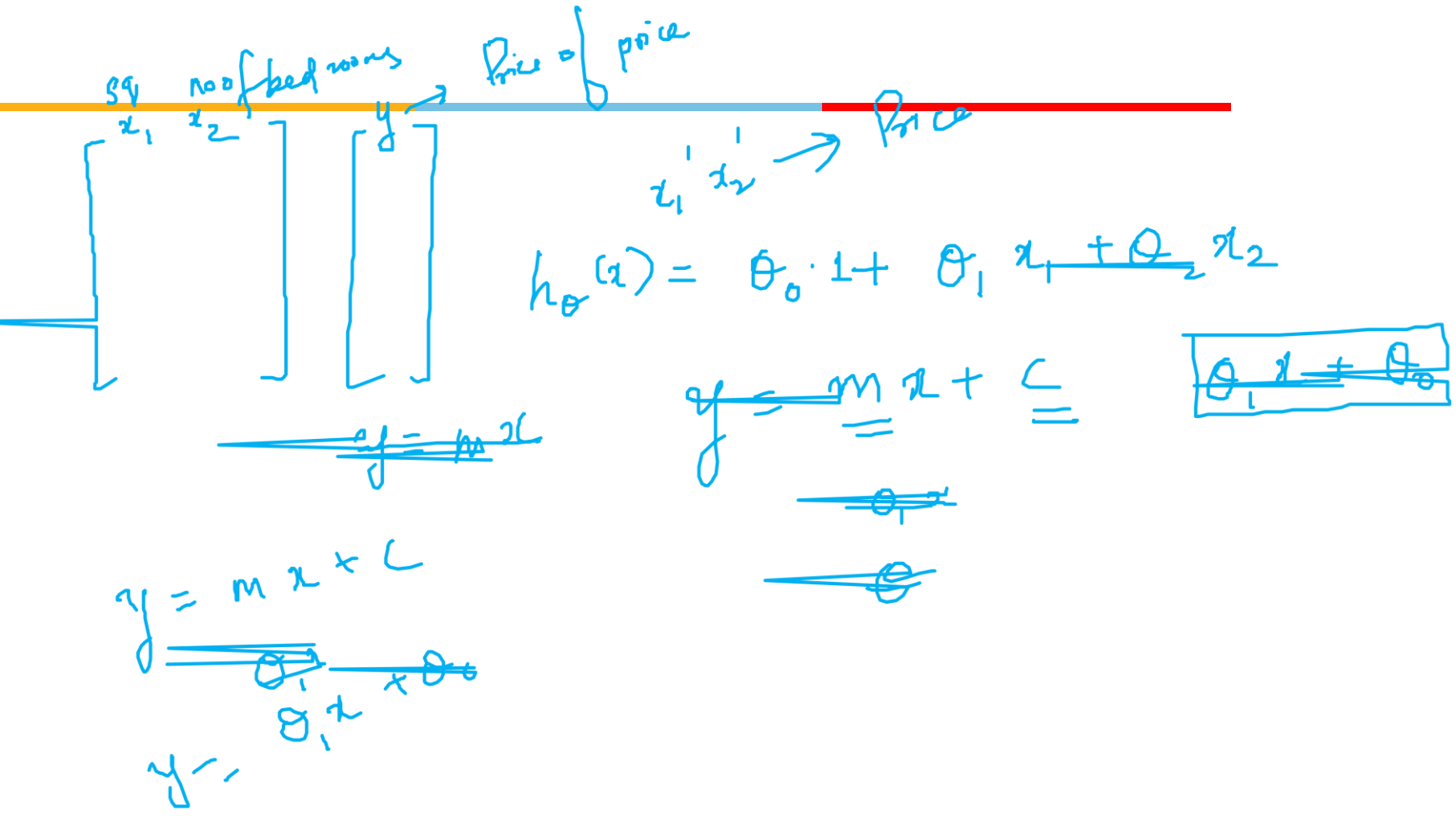
Linear Regression using Optimization

Logistic Regression

Linear Least Square Regression



$$y_1 = mx_1 + c + e_1$$
$$y_2 = mx_2 + c + e_2$$
$$y_3 = mx_3 + c + e_3$$
$$y_4 = mx_4 + c + e_4$$
$$y_5 = mx_5 + c + e_5$$
$$y_6 = mx_6 + c + e_6$$
$$y_7 = mx_7 + c + e_7$$
$$y_8 = mx_8 + c + e_8$$


Sq
No of bedrooms
Price of pice
Price
$h_\theta(x) = \theta_0 \cdot 1 + \theta_1 x_1 + \theta_2 x_2$
$y = mx + c$
$y = mx + c$

$$h_\theta(x) = \theta_0 \cdot 1 + \theta_1 x_1 + \theta_2 x_2 + \cdots \theta_n x_n$$

$$\text{error} = \left(h_\theta(x) - y\right)^2$$

$x^{(2)}$ ⇒ second House

$x_2^{(2)}$

$x_1^{(2)}$

Training data = m

Dataset

	x_1	x_2	y
1	House 1		
→	House 2		
⋮	⋮		
m	House m		

$x^{(i)}$ } i^{th} training data

i^{th} training data

$x^{(i)} = \begin{bmatrix} x_1^{(i)} & x_2^{(i)} \end{bmatrix}$

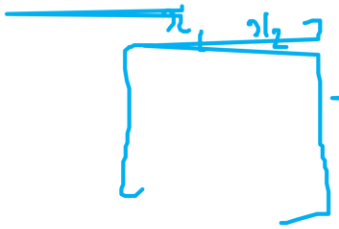
$x_1^{(i)}$ → feature

$x_2^{(i)}$

$$\text{Error} = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

$$= \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

$$h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 + \ldots + \theta_n x_n$$



$$h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 \qquad x_0 = 1$$

$$\theta_0 \cdot x_0 + \theta_1 x_1 + \theta_2 x_2$$

$$[\theta_0 \; \theta_1 \; \theta_2] \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} \Rightarrow \theta^T x$$

$$Error = \frac{1}{2m} \sum_{i=1}^{m} \left(\theta^T x^{(i)} - y^{(i)} \right)^2$$

Gradient Descent

A (function)

B (level set)

$f(x_1, x_2) = x_1^2 + x_2^2$

$x_1^2 + x_2^2 = 5$

$x_1 = 0, x_2 = 1$

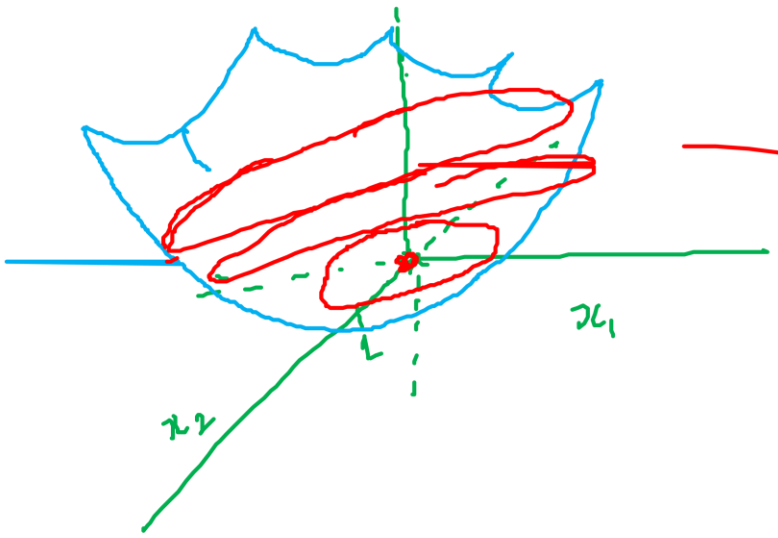
$f(x_1, x_2) = 1$

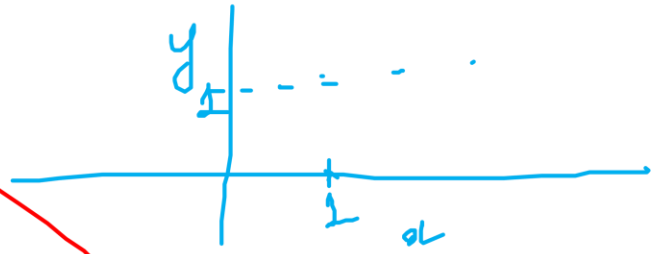
$f(x)$

$y =$ ~~max~~ $<$

$m = 1, c = 0$

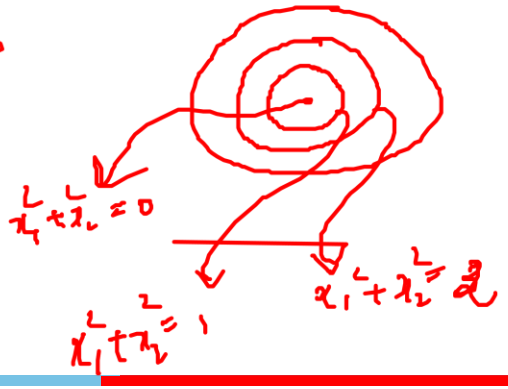
$x = 1$





$x_1^2 + x_2^2 = 0$

$= 1$



Linear regression models the relationship between the input x and the output y as a linear combination of the features:

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$$

where:

- $x = [1, x_1, x_2, ..., x_n]^T$ (including $x_0 = 1$ for bias)
- $\theta = [\theta_0, \theta_1, ..., \theta_n]^T$ are the parameters (weights)
- $h_\theta(x)$ is the predicted output.

In vector form:

$$h_\theta(x) = \theta^T x$$

$\theta_{new} = \theta_{old} - \alpha \frac{\partial Error}{\partial \theta_{old}}$

θ_{new}

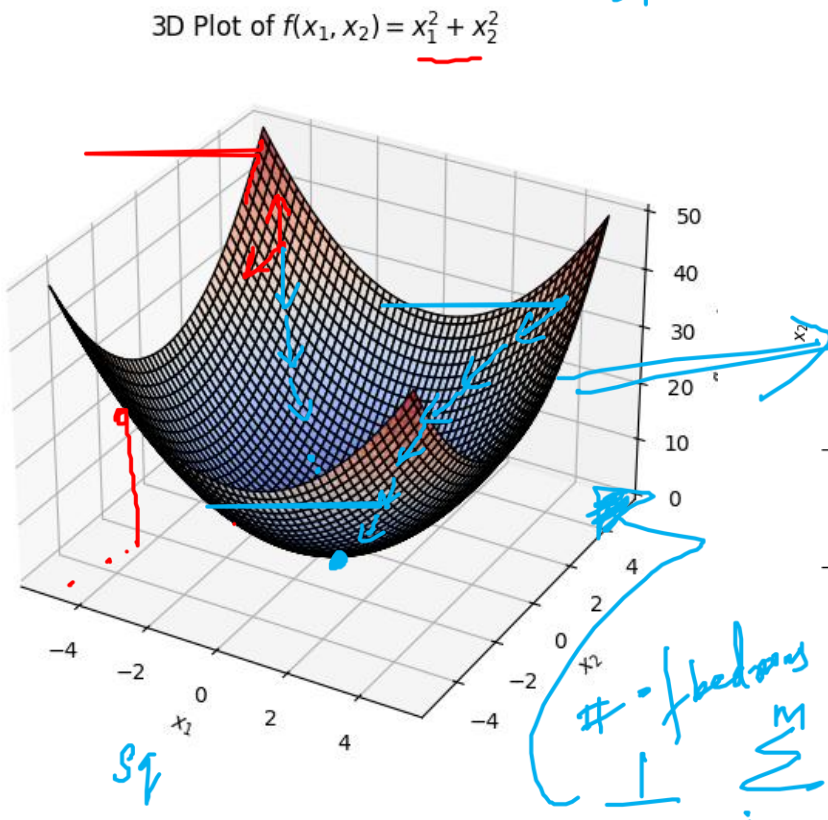
θ = random initialization

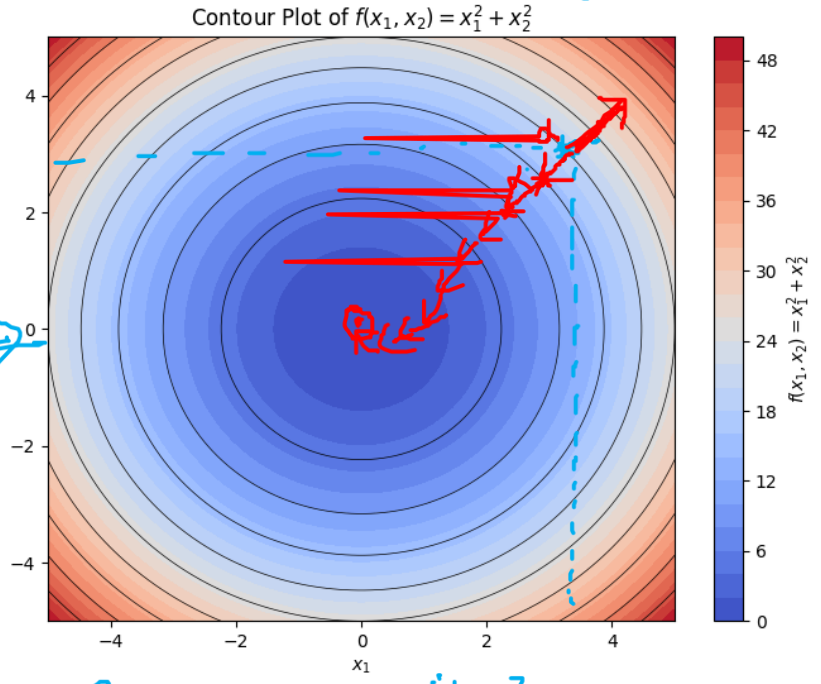
$\theta - \alpha \frac{\partial Error}{\partial \theta}$

α : learning rate

$\theta_{new} = \theta_{old} - \alpha \frac{\partial Error}{\partial \theta_{old}}$

Level set





Sq

# of bedrooms

$\frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$

To measure how well our model predicts, we use the **Mean Squared Error (MSE)**:

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

where:

- m is the number of training examples
- $y^{(i)}$ is the actual output for the i-th training example
- $h_\theta(x^{(i)})$ is the predicted output.
- The factor $\frac{1}{2}$ is used for mathematical convenience when differentiating.

$$\theta_{new} := \theta_{old} - \frac{\partial J(\theta)}{\partial \theta}$$

$$\theta_{new} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix}_{n+1} \qquad \vec{x} = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

$$\frac{\partial J(\theta)}{\partial \theta_0}, \quad \frac{\partial J(\theta)}{\partial \theta_1}, \quad \frac{\partial J(\theta)}{\partial \theta_2}, \quad \ldots \quad \frac{\partial J(\theta)}{\partial \theta_n}$$

$m = 3$, # features $= 2$

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{3} \left(\theta_0 x_0^{(i)} + \theta_1 x_1^{(i)} + \theta_2 x_2^{(i)} - y^{(i)} \right)^2$$

$$\frac{\partial J(\theta)}{\partial \theta_0} = \frac{1}{2m} \left[\left(\theta_0 x_0^{(1)} + \theta_1 x_1^{(1)} + \theta_2 x_2^{(1)} - y^{(1)} \right)^2 + \left(\theta_0 x_0^{(2)} + \theta_1 x_1^{(2)} + \theta_2 x_2^{(2)} - y^{(2)} \right)^2 + \left(\theta_0 x_0^{(3)} + \theta_1 x_1^{(3)} + \theta_2 x_2^{(3)} - y^{(3)} \right)^2 \right]$$

$$\frac{\partial J(\theta)}{\partial \theta_0} = \frac{1}{2m} \left[2\left(\theta_0 x_0^{(1)} + \theta_1 x_1^{(1)} + \theta_2 x_2^{(1)} - y^{(1)} \right) \times x_0^{(1)} + 2\left(\theta_0 x_0^{(2)} + \theta_1 x_1^{(2)} + \theta_2 x_2^{(2)} - y^{(2)} \right) x_0^{(2)} + 2\left(\theta_0 x_0^{(3)} + \theta_1 x_1^{(3)} + \theta_2 x_2^{(3)} - y^{(3)} \right) x_0^{(3)} \right]$$

$f(x) = (x-2)^2$

$2(x-2)$

$$\frac{\partial J(\theta)}{\partial \theta_0} = \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right) \times x_0^{(i)}$$

$$\frac{\partial J(\theta)}{\partial \theta} \Rightarrow$$

$$\begin{bmatrix} \frac{\partial J(\theta)}{\partial \theta_0} \\ \frac{\partial J(\theta)}{\partial \theta_1} \\ \vdots \\ \frac{\partial J(\theta)}{\partial \theta_n} \end{bmatrix}$$

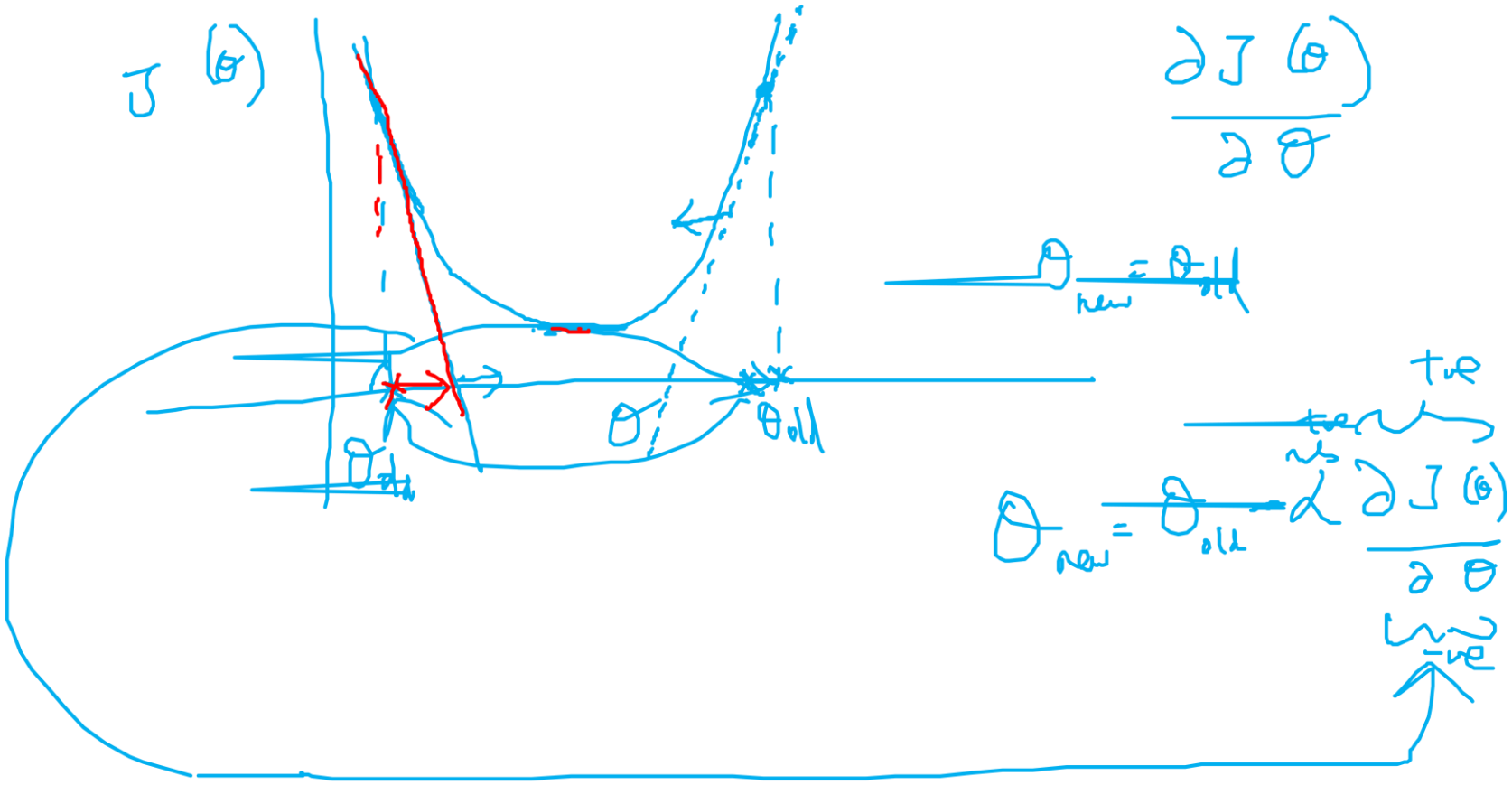
scalar

scalar

vector

$$\theta_{new} = \theta_{old} - \left(\frac{\partial J(\theta)}{\partial \theta}\right)$$

-ve



Gradient descent is used to minimize $J(\theta)$. The update rule is:

$$\theta_j := \theta_j - \alpha \frac{\partial J(\theta)}{\partial \theta_j}$$

where:

- α is the **learning rate**, controlling step size
- $\frac{\partial J(\theta)}{\partial \theta_j}$ is the gradient (partial derivative)

Computing the gradient:

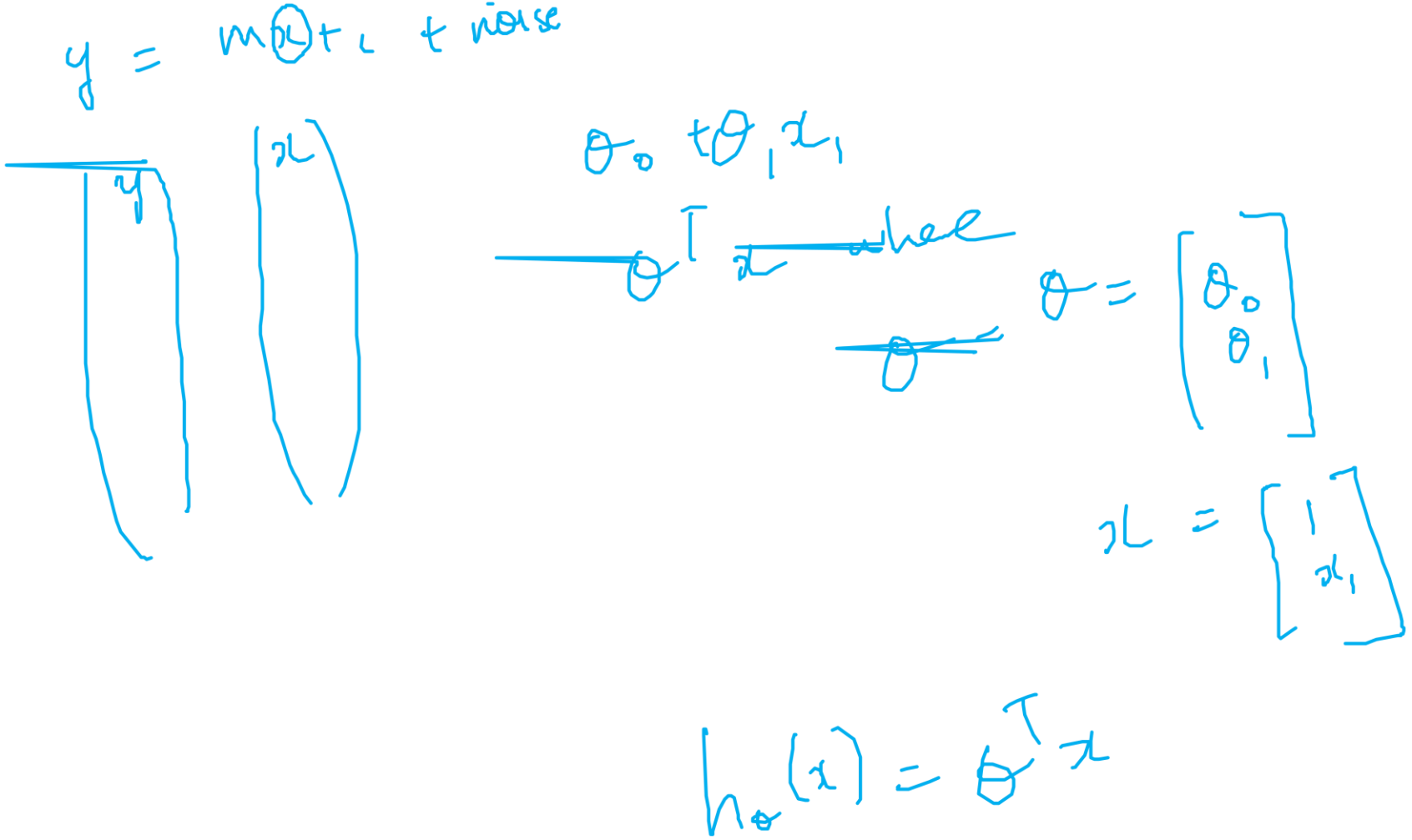
$$\frac{\partial J(\theta)}{\partial \theta_j} = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

Thus, the update step becomes:

$$\theta_j := \theta_j - \alpha \cdot \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

This process repeats until convergence (i.e., when $J(\theta)$ stops changing significantly).

Coding from Scratch



Logistic Regression

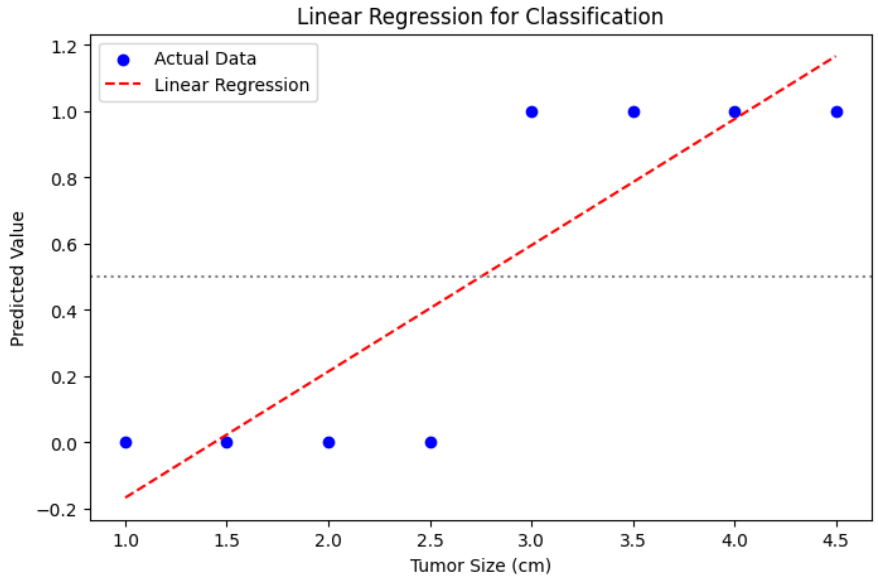
innovate achieve lead

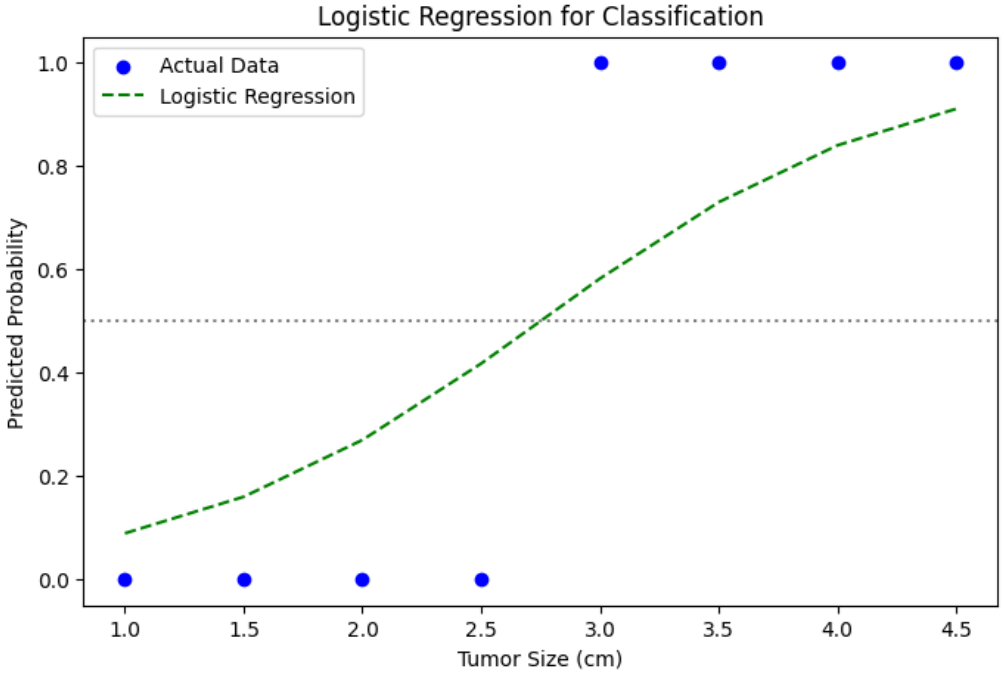
Scenario

Doctors want to classify tumors as **benign (0) or malignant (1)** based on tumor size. Larger tumors are more likely to be malignant, but there's no hard cutoff—so we use logistic regression to model this probability.

Hypothetical Data

Tumor Size (cm)	Diagnosis (Benign = 0, Malignant = 1)
1.0	0 (Benign)
1.5	0
2.0	0
2.5	0
3.0	1
3.5	1
4.0	1
4.5	1

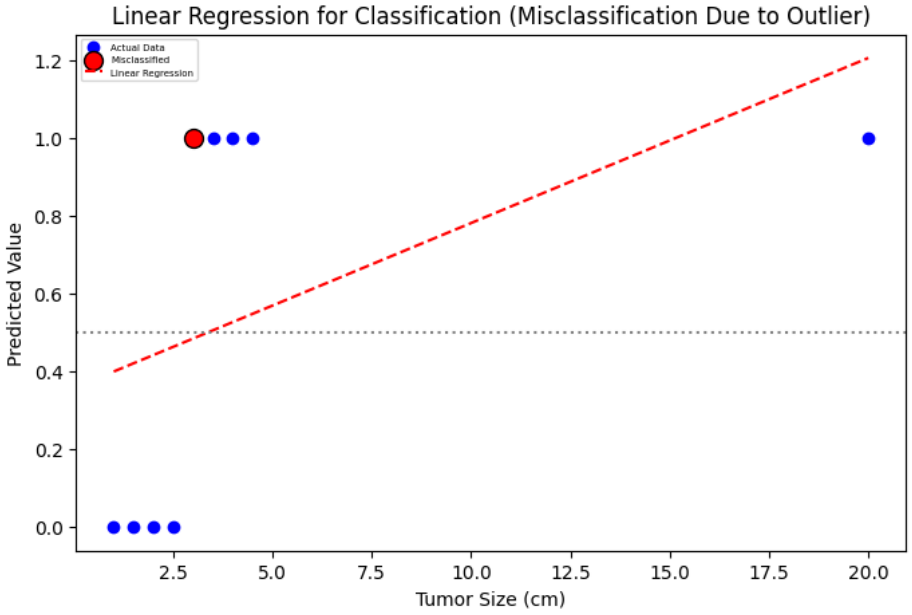

Linear Regression for Classification
Actual Data
Linear Regression
Predicted Value
Tumor Size (cm)
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5

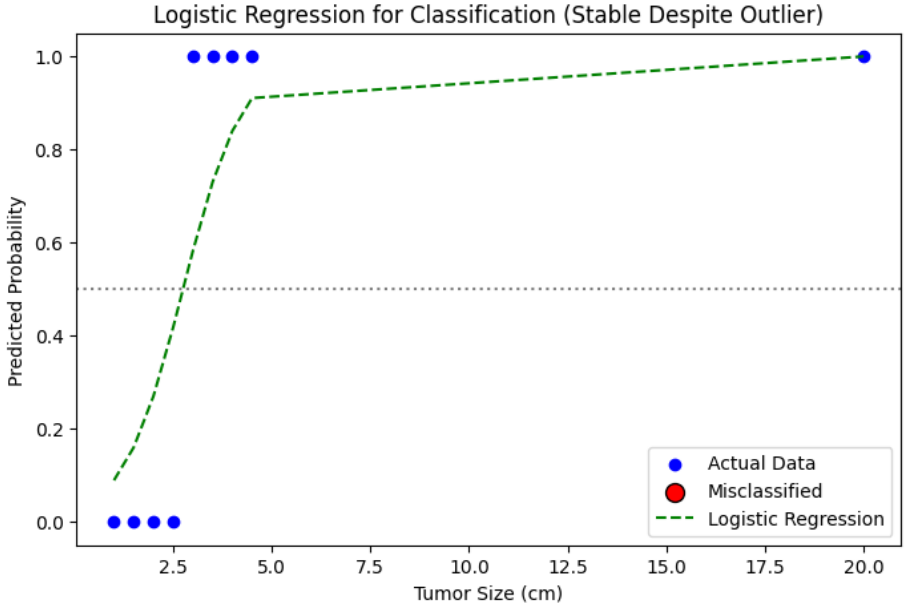

Logistic Regression for Classification
Actual Data
Logistic Regression
Predicted Probability
Tumor Size (cm)
1.0
0.8
0.6
0.4
0.2
0.0
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5

Effect of Outlier

Tumor Size (cm)	Diagnosis (0 = Benign, 1 = Malignant)
1.0	0
1.5	0
2.0	0
2.5	0
3.0	1
3.5	1
4.0	1
4.5	1
20.0	**1** (Outlier)

Effect of Outlier





Threshold = 0.5,
Accuracy = 0.89

1. Effect on Linear Regression

- Linear regression tries to fit a straight line, minimizing squared errors.
- A **large outlier (e.g., tumor size = 10 cm, malignant = 1)** will **pull the line upwards**, making it **steeper**.
- This **shifts the decision threshold** (where the prediction crosses 0.5), possibly classifying smaller tumors incorrectly.

2. Effect on Logistic Regression

- Logistic regression models **probabilities** using the **sigmoid function**, which is bounded between **0 and 1**.
- A large outlier **won't distort the S-curve** as much because it saturates towards 1.
- The decision boundary (where probability = 0.5) **remains relatively stable**.

Logistic Regression

The hypothesis function in **logistic regression** is given by:

$$h_\theta(x) = g(\theta^T x)$$

where:

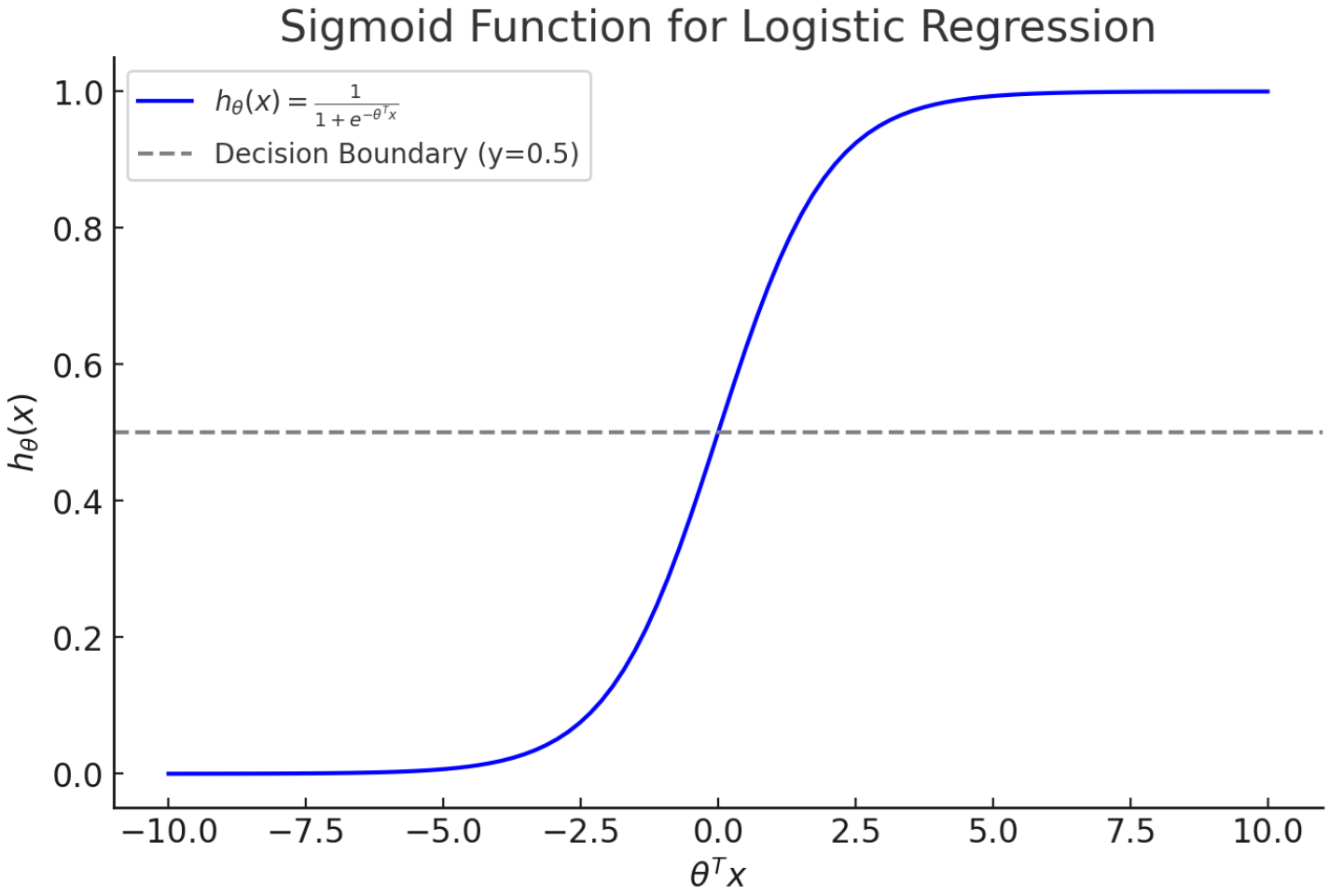
- θ is the parameter vector (weights).
- x is the feature vector (input data).
- $\theta^T x$ represents the **linear combination** of features and parameters.
- $g(z)$ is the **sigmoid function**, defined as:

$$g(z) = \frac{1}{1 + e^{-z}}$$

Thus, expanding $h_\theta(x)$:

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

This function maps any real number to the range **(0,1)**, making it suitable for **probability estimation** in classification problems.


Sigmoid Function for Logistic Regression
$h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$
Decision Boundary (y=0.5)
$h_\theta(x)$
1.0
0.8
0.6
0.4
0.2
0.0
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
$\theta^T x$

Mathematical Formulation of Logistic Regression

We have already defined the **hypothesis function** as:

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

Now, let's go through the remaining key mathematical components of logistic regression.

1. Probability Interpretation

Since $h_\theta(x)$ represents the probability of $y = 1$, we can write:

$$P(y = 1|x; \theta) = h_\theta(x)$$

$$P(y = 0|x; \theta) = 1 - h_\theta(x)$$

Likelihood Function

innovate achieve lead

1. Likelihood Function

Given a dataset $\{(x^{(i)}, y^{(i)})\}_{i=1}^{m}$, we assume that each training example follows a **Bernoulli distribution**:

$$P(y|x;\theta) = h_\theta(x)^y(1 - h_\theta(x))^{(1-y)}$$

For all m training samples, assuming independence, the **likelihood function** is the product of the probabilities for all data points:

$$L(\theta) = \prod_{i=1}^{m} h_\theta(x^{(i)})^{y^{(i)}}(1 - h_\theta(x^{(i)}))^{(1-y^{(i)})}$$

where:

- $h_\theta(x^{(i)}) = \frac{1}{1+e^{-\theta^T x^{(i)}}}$ is the predicted probability of $y = 1$.
- If $y^{(i)} = 1$, the term $(1 - h_\theta(x^{(i)}))^{(1-y^{(i)})}$ vanishes.
- If $y^{(i)} = 0$, the term $h_\theta(x^{(i)})^{y^{(i)}}$ vanishes.

Maximizing Likelihood

innovate achieve lead

Maximizing the Likelihood Function in Logistic Regression

Once we have the **likelihood function**:

$$L(\theta) = \prod_{i=1}^{m} h_\theta(x^{(i)})^{y^{(i)}} (1 - h_\theta(x^{(i)}))^{(1-y^{(i)})}$$

our goal is to find the parameters θ that **maximize** this function. In other words, we want to find the **best** θ such that the predicted probabilities align as closely as possible with the actual labels in the training data.

Why Maximize the Likelihood?

- The likelihood function represents the probability of observing the given dataset, assuming the logistic regression model is correct.
- A higher likelihood means that the model's predicted probabilities closely match the actual outcomes.
- By maximizing $L(\theta)$, we are **choosing parameters θ that make the observed data most probable** under our model.

Step 1: Taking the Log of the Likelihood

Given the likelihood function:

$$L(\theta) = \prod_{i=1}^{m} h_\theta(x^{(i)})^{y^{(i)}} (1 - h_\theta(x^{(i)}))^{(1-y^{(i)})}$$

we take the **log-likelihood**:

$$\ell(\theta) = \log L(\theta) = \sum_{i=1}^{m} \left[y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)})) \right]$$

To ensure consistency with loss functions commonly used in machine learning, we take the **average log-likelihood** (by including $\frac{1}{m}$):

$$\ell(\theta) = \frac{1}{m} \sum_{i=1}^{m} \left[y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)})) \right]$$

Step 3: Converting to a Minimization Problem

Optimization algorithms typically **minimize** a function rather than maximize it. Instead of maximizing $\ell(\theta)$, we minimize its **negative**:

$$J(\theta) = -\ell(\theta)$$

Substituting the log-likelihood:

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

This is the **binary cross-entropy loss** (also called **log-loss**), which is the function we minimize using **gradient descent**.

Gradient Descent for Parameter Estimation

The gradient of the **log-loss function** with respect to θ_j is:

$$\frac{\partial J(\theta)}{\partial \theta_j} = \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right) x_j^{(i)}$$

where:

- m is the number of training examples.
- $h_\theta(x^{(i)}) = \frac{1}{1+e^{-\theta^T x^{(i)}}}$ is the predicted probability.
- $y^{(i)}$ is the actual class label (0 or 1).
- $x_j^{(i)}$ is the j-th feature of the i-th training example.

We update θ_j using **gradient descent**:

$$\theta_j := \theta_j - \alpha \cdot \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right) x_j^{(i)}$$

where:

- α is the **learning rate**, controlling how big the update steps are.

Coding