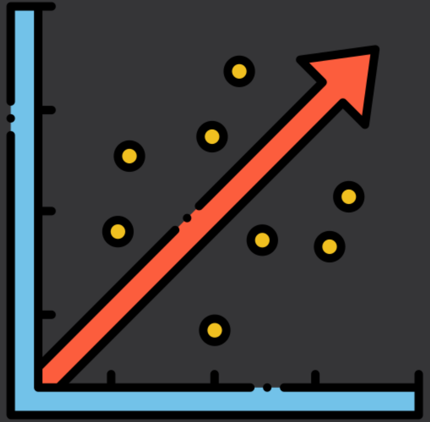


Multiple Linear Regression

Lecture 9 – HCCDA-AI

Imran Nawar

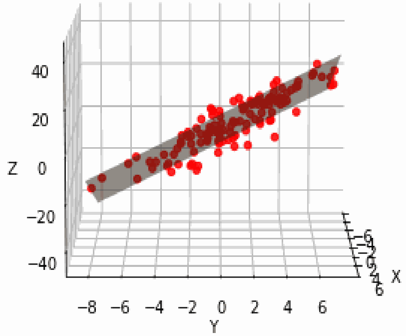
28 June 2025

Linear Regression with Multiple Variables

- Linear regression with multiple variable also known as **multiple linear regression** or **multivariate linear regression**, extends the concept of simple linear regression to the case where there are multiple independent variables.

- **Key Points:**
  - It predicts outcomes based on multiple input features.
  - The relationship is represented by a linear equation:

$$y = b_0 + \ b_1 x_1 + b_2 x_2 + \cdots + b_n x_n$$



  - where:
    - y = dependent variable (target)
    - $x_1, x_2, \ldots, x_n$ = independent variables (features)
    - b_0= intercept
    - $b_1, b_2, \ldots, b_n$= regression coefficients (weights)
  - Assumes a linear relationship between the dependent and independent variables.

Single feature (variable)

Size (feet²) x	Price ($1000) y
2104	460
1416	232
1534	315
852	178
…	…

$$h_\theta(x) = \theta_0 + \theta_1 x$$

Multiple features (variables).

Size (feet^2)	Number of bedrooms	Number of floors	Age of home (years)	Price ($1000)
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178
…	…	…	…	…

Notation:

n = number of features

$x^{(i)}$ = input (features) of i^{th} training example.

$x_j^{(i)}$ = value of feature j in i^{th} training example.

Hypothesis:

- Previously: $$h_\theta(x) = \theta_0 + \theta_1 x$$

Multivariate Linear Regression

Hypothesis: Vectorization

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

For convenience of notation, define $x_0 = 1$

Multiple Linear Regression

Cost Function for Multiple Variables

Hypothesis: $h_\theta(x) = \theta^T x = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$

Parameters: $\theta_0, \theta_1, \ldots, \theta_n$

Cost function:

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

Multiple Linear Regression

Gradient Descent for Multiple Variables

Hypothesis: $h_\theta(x) = \theta^T x = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$

Parameters: $\theta_0, \theta_1, \ldots, \theta_n$

Cost function:

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

Gradient descent:

$$\text{Repeat } \{$$

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \ldots, \theta_n)$$

$$\}$$

(simultaneously update for every $j = 0, \ldots, n$)

Hypothesis: $h_\theta(x) = \theta^T x = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$

Parameters: $\theta_0, \theta_1, \ldots, \theta_n$

Cost function:

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

Gradient descent:

Repeat $\{$

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \ldots, \theta_n)$$

$\}$

(simultaneously update for every $j = 0, \ldots, n$)

Gradient Descent

Previously (n=1):

Repeat $\{$

$$\theta_0 := \theta_0 - \alpha \underbrace{\frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})}_{\frac{\partial}{\partial \theta_0} J(\theta)}$$

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

(simultaneously update θ_0, θ_1)

$\}$

New algorithm $(n \geq 1)$:

Repeat $\{$

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

(simultaneously update θ_j for $j = 0, \ldots, n$)

$\}$

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_0^{(i)}$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_1^{(i)}$$

$$\theta_2 := \theta_2 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_2^{(i)}$$

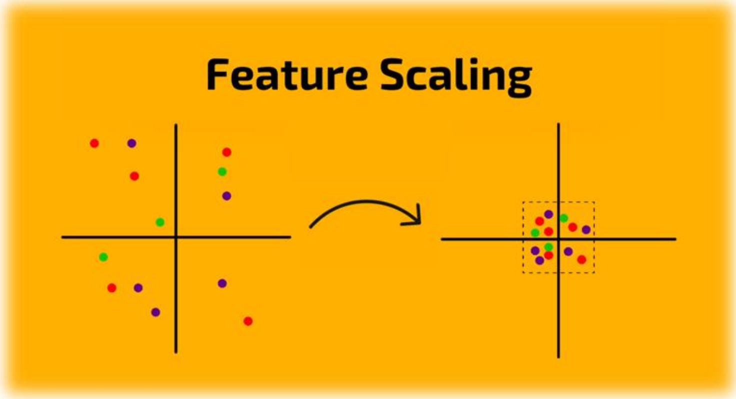
$\ldots$

Multiple Linear Regression

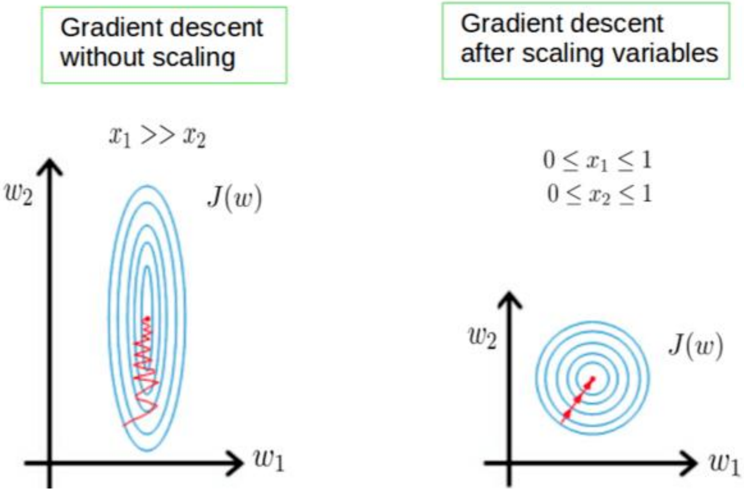
Gradient Descent in practice I: Feature Scaling

Feature Scaling

- Feature Scaling is a technique that enable gradient descent to run much faster.
- It is a preprocessing technique used in machine learning to standardize or normalize the range of independent variables or features of the dataset.



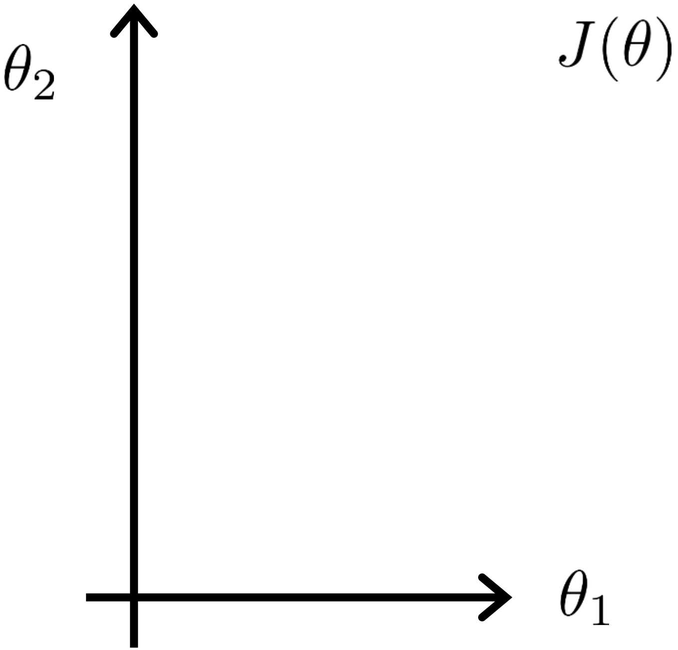
- **Idea:** Make sure features are on a similar scale.



Feature Scaling

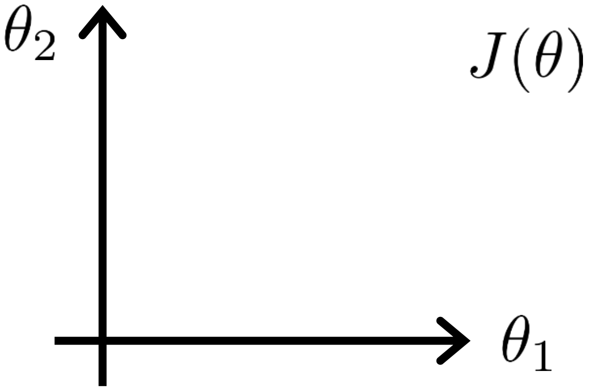
E.g. x_1 = size (0-2000 feet^2)

x_2 = number of bedrooms (1-5)



$$x_1 = \frac{\text{size (feet}^2\text{)}}{2000}$$

$$x_2 = \frac{\text{number of bedrooms}}{5}$$



Mean normalization

Replace x_i with $x_i - \mu_i$ to make features have approximately zero mean (Do not apply to $x_0 = 1$).

E.g. $x_1 = \frac{size - 1000}{2000}$

$x_2 = \frac{\#bedrooms - 2}{5}$

$$-0.5 \leq x_1 \leq 0.5, -0.5 \leq x_2 \leq 0.5$$

Multiple Linear Regression

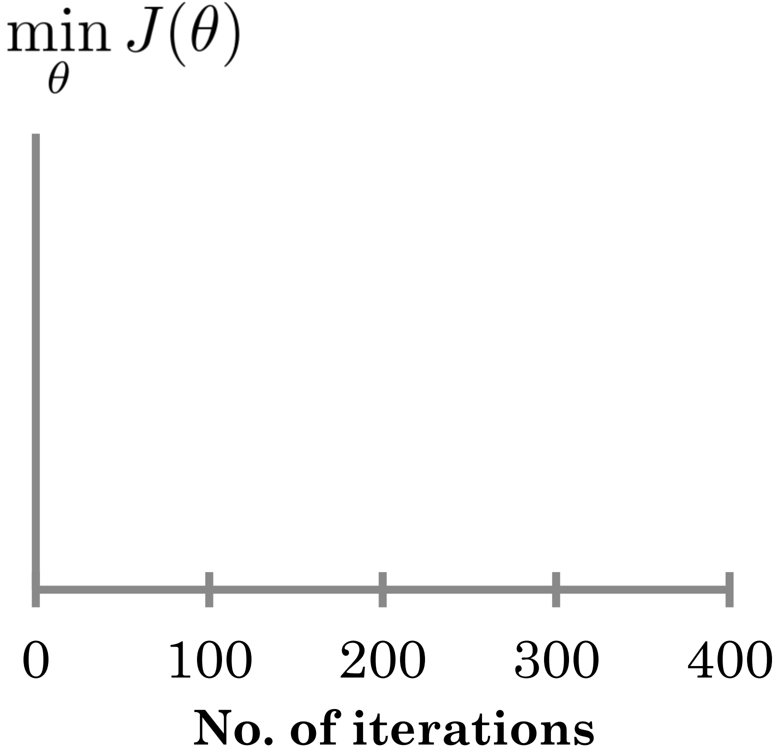
Gradient Descent in practice II: Learning Rate

Gradient descent

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

- "Debugging": How to make sure gradient descent is working correctly.
- How to choose learning rate α .

Making sure gradient descent is working correctly

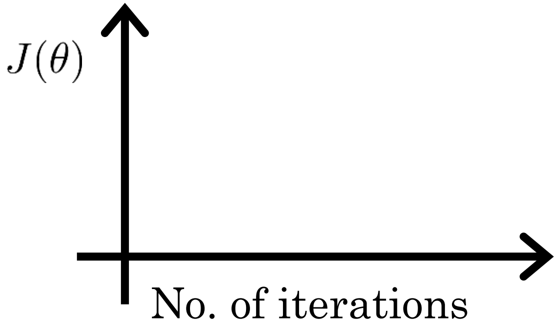


Example automatic Convergence test:

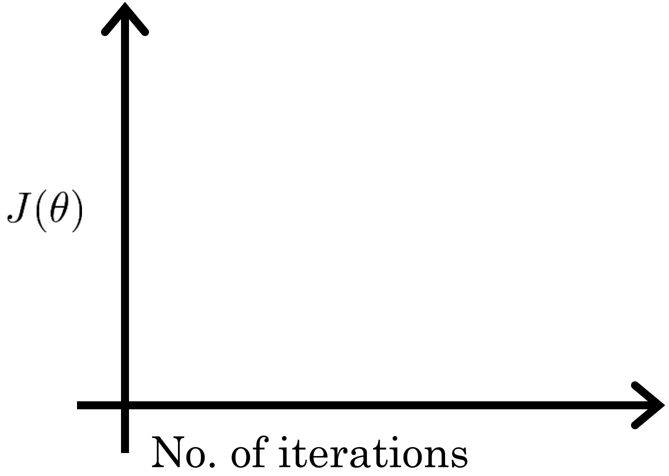
Declare convergence if $J(\theta)$ decreases by less than 10^{-3} in one iteration.

For different application, it may take different iteration, 30, 3000, etc.

Making sure gradient descent is working correctly



Gradient descent not working.
Use smaller α .



- For sufficiently small α, $J(\theta)$ should decrease on every iteration.
- But if α is too small, gradient descent can be slow to converge.

Summary:

- If α is too small: Slow convergence.
- If α is too large: $J(\theta)$ may not decrease on every iteration; may not converge.

To choose α, try

$$\ldots, 0.001, \qquad , 0.01, \qquad , 0.1, \qquad , 1, \ldots$$

Multiple Linear Regression

Feature Engineering

Feature Engineering

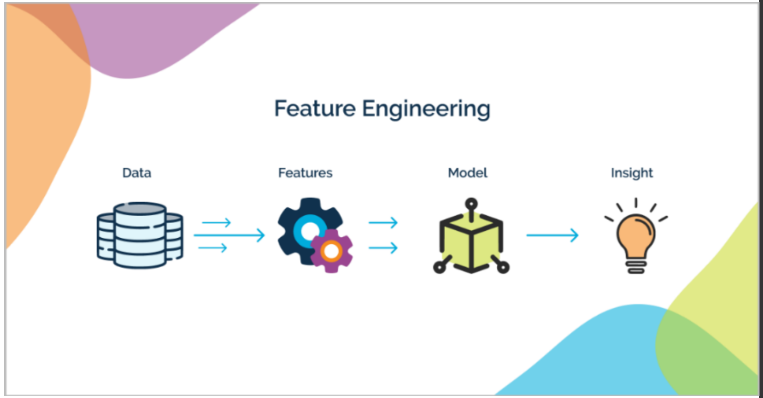
- Use intuition to design new features, by transforming or combining original features.
- For example:

  House Price Prediction

$$h_\theta(x) = \theta_0 + \theta_1 \times frontage + \theta_2 \times depth$$

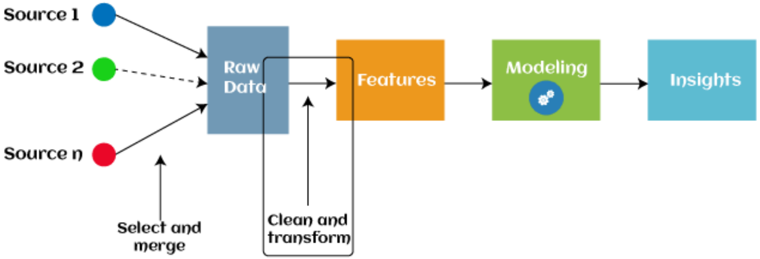
Feature Engineering

- Feature engineering is the process of creating, transforming, or combining features from existing data to make a machine learning model more effective.
- It helps extract meaningful patterns that improve model performance.



Why is Feature Engineering Important?

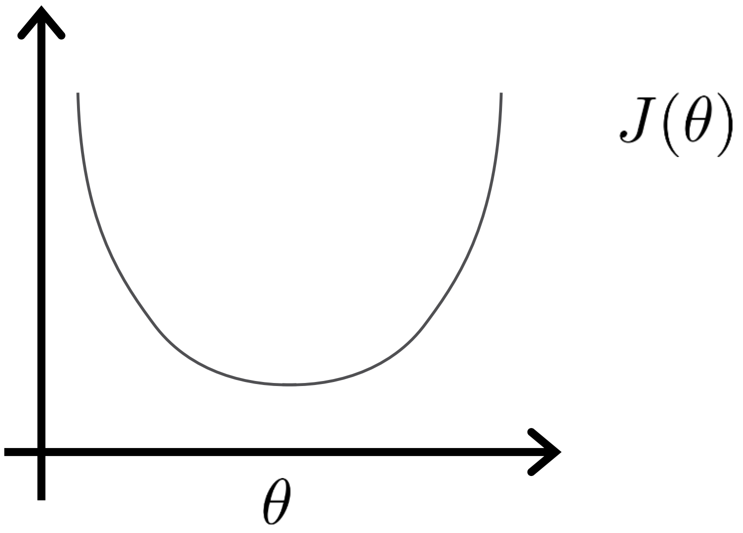
- Improves model accuracy by providing better input features.
- Helps capture hidden patterns in the data.
- Reduces the need for complex models by improving input quality.



Multiple Linear Regression

Normal Equation

Gradient Descent



Normal equation: Method to solve for θ analytically.

Normal Equation

- Used for linear regression and linear models.
- Solve for θ_i without iterations.
- Where as it turns out gradient descent is a great method for minimizing the cost function $J(\theta_0, \theta_1)$. The normal equation is a method used to find the optimal parameters for linear regression model analytically.
- **Disadvantages:**
  - Does not generalize to other learning algorithms.
  - Slow when number of features are large (> 10000).
  - Invertibility issue.

$$\boxed{\theta = \left(X^T X\right)^{-1} . \left(X^T y\right)}$$

Examples: $m = 4$.

	Size (feet²)	Number of bedrooms	Number of floors	Age of home (years)	Price ($1000)
x_0	x_1	x_2	x_3	x_4	y
1	2104	5	1	45	460
1	1416	3	2	40	232
1	1534	3	2	30	315
1	852	2	1	36	178

$$X = \begin{bmatrix} 1 & 2104 & 5 & 1 & 45 \\ 1 & 1416 & 3 & 2 & 40 \\ 1 & 1534 & 3 & 2 & 30 \\ 1 & 852 & 2 & 1 & 36 \end{bmatrix} \qquad y = \begin{bmatrix} 460 \\ 232 \\ 315 \\ 178 \end{bmatrix}$$

$$\theta = (X^T X)^{-1} X^T y$$

Examples: $m = 5$.

x_0	Size (feet²) x_1	Number of bedrooms x_2	Number of floors x_3	Age of home (years) x_4	Price ($1000) y
1	2104	5	1	45	460
1	1416	3	2	40	232
1	1534	3	2	30	315
1	852	2	1	36	178
1	3000	4	1	38	540

$$X = \begin{bmatrix} 1 & 2104 & 5 & 1 & 45 \\ 1 & 1416 & 3 & 2 & 40 \\ 1 & 1534 & 3 & 2 & 30 \\ 1 & 852 & 2 & 1 & 36 \\ 1 & 3000 & 4 & 1 & 38 \end{bmatrix} \qquad y = \begin{bmatrix} 460 \\ 232 \\ 315 \\ 178 \\ 540 \end{bmatrix}$$

$$\Theta = (X^T X)^{-1} X^T y$$

Andrew N

$$\theta = (X^T X)^{-1} X^T y$$

$(X^T X)^{-1}$ is inverse of matrix $X^T X$

NumPy: $np.linalg.pinv(X.T \ @ \ X) \ @ \ X.T \ @ \ y$

m training examples, n features.

Gradient Descent

- Need to choose α.
- Needs many iterations.
- Works well even when n is large.

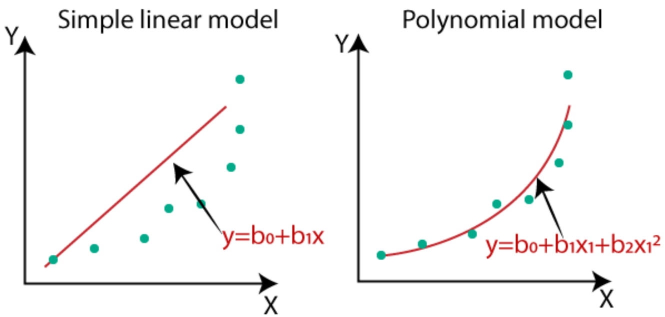
Normal Equation

- No need to choose α.
- Don't need to iterate.
- Need to compute $(X^T X)^{-1}$
- Slow if n is very large.

Polynomial Regression

Polynomial Regression

- Polynomial regression is an extension of linear regression where the relationship between the **independent variable(s) (x)** and the **dependent variable (y)** is modeled as a **polynomial equation**.
- Unlike simple linear regression, which fits a **straight line**, polynomial regression captures **curved, non-linear** patterns in data.



- ***Mathematical Representation***

$$y = b_0 + b_1x_1 + b_2x_1^2 + \cdots + b_nx_1^n$$

where:

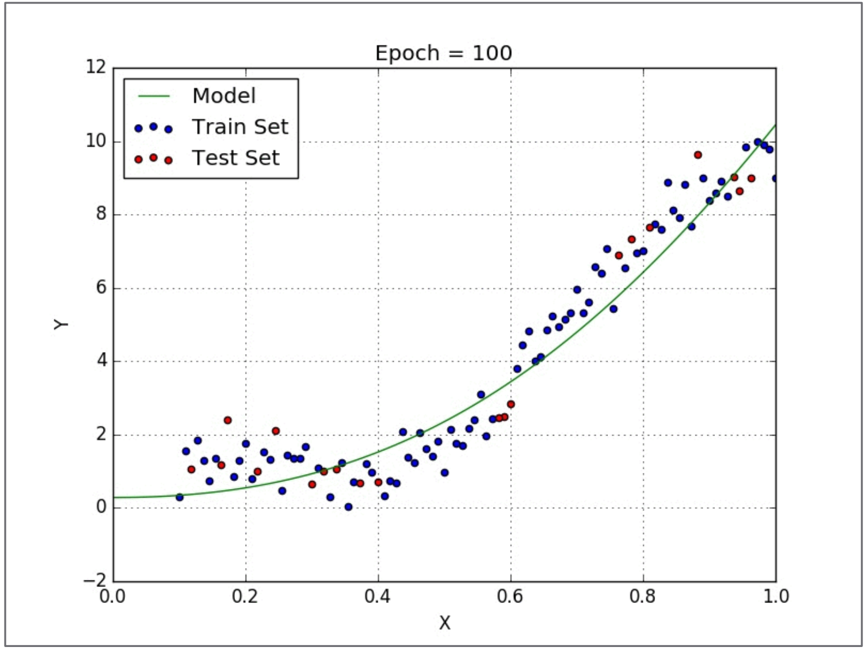
- y = dependent variable (output)
- $x_1, x_2, \ldots, x_n$ = independent variables (features/input)
- b_0= intercept
- $b_1, b_2, \ldots, b_n$= regression coefficients (weights)
- n = degree of the polynomial

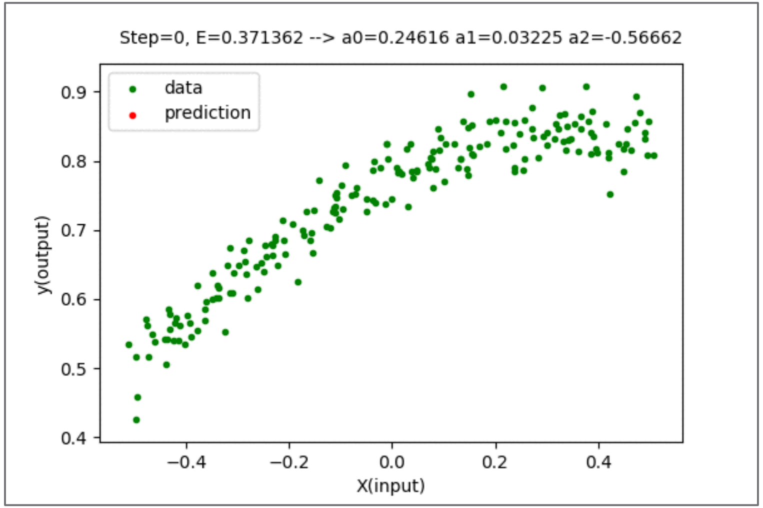
- **When to Use Polynomial Regression?**
  - When data shows a **non-linear trend** (i.e., a straight line doesn't fit well).
  - If the relationship between **x** and **y** forms a curve instead of a straight line.

Simple Linear Regression	$y = b_0 + b_1x_1$
Multiple Linear Regression	$y = b_0 + b_1x_1 + b_2x_2 + \ldots + b_nx_n$
Polynomial Linear Regression	$y = b_0 + b_1x_1 + b_2x_1^2 + \ldots + b_nx_1^n$

Polynomial Regression

The visualizations show how polynomial regression fits a curved line to the data, capturing non-linear patterns that a straight line cannot.





Hypothesis: $h_\theta(x) = \theta_0 + \ \theta_0 x + \ \theta_0 x^2 + \cdots + \ \theta_n x^n$

In Vectorized form, this can be written as:

$$h_\theta(x) = \theta^T \mathrm{X}$$

Parameters: $\theta_0, \theta_1, \ldots, \theta_n$

Cost function:

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

Gradient descent:

Repeat $\{$

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \ldots, \theta_n)$$

$\}$

(simultaneously update for every $j = 0, \ldots, n$)

Thank You