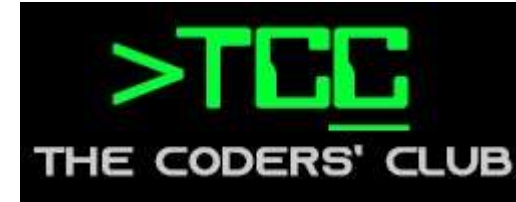


MACHINE LEARNING



LINEAR REGRESSION WITH MULTIPLE VARIABLES

WEEK 2

Linear Regression with Multiple Variables

For a single variable, the linear regression was given by,
 $h_{\theta}(x) = \theta_0 + \theta_1 x$

Example:

Size (sq. feet) (x)	Price (\$1000) (y)
2104	460
1416	232
1534	315
...	...

Here we have only one feature (x) which is the size of house in sq. feet.

Multiple Features

Example:

Size (sq. feet) (x_1)	No. of bedrooms (x_2)	No. of floors (x_3)	Age of home in years (x_4)	Price (\$1000) y
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178
...

Notations:

n = no. of features

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

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

Example:

From the given dataset, values of $x^{(2)}$ and $x_3^{(2)}$ are given by:

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

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

Size (sq. feet)	No. of bedrooms	No. of floors	Age of home in years	Price (\$1000)
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178
...

In the training set above, what is $x_1^{(4)}$?

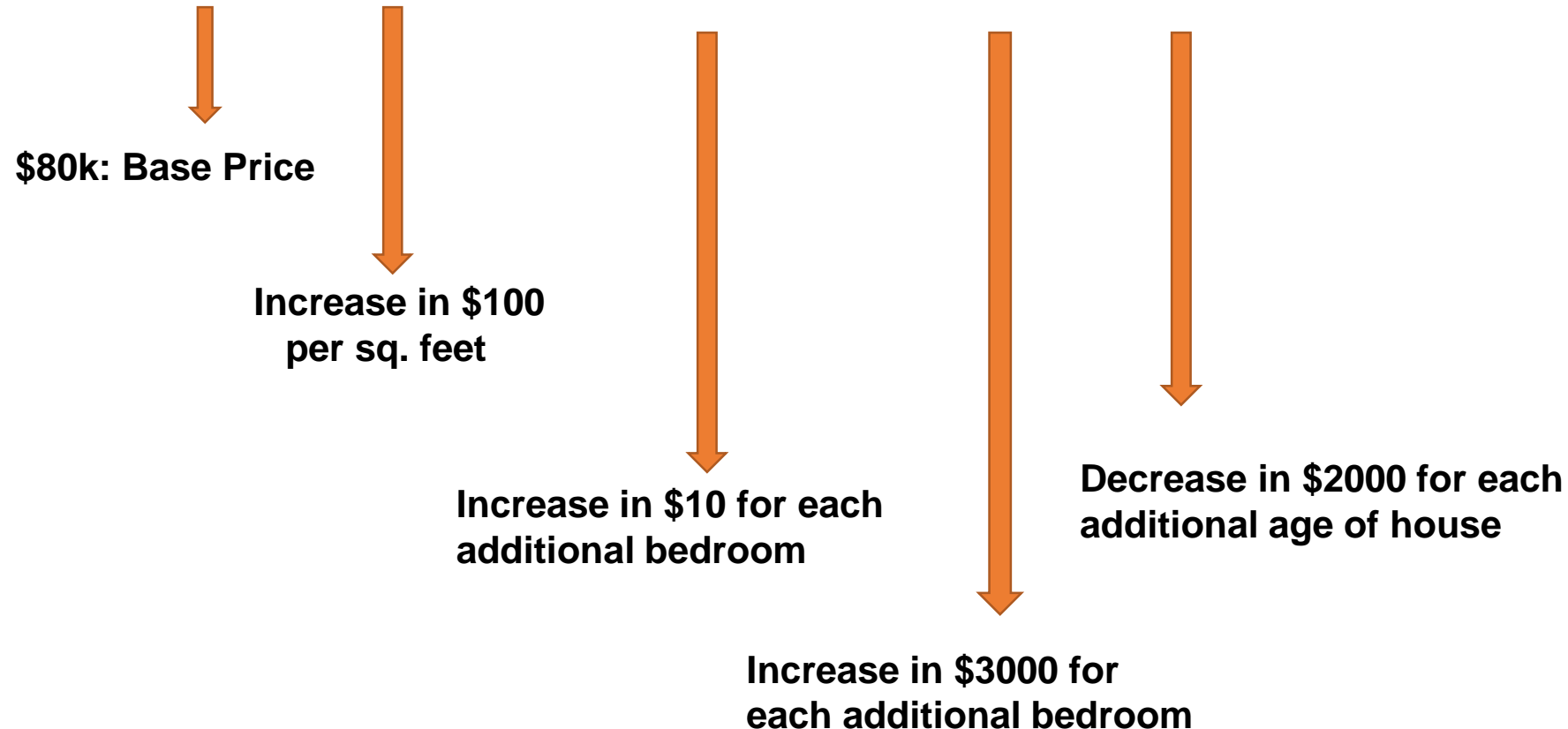
- The size (in sq. feet) of the 1st home in the training set
- The age (in years) of the 1st home in the training set
- The size (in sq. feet) of the 4th home in the training set
- The age (in years) of the 4th home in the training set

Hypothesis for linear regression in multiple variables is given by,

$$h_{\theta}(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \dots + \theta_n x_n$$

Example:

$$h_{\theta}(x) = 80 + (0.1)(x_1) + (0.01)(x_2) + (3)(x_3) - (2)(x_4)$$



$$h_{\theta}(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \dots + \theta_n x_n$$


This can be represented as,

$$h_{\theta}(x) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \dots + \theta_n x_n$$

where $x_0 = 1$ i.e. $x_0^{(i)} = 1$

In matrix form, they can be represented by,

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \dots \\ \theta_n \end{bmatrix}, \quad X = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \dots \\ x_n \end{bmatrix}$$



$(n+1) \times 1$ vector $(n+1) \times 1$ vector

$$\theta^T = \begin{bmatrix} \theta_0 & \theta_1 & \theta_2 & \theta_3 & \dots & \theta_n \end{bmatrix}, \quad X = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \dots \\ x_n \end{bmatrix}$$

$1 \times (n+1)$ matrix

$(n+1) \times 1$ vector

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

Hence, we have “vectorized” our hypothesis function for multiple variables.

Summary

Hypothesis:

$$h_{\theta}(x) = \theta^T x = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \dots + \theta_n x_n$$

Parameters:

$$\theta_0, \theta_1, \theta_2, \theta_3, \dots, \theta_n = \theta \quad (n+1) \text{ dimensional vector}$$

Cost Function:

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

When there are n features, we define the cost function as

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

For linear regression, which of the following are also equivalent and correct definitions of $J(\theta)$?

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

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^m \left(\left(\sum_{j=0}^n \theta_j x_j^{(i)} \right) - y^{(i)} \right)^2 \quad (\text{Inner sum starts at 0}) \quad \checkmark$$

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^m \left(\left(\sum_{j=1}^n \theta_j x_j^{(i)} \right) - y^{(i)} \right)^2 \quad (\text{Inner sum starts at 1})$$

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^m \left(\left(\sum_{j=0}^n \theta_j x_j^{(i)} \right) - \left(\sum_{j=0}^n y_j^{(i)} \right) \right)^2$$

Gradient Descent for Multiple Variables

repeat until convergence

{

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta) \quad (\text{simultaneously update for every } j=0, 1, 2, 3, \dots n)$$

}

The gradient descent equation itself is generally the same form; we just have to repeat it for our 'n' features:

repeat until convergence: {

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

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

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

...

}

In general, the gradient descent for multiple variables can be represented by,

For n features,


```
repeat until convergence: {  
   $\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) \cdot x_j^{(i)}$     for j := 0...n  
}
```

Gradient Descent in Practice I – Feature Scaling

- We can speed up gradient descent by having each of our input values in roughly the same range.
- This is because θ will descend quickly on small ranges and slowly on large ranges, and so will oscillate inefficiently down to the optimum when the variables are very uneven.

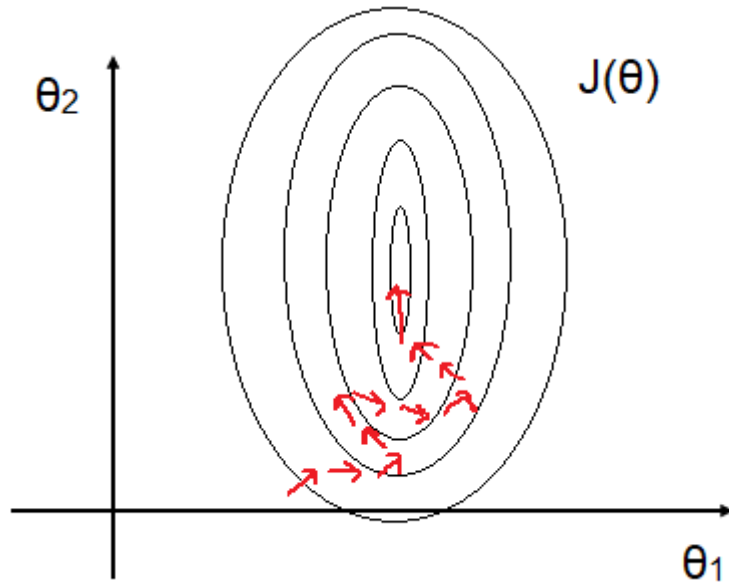
Example:

x_1 = size (0-2000 sq. feet)  Range is in thousands

x_2 = no. of bedrooms (1-5)  Range is in ones

Hence, $\text{range}(x_1) \gg \text{range}(x_2)$

If we plot the contours of the cost function $J(\theta)$ which is a function of θ_1 (size) and θ_2 (no. of bedrooms), it will look something like this.



- Plot consists of tall and narrow ellipses which is not very symmetric.
- Running gradient descent on this may end up taking a very long time and can oscillate back and forth.

Method I: Feature Scaling

To make the gradient descent run more efficiently, we can scale down the features so that their ranges are approximately matched.

Formula: (Input value) / (maximum value)

Example:

(i) $x_1 = (\text{size in sq. feet}) / 2000$

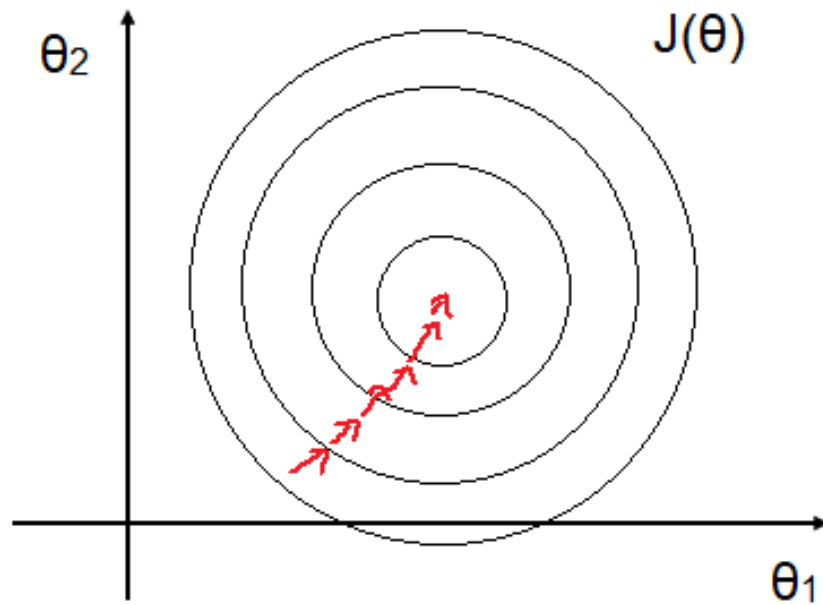
(ii) $x_2 = (\text{no. of bedrooms}) / 5$

This brings down the range roughly in between $-1 \leq x_i \leq 1$

It is not exactly necessary to have the range only in between -1 and 1.

We just have to take care that the difference between the ranges of the features is not too big.

The contour plot for the cost function looks something like this after scaling it down.



- The plot looks much more symmetric and circular now.
- Running gradient descent on this gives a much more direct path to the global minimum rather than taking a much more convoluted path.

Method II: Mean Normalization

Mean Normalization is another method to reduce the range of features so that they are approximately matched.

It is given by:

$$x_i = (x_i - \mu_i) / s_i$$

where, μ_i = average of all the values of feature x_i
 s_i = standard deviation (max-min)

Example:

- (i) $x_1 = (\text{size} - 1000) / 2000$
- (ii) $x_2 = (\text{\#bedrooms} - 2) / 5$

This brings down the range roughly in between $-0.5 \leq x_i \leq 0.5$

Suppose you are using a learning algorithm to estimate the price of houses in a city. You want one of your features x_i to capture the age of the house. In your training set, all of your houses have an age between 30 and 50 years, with an average age of 38 years. Which of the following would you use as features, assuming you use feature scaling and mean normalization?

- $x_i = \text{age of house}$
- $x_i = (\text{age of house})/50$
- $x_i = (\text{age of house}-38)/50$
- $x_i = (\text{age of house}-38)/20$

Gradient Descent in Practice II – Learning Rate

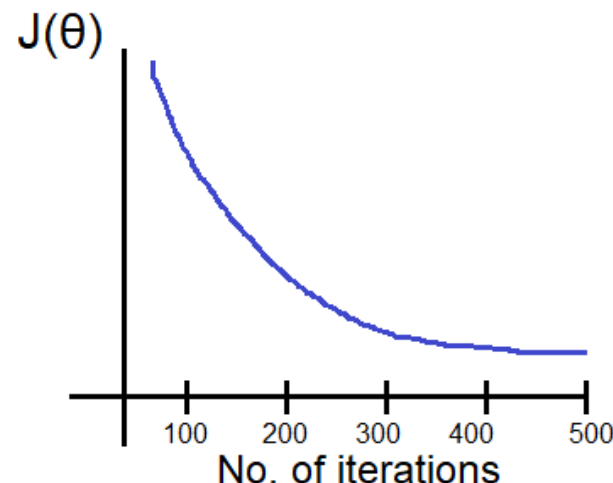
Gradient Descent for multiple variables is given by,

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

How do we know if our gradient descent is working correctly or not?

Method I: “Debugging”

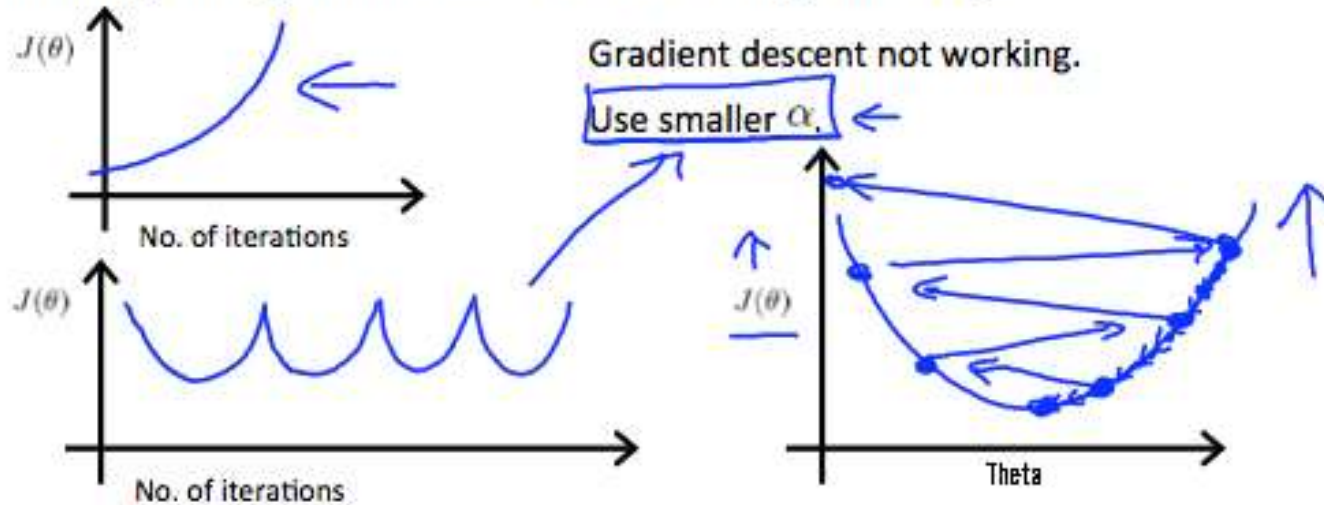
- Plot a graph of number of iterations (x-axis) vs cost function $J(\theta)$ (y-axis)
- If $J(\theta)$ increases with no. of iterations, we probably need to decrease α



Method II: “Automatic Convergence Test”

- We find whether $J(\theta)$ is converging or not by setting some threshold value, let's say, 'E', for each iteration.
- For example, if value of $J(\theta)$ decreases by $E=10^{-3}$ for each successive iteration, then we say that our gradient descent is working correctly.
- This method is generally not used because setting the value of E is another additional task.

Making sure gradient descent is working correctly.

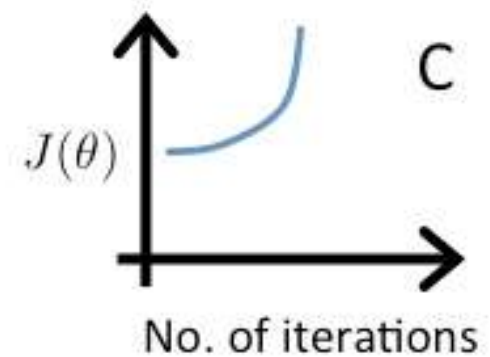
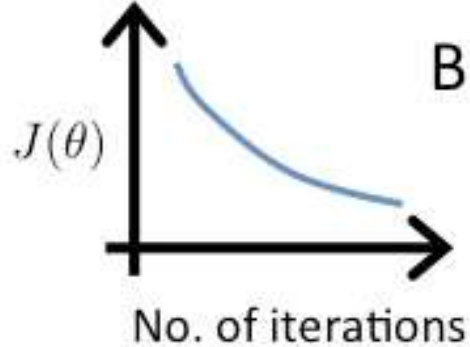
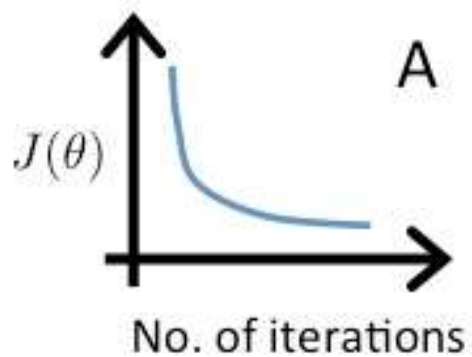


- For sufficiently small α , $J(\theta)$ should decrease on every iteration.
- But if α is too small, gradient descent can be slow to converge.

Summary:

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

Suppose a friend ran gradient descent three times, with $\alpha = 0.01$, $\alpha = 0.1$ and $\alpha = 1$, and got the following three plots (labelled A, B and C).



Which plot corresponds to which value of α ?

- A is $\alpha = 0.01$, B is $\alpha = 0.1$, C is $\alpha = 1$
- A is $\alpha = 0.1$, B is $\alpha = 0.01$, C is $\alpha = 1$
- A is $\alpha = 1$, B is $\alpha = 0.01$, C is $\alpha = 0.1$
- A is $\alpha = 1$, B is $\alpha = 0.1$, C is $\alpha = 0.01$

Polynomial Regression

Housing prices prediction

Suppose the hypothesis depends upon two features – frontage and depth.

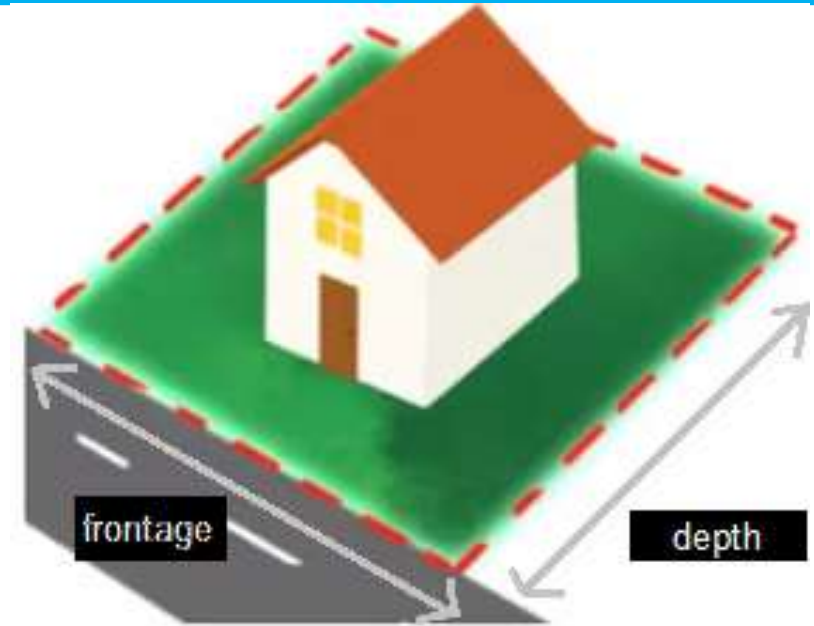
$$h_{\theta}(x) = \theta_0 + \theta_1 \times \text{frontage} + \theta_2 \times \text{depth}$$

We can **combine** multiple features into one.

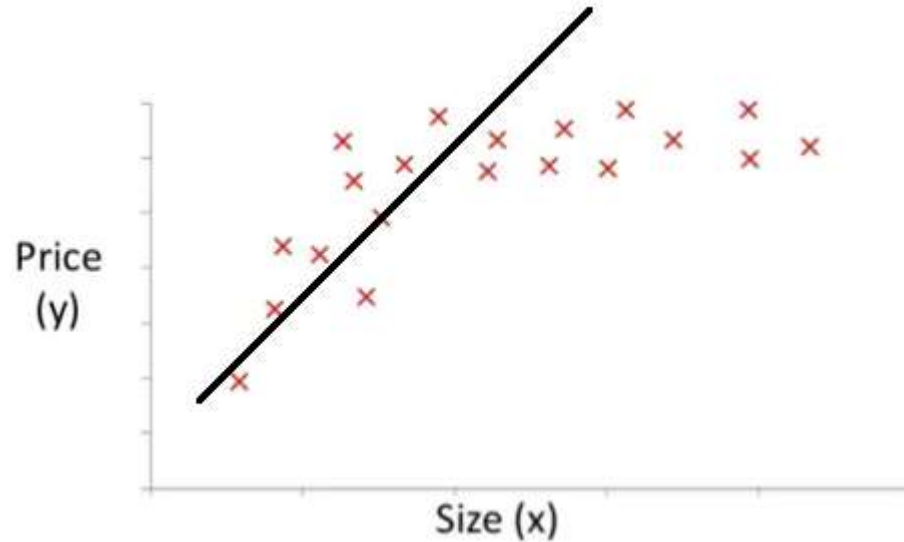
For example, we can combine x_1 and x_2 into a new feature x_3 by taking $x_1 \cdot x_2$.

Therefore, $x_3 = \text{Area} = \text{frontage} \times \text{depth}$

$$h_{\theta}(x) = \theta_0 + \theta_1 \times \text{area}$$

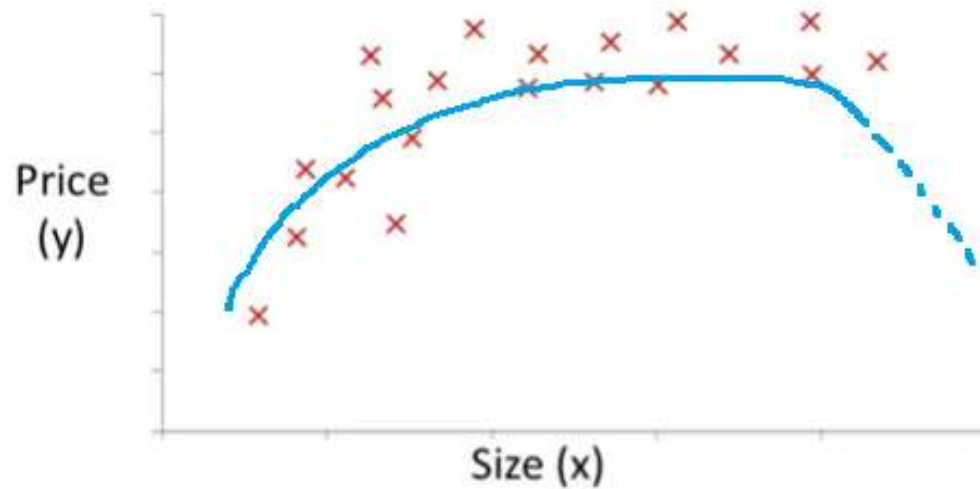


Our hypothesis function need not be linear (a straight line) if that does not fit the data well.



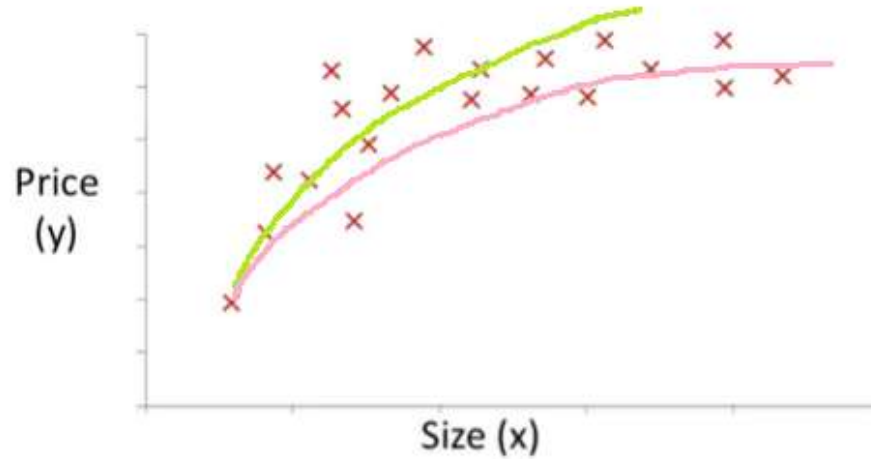
Linear Regression Model: $h_{\theta}(x) = \theta_0 + \theta_1 x_1$

- This function does not fit data quite well.
- We can **change the behavior or curve** of our hypothesis function by making it a quadratic, cubic or square root function (or any other form).



Square Function Model: $h_{\theta}(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_1^2$

- Fits the data better compared to Linear Regression Model.
- Creates a new feature $x_2 = x_1^2$
- Drawback: After some time, the price decreases as the size increases.



Cubic Function Model:

$$h_{\theta}(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \theta_3 x_1^3$$

- Creates two new features: $x_2 = x_1^2$ and $x_3 = x_1^3$

Square Root Function Model:

$$h_{\theta}(x) = \theta_0 + \theta_1 x_1 + \theta_2 \sqrt{x_1}$$

- Creates a new feature: $x_2 = \sqrt{x_1}$

One important thing to keep in mind is, if you choose your features this way then feature scaling becomes very important.

Suppose you want to predict a house's price as a function of its size. Your model is $h_{\theta}(x) = \theta_0 + \theta_1(\text{size}) + \theta_2\sqrt{(\text{size})}$.

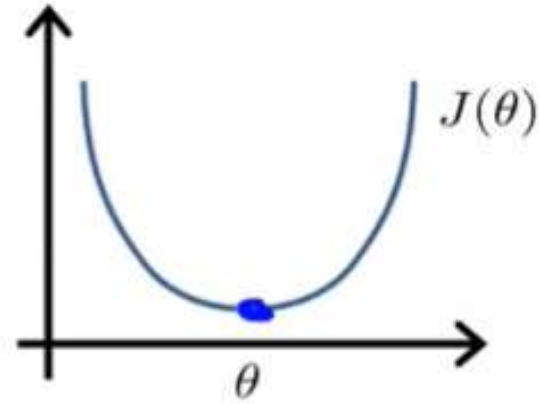
Suppose size ranges from 1 to 1000 (feet²). You will implement this by fitting a model $h_{\theta}(x) = \theta_0 + \theta_1x_1 + \theta_2x_2$.

Finally suppose you want to use feature scaling (without mean normalization). Which of the following choices for x_1 and x_2 should you use? (Note: $\sqrt{1000} \approx 32$)

- $x_1 = \text{size}, x_2 = 32\sqrt{(\text{size})}$
- $x_1 = 32(\text{size}), x_2 = \sqrt{(\text{size})}$
- $x_1 = \text{size}/1000, x_2 = \sqrt{(\text{size})} / 32$
- $x_1 = \text{size}/32, x_2 = \sqrt{(\text{size})}$

Example:

$$J(\theta) = a\theta^2 + b\theta + c$$



How do we find the value of θ that reaches to global minimum?

maxima-minima using derivative

- Differentiate $J(\theta)$ w.r.t θ
- Equate $\frac{d}{d\theta} J(\theta) = 0$
- Solve for θ

Normal Equation

- Normal Equation method is an alternative to Gradient Descent algorithm
- Gives the optimum value of θ in just one step, thereby eliminating iterations

Example:

Size (sq. feet) (x_1)	No. of bedrooms (x_2)	No. of floors (x_3)	Age of home in years (x_4)	Price (\$1000) (y)
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178

We can find the optimum value of θ by using Normal Equation method.

Step 1: Add a new column of feature x_0 with values of each row equal to 1.

(x_0)	Size (sq. feet) (x_1)	No. of bedrooms (x_2)	No. of floors (x_3)	Age of home in 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

Step 2: Create the feature matrix X .

$$X = \begin{bmatrix} 1 & 2104 & 5 & 1 & 45 \\ 1 & 1416 & 3 & 2 & 40 \\ 1 & 1534 & 3 & 2 & 30 \\ 1 & 852 & 2 & 1 & 36 \end{bmatrix}$$

X is also called "Design Matrix"

Step 3: Create the output matrix Y.

$$y = \begin{bmatrix} 460 \\ 232 \\ 315 \\ 178 \end{bmatrix}$$

Step 4: Compute X^T

Step 5: Compute $\theta = (X^T X)^{-1} X^T y$

- This equation is called “**Normal Equation**” that gives the optimum value of θ in just one step.
- Feature Scaling is **not required** in Normal Equation

Comparing Gradient Descent & Normal Equation

Gradient Descent	Normal Equation
Need to choose alpha	No need to choose alpha
Needs many iterations	No need to iterate
Time complexity: $O(kn^2)$	Time complexity: $O(n^3)$
Works well when n is large	Slow if n is large

Time complexity of normal equation is $O(n^3)$

If we have a large number of features, normal equation becomes slow

In practice, when $n > 10,000$, iterative method is preferred over normal equation

Suppose you have the training set in the table below

age (x_1)	height in cm (x_2)	weight in kg (y)
4	89	16
9	124	28
5	103	20

You would like to predict a child's weight as a function of his age and height with the model:
 $\text{weight} = \theta_0 + \theta_1(\text{age}) + \theta_2(\text{height})$

What are X and y ?

$$X = \begin{bmatrix} 4 & 89 \\ 9 & 124 \\ 5 & 103 \end{bmatrix}, y = \begin{bmatrix} 16 \\ 28 \\ 20 \end{bmatrix}$$

$$X = \begin{bmatrix} 1 & 4 & 89 \\ 1 & 9 & 124 \\ 1 & 5 & 103 \end{bmatrix}, y = \begin{bmatrix} 16 \\ 28 \\ 20 \end{bmatrix}$$

$$X = \begin{bmatrix} 4 & 89 & 1 \\ 9 & 124 & 1 \\ 5 & 103 & 1 \end{bmatrix}, y = \begin{bmatrix} 16 \\ 28 \\ 20 \end{bmatrix}$$

$$X = \begin{bmatrix} 1 & 4 & 89 \\ 1 & 9 & 124 \\ 1 & 5 & 103 \end{bmatrix}, y = \begin{bmatrix} 16 \\ 28 \\ 20 \end{bmatrix}$$



Non-invertibility of Normal Equation

What if inverse of $X^T X$ does not exist?

Common causes of non-invertibility of $X^T X$ are:

- Redundant features, where two features are very closely related i.e. they are linearly dependent

Example:

x_1 = size (in feet²)

x_2 = size (in m²)

$$[1\text{m} = 3.28 \text{ feet} \Rightarrow x_1 = (3.28)^2 x_2]$$

- Too many features (e.g. $m \leq n$)
In this case, delete some features or use “**regularization**”