



Linear regression with multi variables

Ali Aburas, PhD

Linear Regression (example-1)



- Home prices in Tripoli Libya


area	price
2600	550000
3000	565000
3200	610000
3600	680000
4000	725000

Given these home prices find out prices whose area is,

3300
5000

Hypothesis: $h_{\theta}(x) = \theta_0 + \theta_1 * \text{area}$

Linear regression with multiple variables



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

→

$m=47$

Notation:

n = number of features $n=4$

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

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

$$X^{(2)} = \begin{bmatrix} 1416 \\ 3 \\ 2 \\ 40 \end{bmatrix}$$

$$x_3^{(2)} = 2$$

Linear regression with multiple variables

The diagram illustrates the components of a linear regression equation. It features two equations: a specific one at the top and a general one at the bottom. Arrows connect the terms in the specific equation to their corresponding parts in the general equation.

Specific Equation: $price = \theta_1 * area + \theta_2 * bedrooms + \theta_3 * age + \theta_0$

General Equation: $y = \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \theta_0$

Annotations:

- Dependent variable:** Points to $price$ in the specific equation and y in the general equation.
- Independent variables (**features**):** Points to $area$, $bedrooms$, and age in the specific equation, and x_1 , x_2 , and x_3 in the general equation.
- Coefficients:** Points to θ_1 , θ_2 , and θ_3 in both equations.
- Intercept:** Points to θ_0 in both equations.

Linear regression with multiple variables

Model/ Hypothesis: Previously: $h_{\theta}(x) = \theta_0 + \theta_1 x$

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

The weights used by the model indicate the effect of each descriptive feature on the predictions returned by the model

$$h_{\theta} = 80 + 0.1x_1 + 3x_2 + 0.01x_3 -$$

Base Price


size

↑ no. of bedroom

↓ age of house

↑ no. of floor

Linear Regression model prediction


$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$$


- \hat{y} is the predicted value.
- n is the number of features.
- x_i is the i^{th} feature value.
- θ_j is the j^{th} model parameter (including the bias term θ_0 and the feature weights $\theta_1, \theta_2, \dots, \theta_n$).

This can be written much more concisely using a vectorized form,

$$\hat{y} = h_{\theta}(\mathbf{x}) = \boldsymbol{\theta} \cdot \mathbf{x}$$

- $\boldsymbol{\theta}$ is the model's *parameter vector*, containing the bias term θ_0 and the feature weights θ_1 to θ_n .
- \mathbf{x} is the instance's *feature vector*, containing x_0 to x_n , with x_0 always equal to 1.
- $\boldsymbol{\theta} \cdot \mathbf{x}$ is the dot product of the vectors $\boldsymbol{\theta}$ and \mathbf{x} , which is of course equal to $\theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$.
- h_{θ} is the hypothesis function, using the model parameters $\boldsymbol{\theta}$.

Linear Regression model prediction (vectorized form)


$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$$

For convenience of notation, define $x_0 = 1$

This can be written much more concisely using a vectorized form,

$$\boldsymbol{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix} \quad \mathbf{x} = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix} \quad \boldsymbol{\theta}^T = \begin{bmatrix} \theta_0 & \theta_1 & \cdots & \theta_n \end{bmatrix}$$

$\hat{y} = h_{\boldsymbol{\theta}}(\mathbf{x}) = \boldsymbol{\theta}^T \mathbf{x}$, where $\boldsymbol{\theta}^T$ is the *transpose* of $\boldsymbol{\theta}$

- $\boldsymbol{\theta}$ is the model's *parameter vector*, containing the bias term θ_0 and the feature weights θ_1 to θ_n .
- \mathbf{x} is the instance's *feature vector*, containing x_0 to x_n , with x_0 always equal to 1.
- $\boldsymbol{\theta} \cdot \mathbf{x}$ is the dot product of the vectors $\boldsymbol{\theta}$ and \mathbf{x} , which is of course equal to $\theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$.
- $h_{\boldsymbol{\theta}}$ is the hypothesis function, using the model parameters $\boldsymbol{\theta}$.

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: $\theta_0, \theta_1, \dots, \theta_n$

Cost function:

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

Gradient descent:

Repeat {

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

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

Gradient Descent (when $n \geq 1$)

Gradient Descent

Previously ($n=1$):

Repeat {

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

$$\underbrace{\hspace{10em}}_{\frac{\partial}{\partial \theta_0} J(\theta)}$$

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

(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
 $j = 0, \dots, n$)

}

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

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

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

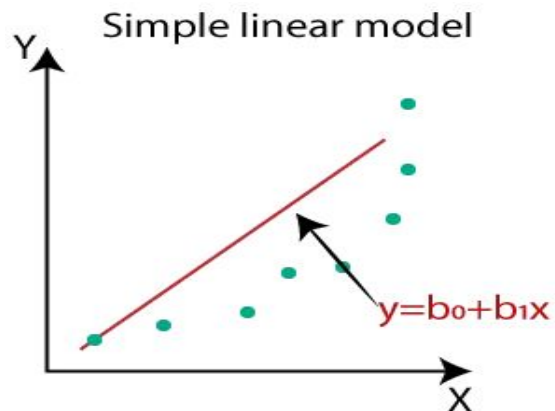
$$x_0 = 1$$

Polynomial Regression

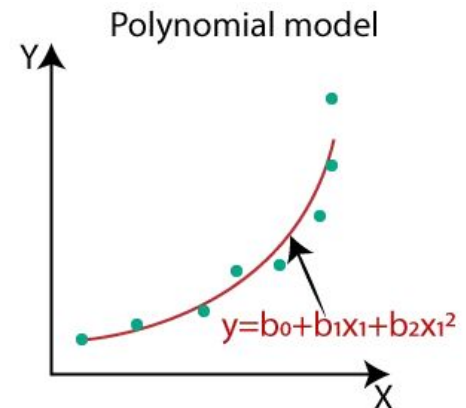
Polynomial Regression

- **Polynomial Regression** is a form of regression analysis in which the relationship between the independent variables and dependent variables are modeled in the N^{th} degree polynomial.
- Polynomial Regression is a special case of Linear Regression where we fit the polynomial equation on the data with a curvilinear relationship between the dependent and independent variables.

Clearly, a straight line will never fit this data properly



Polynomial Regression model (2nd-degree polynomial)



Math Behind Polynomial Regression!

Simple
Linear
Regression

$$y = b_0 + b_1 x_1$$

Multiple
Linear
Regression

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

Polynomial
Linear
Regression

$$y = b_0 + b_1 x_1 + b_2 x_1^2 + \dots + b_n x_1^n$$

Now, the value of \mathbf{b} is found out by matrix multiplication

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & x_1 & x_1^2 & \dots & x_1^m \\ 1 & x_2 & x_2^2 & \dots & x_2^m \\ 1 & x_3 & x_3^2 & \dots & x_3^m \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \dots & x_n^m \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \vdots \\ \beta_m \end{bmatrix}$$

Cost Function of Polynomial Regression

- **Cost Function** is a function that **measures the performance of a Machine Learning model** for given data.
- Cost Function is basically the calculation of the error between predicted values and expected values and **presents it in the form of a single real number**.
- Difference between **Cost Function** and **Loss Function**,
 - **Cost Function** is the average of error of n-sample in the data and **Loss Function** is the error for individual data points.
 - In other words, **Loss Function** is for one training example, **Cost Function** is for the entire training set.
- **Cost Function of Polynomial Regression** can also be taken to be Mean Squared Error, However there will be a slight change in the equation.

$$J = \frac{1}{n} \sum_{i=1}^n (pred_i - y_i)^2$$

#Cost Function of Linear Regression

```
J = 1/n*sum(square(pred - y))
```

Which, can also be written as :

```
J = 1/n*sum(square(pred-(b0 + b1x1))) i.e, y = mx+b
```

#Cost Function of Polynomial Regression

```
J = 1/n*sum(square(pred - y))
```


However, here the equation of y will change. So, the equation can also be written as:

```
J = 1/n*sum(square(pred - (b0 + b1x + b2x^2 + b3x^3.....)))
```

Generalization in Machine Learning

—

What is Generalization in Machine Learning?

- 
- The goal of a good machine learning model is to generalize well from the training data to any data from the problem domain. This allows us to make predictions in the future on data the model has never seen. **“learning general concepts from specific examples”**
 - A spam email classifier is a great example of generalization in machine learning.
 - Suppose you have a training dataset containing emails labeled as either spam or not spam
 - and your goal is to build a model that can accurately classify incoming emails as spam or legitimate based on their content.
 - During the training phase, the machine learning algorithm learns from the set of labeled emails, extracting relevant features and patterns to make predictions.
 - The true test of the model's effectiveness lies when new emails arrive, the model needs to accurately classify them as spam or legitimate without prior exposure to their content. **This is where generalization comes in.**
 - ***Overfitting and underfitting*** are the two biggest causes for **poor performance of machine learning algorithms.**