LECTURE 3

AGENDA

- Revisit Simple Linear Regression
- Multivariate Regression
- Polynomial Regression
- Feature Scaling & Contour Plots
- Practical Sessions

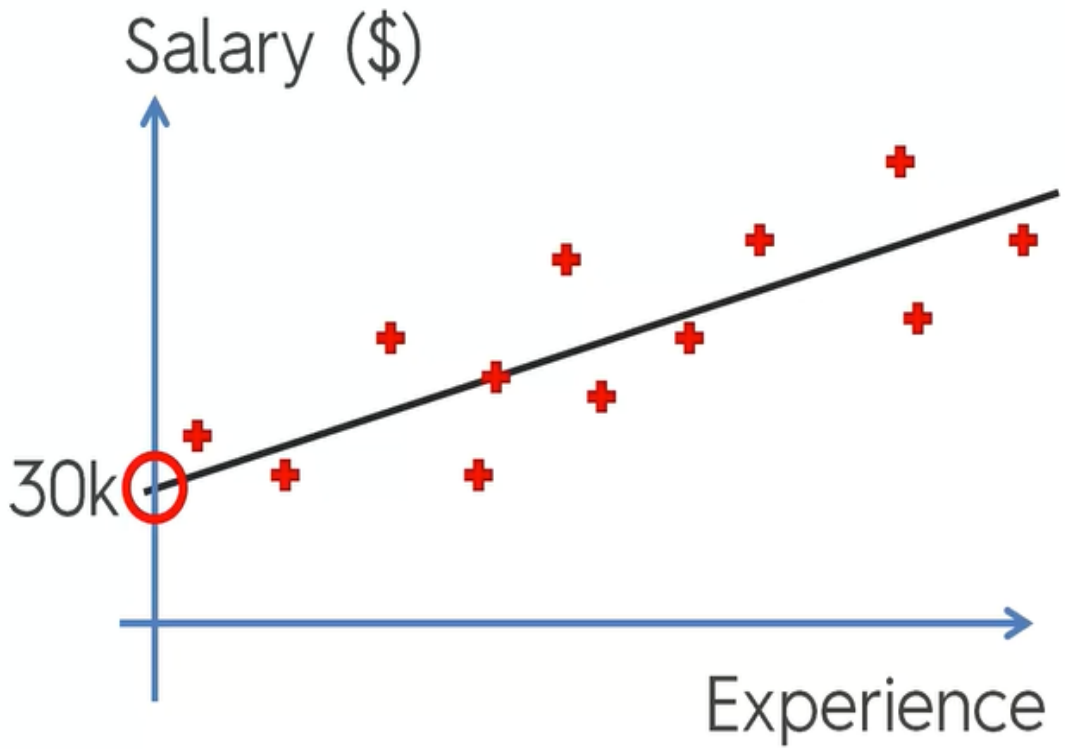
RECAP

SIMPLE LINEAR REGRESSION

- It is the type of Regression when output is dependant on single variable.
- Eg: Price of a House(Y) depends only on Area of the house (x)
- Model is always a line with two parameters b0 and b1

$$y = b_0 + b_1 * x_1$$

Simple Linear Regression:

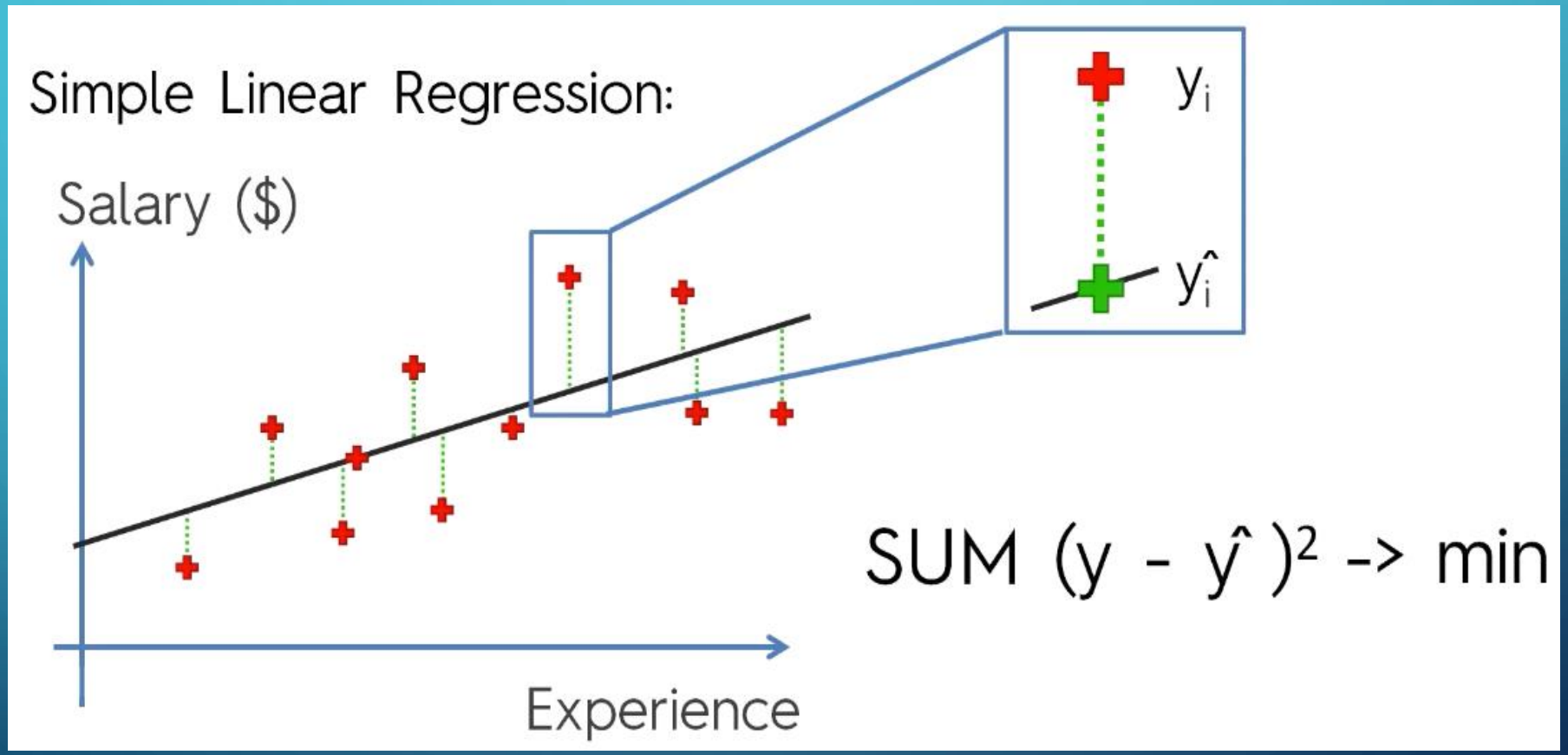


$$y = b_0 + b_1 * x$$

Salary = b_0 + b_1 *Experience

COST FUNCTION

- Mean Squared Errors (MSE): Aim is to minimise this cost function



OBJECTIVE OF COST FUNCTION

- Quantify the errors in the prediction
- Minimise the cost function to find the optimal parameters
- Also used to monitor the model performance
- Only way to visualise the model in higher dimensions

GRADIENT DESCENT ALGORITHM

OBJECTIVE:

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

Parameters: θ_0, θ_1

Cost Function: $J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^{2\,2}$

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

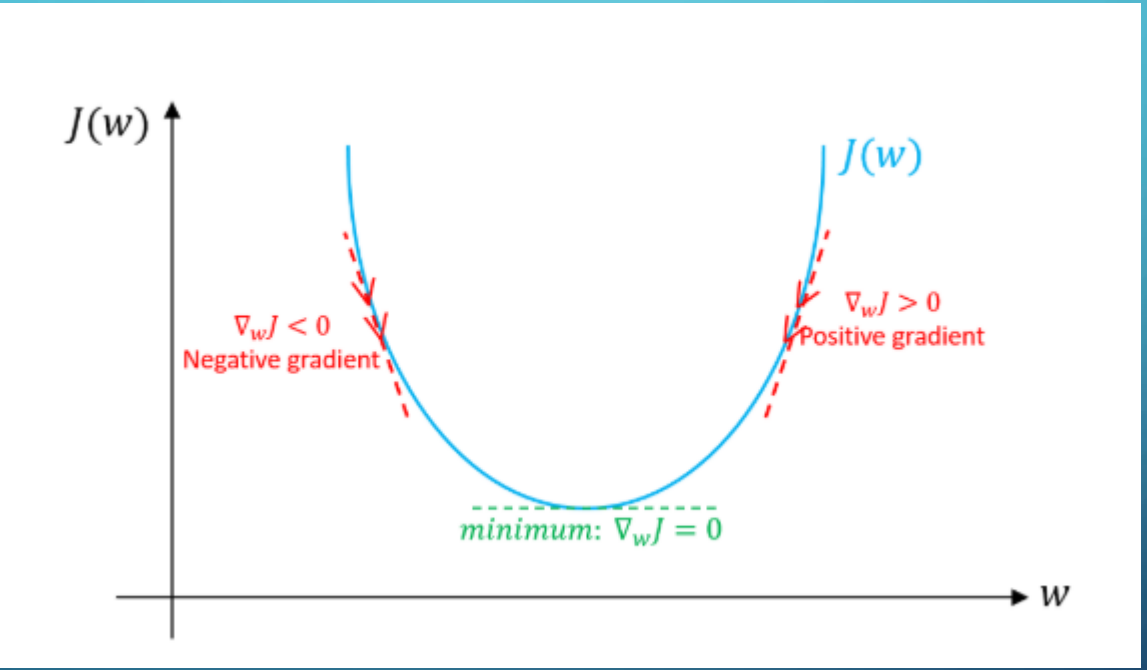
Have some function $J(\theta_0, \theta_1)$

Want $\min_{\theta_0, \theta_1} J(\theta_0, \theta_1)$

Outline:

- Start with some θ_0, θ_1
- Keep changing θ_0, θ_1 to reduce $J(\theta_0, \theta_1)$

until we hopefully end up at a minimum


$J(w)$
$J(w)$
$\nabla_w J < 0$
Negative gradient
$\nabla_w J > 0$
Positive gradient
$minimum$: $\nabla_w J = 0$
w

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 Model

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

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

Repeat until convergence

{

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

}

Advantage:

- As we approach minima, gradient descent will automatically take smaller steps. So, no need to decrease alpha over time.
- Gradient Descent can converge to minima even with fixed learning constant

MULTIVARIATE LINEAR REGRESSION

- Unlike Simple Linear Regression, here output depends on multiple input features
- Eg: Price of House (Y) depends on features like Area(x1), Number of Rooms(x2), Number of floors(x3), Distance from Highway(x4)
- Parameters will be $\theta 0, \theta 1, \theta 2, \theta 3, \theta 4$

Multiple features (variables).

Size (feet²)	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.

Regressions

Simple Linear Regression

$$y = b_0 + b_1 * x_1$$

Multiple Linear Regression

Dependent variable (DV) — Independent variables (IVs)

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

COST FUNCTION

- It remains same ie: Mean Squared Error
- However, cost function will now depend on multiple parameters :

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

- Now we have to minimise the cost function with respect to all the parameters for multiple features

Gradient descent:

Repeat {

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

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

NEW ALGORITHM FOR GRADIENT DESCENT

- n is the number of features

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 $j = 0, \ldots, n$)

$\}$

PARAMETER UPDATES:

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

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

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

- We will revisit again in the practical session

FEATURE SCALING

- Scaling the feature means bringing the values of all the features in the same range
- Eg: Area of the house(x1) values are much larger than values of Number of rooms(x2)

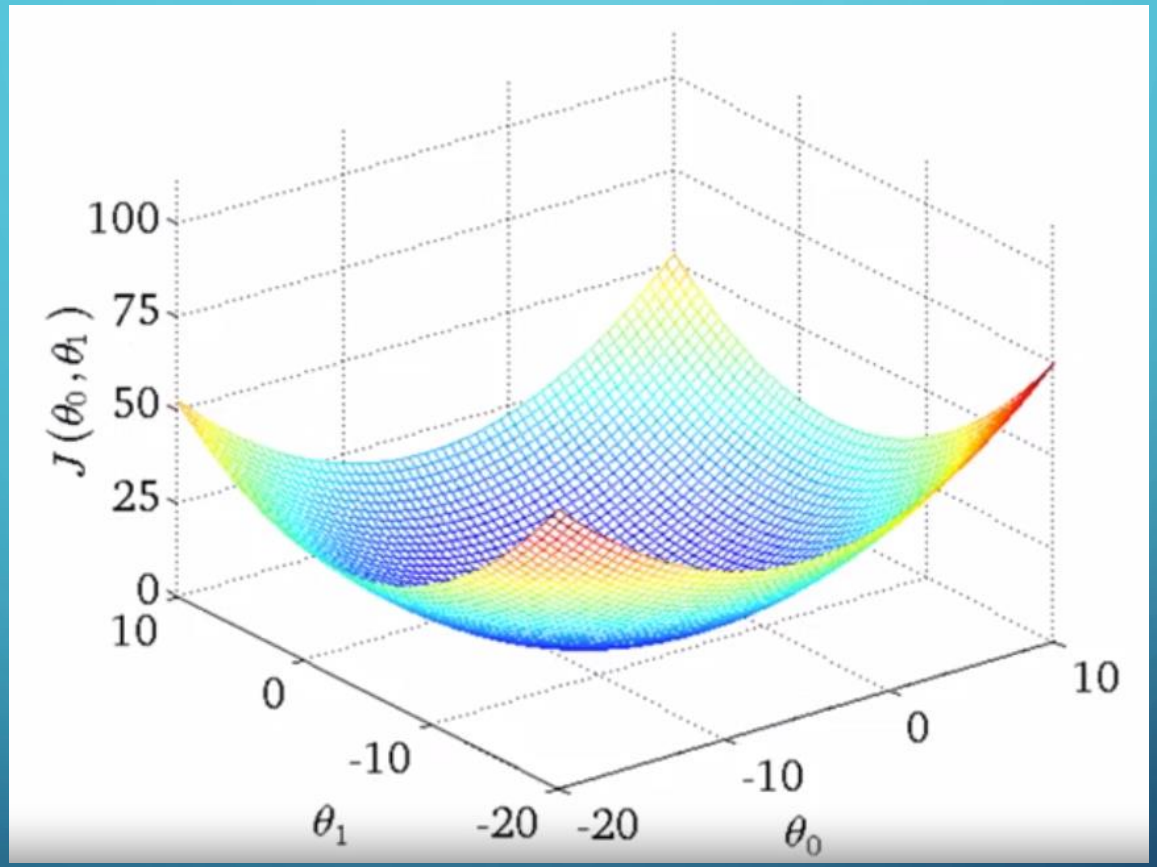
Feature Scaling

Idea: Make sure features are on a similar scale.

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

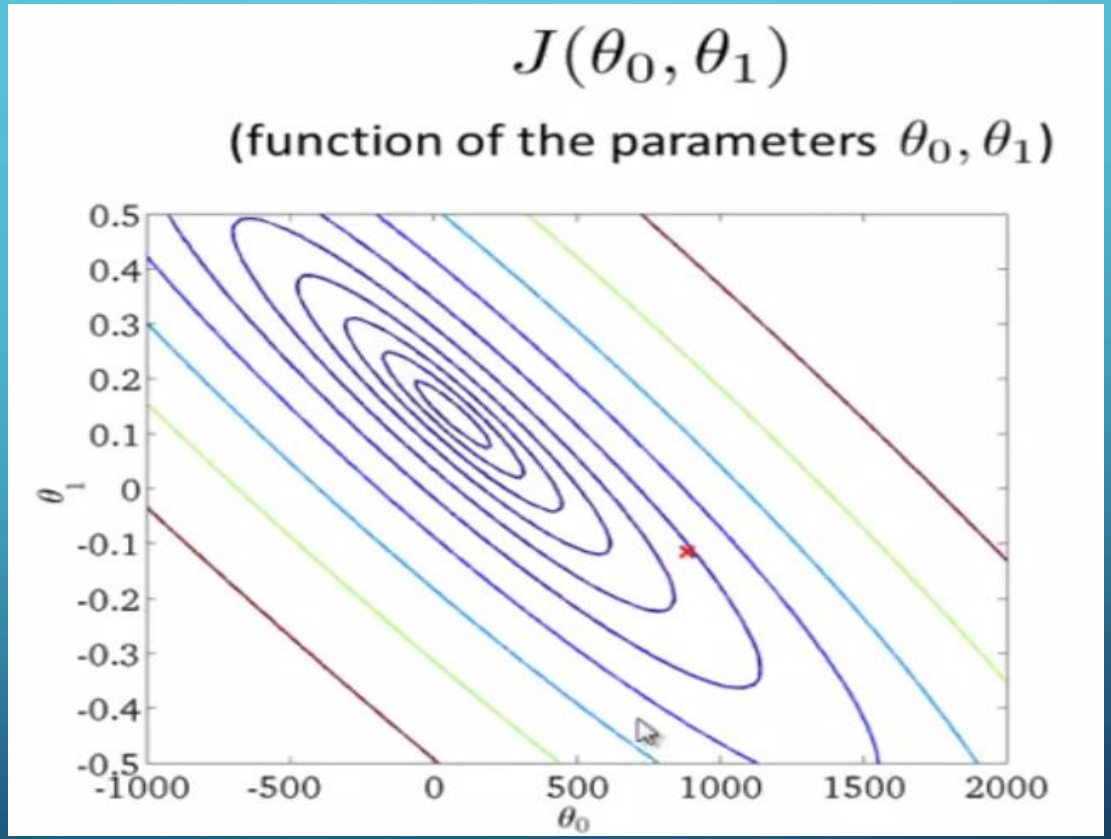
x_2 = number of bedrooms (1-5)

- Cost function for two features can be represented by a 3-D convex curve



Contour Plots:

- It is used to represent the 3-D curve on 2-D plane

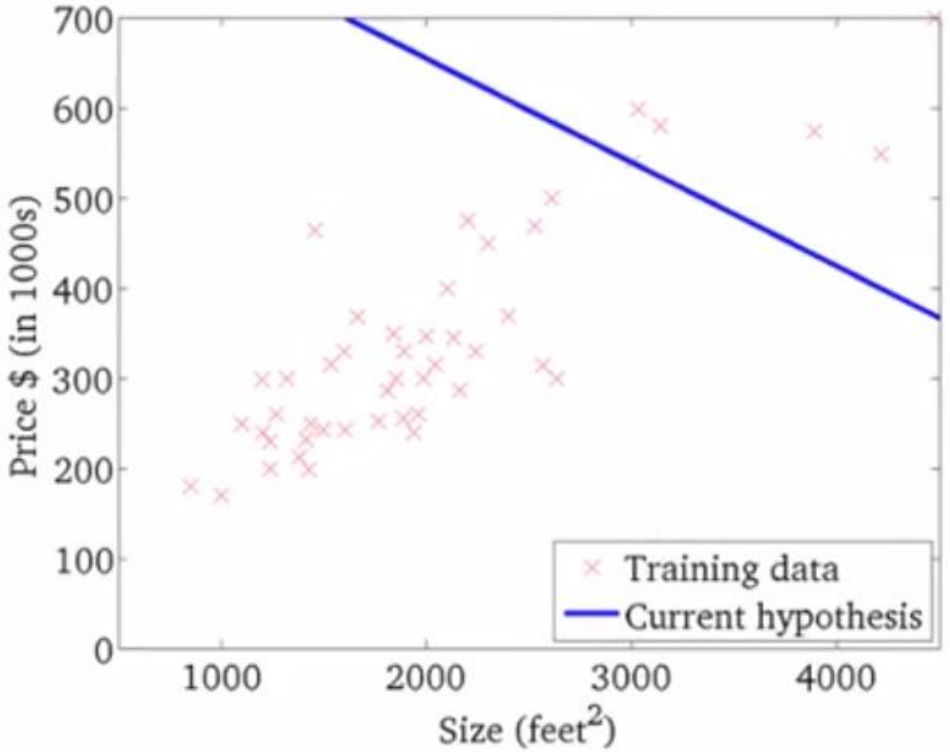


INTUITION OF CONTOUR PLOTS

- Lets see how contour plots can be used to visualise the cost function minimisation

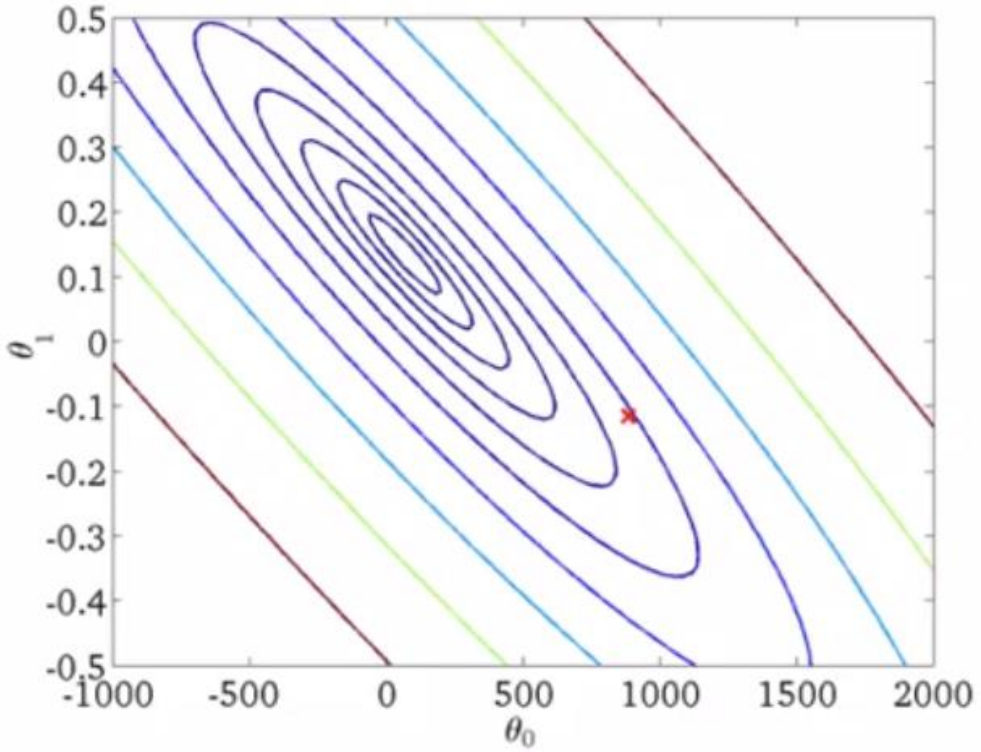
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



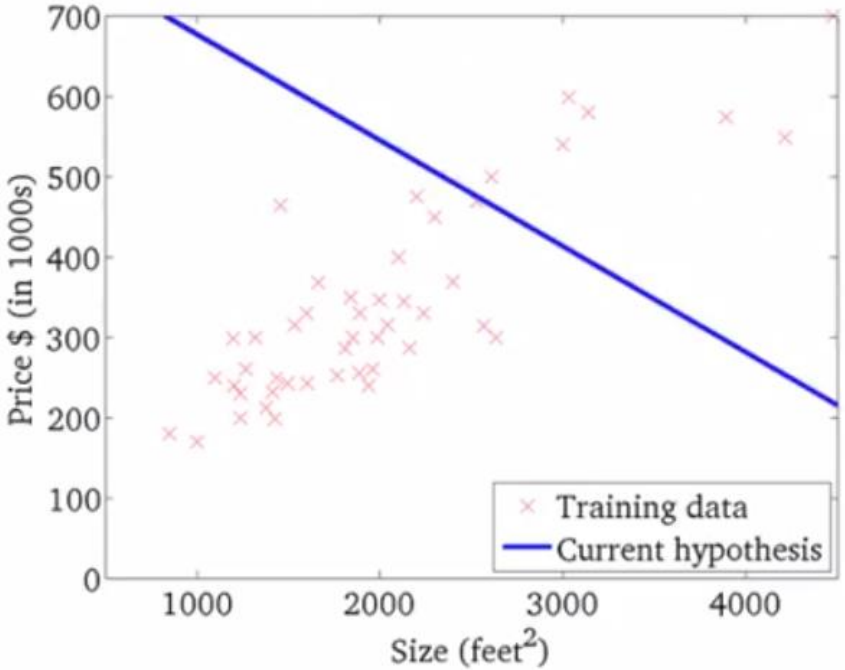
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



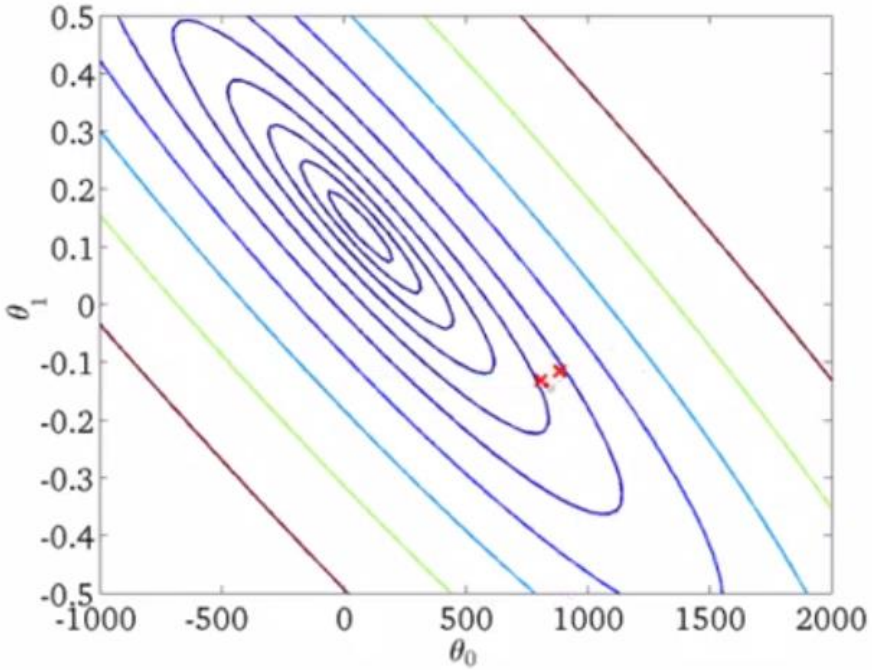
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



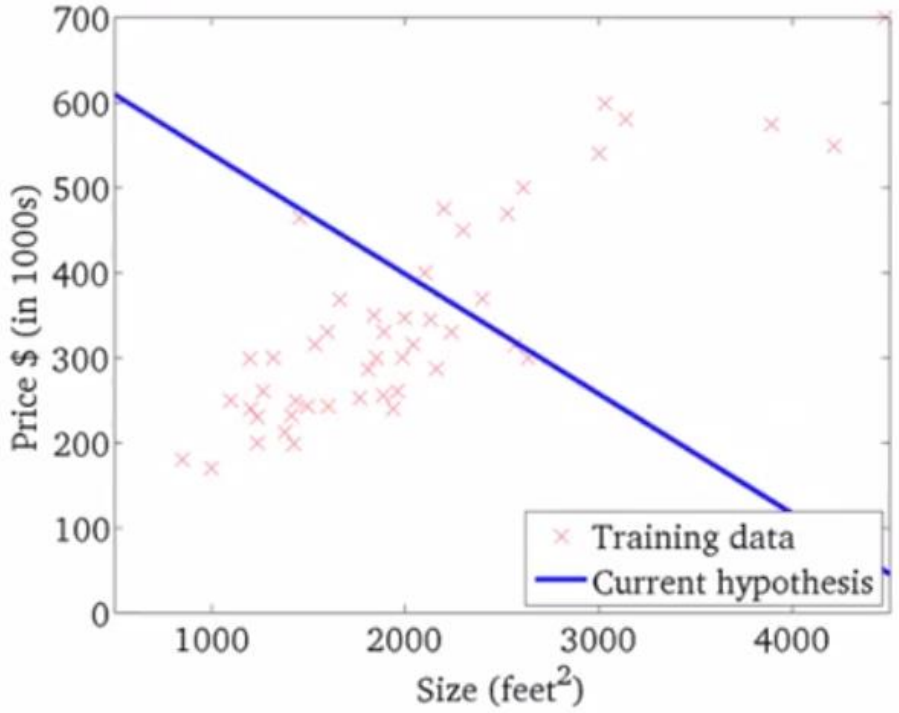
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



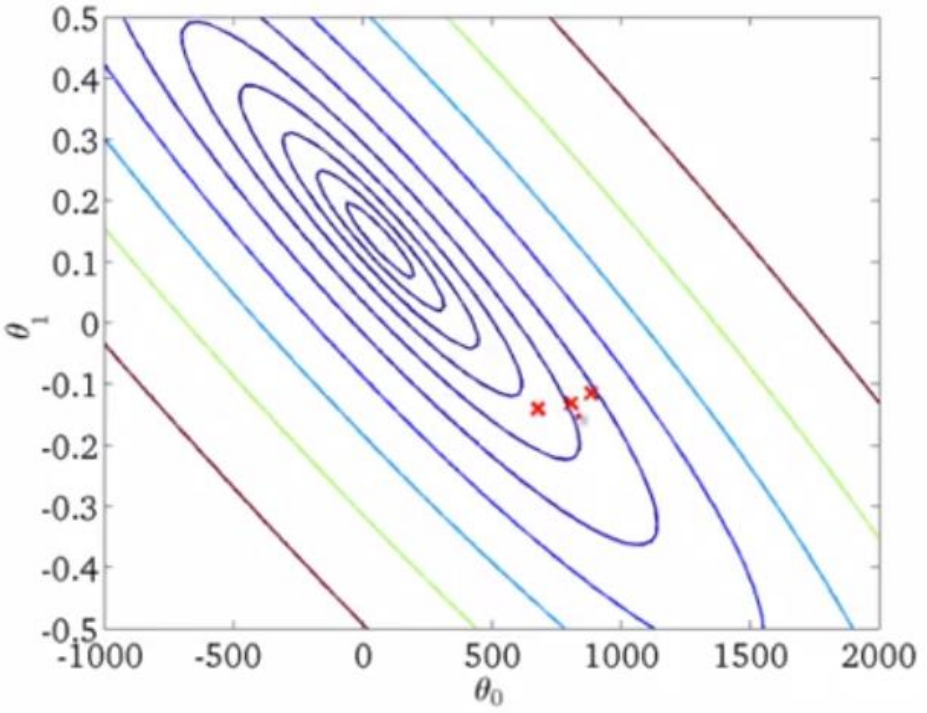
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



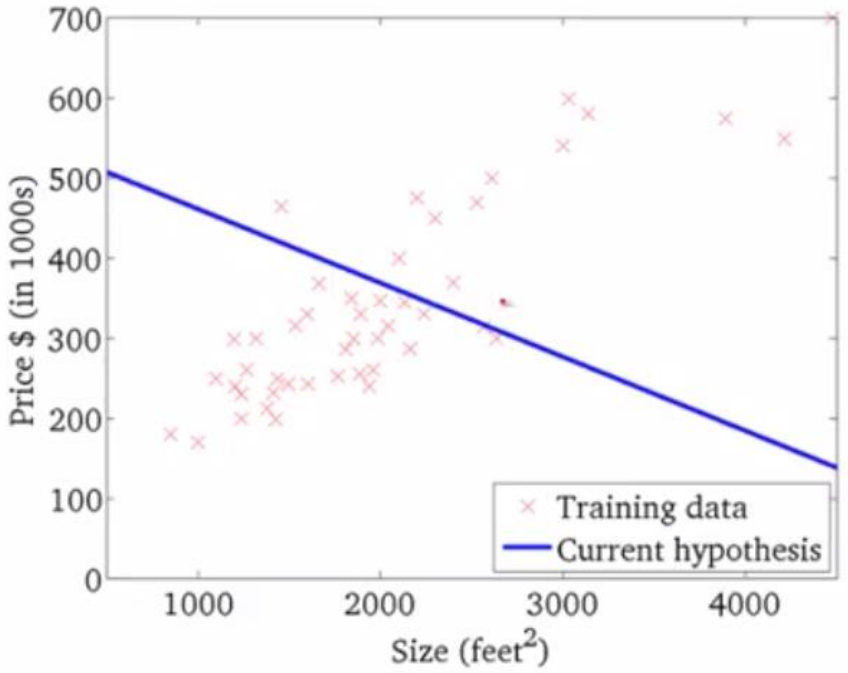
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



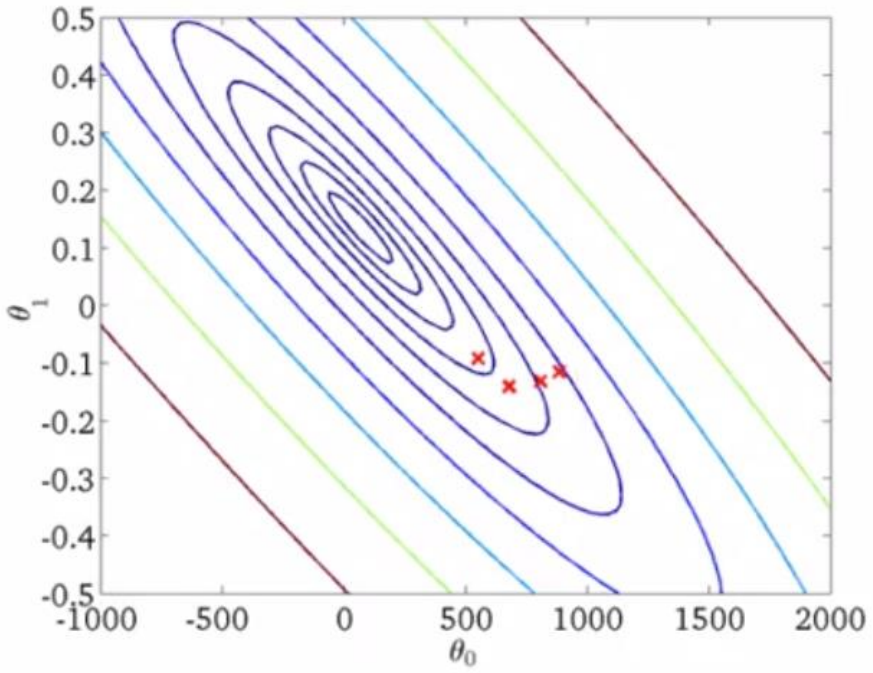
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



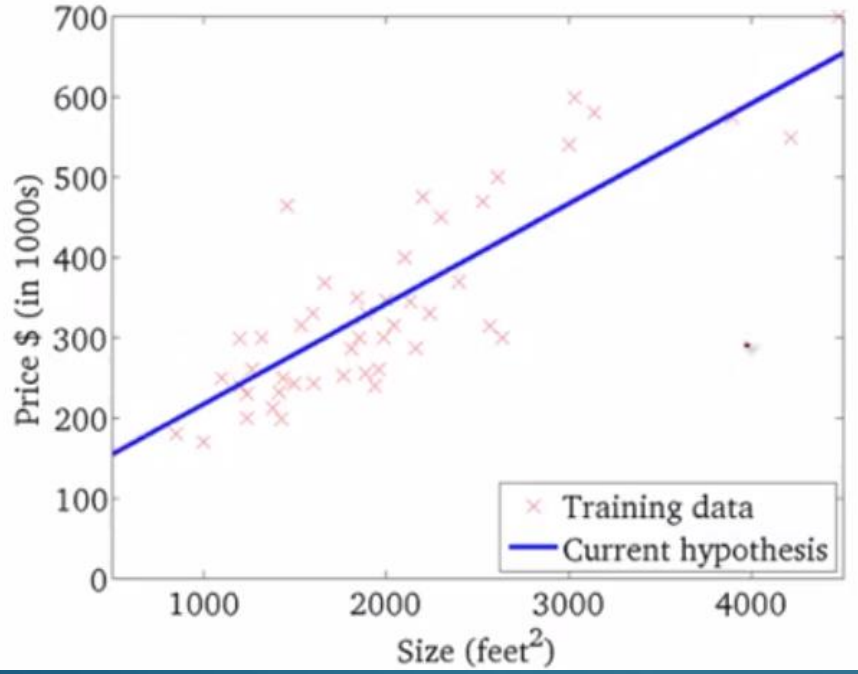
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



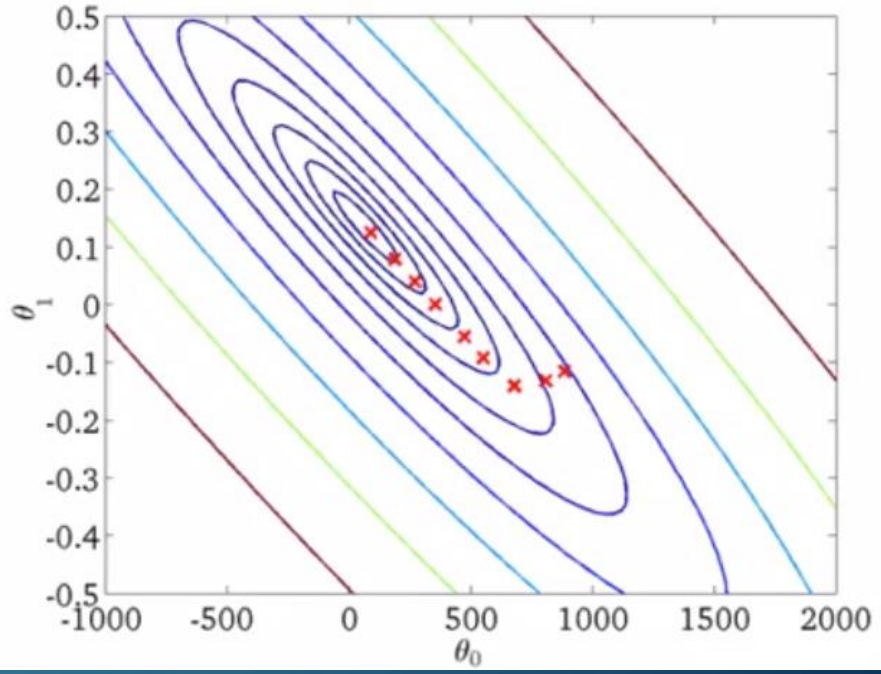
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



CONTOUR PLOT FOR FEATURE SCALING

EXAMPLE:

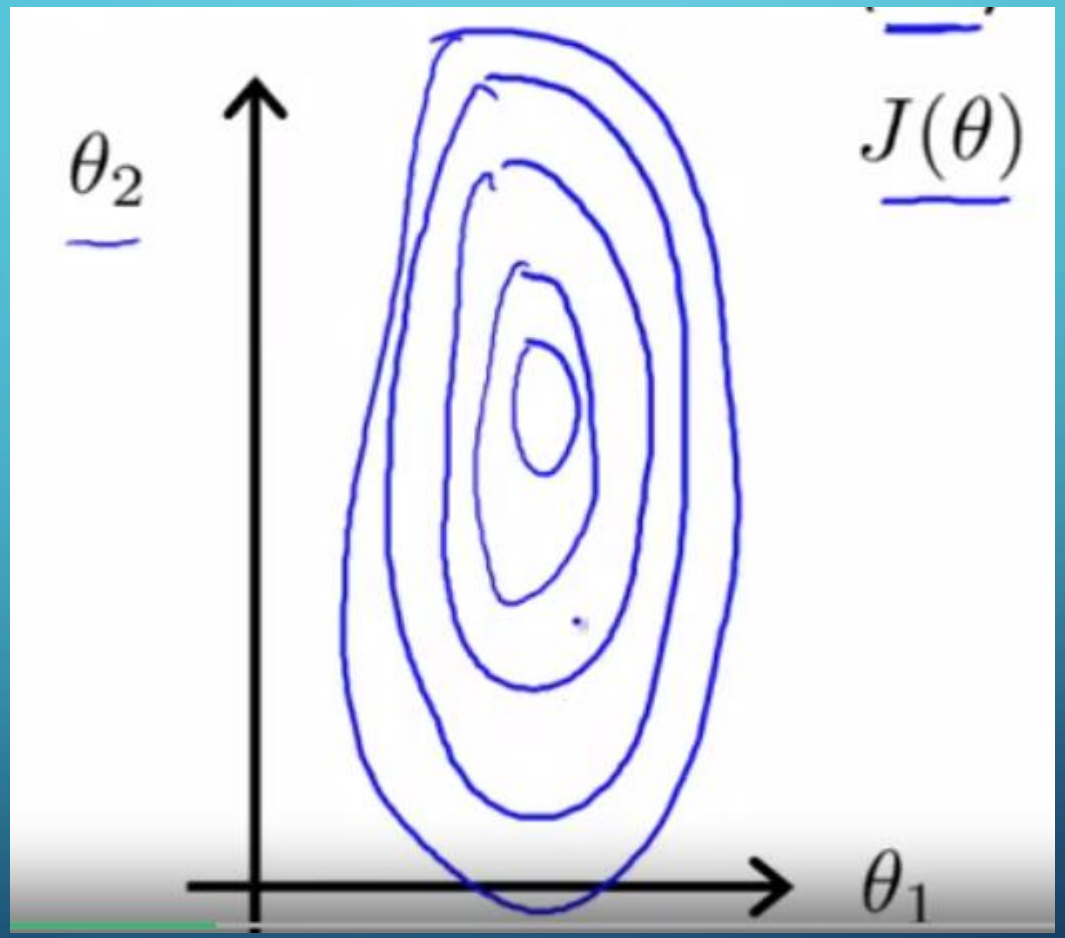
Feature Scaling

Idea: Make sure features are on a similar scale.

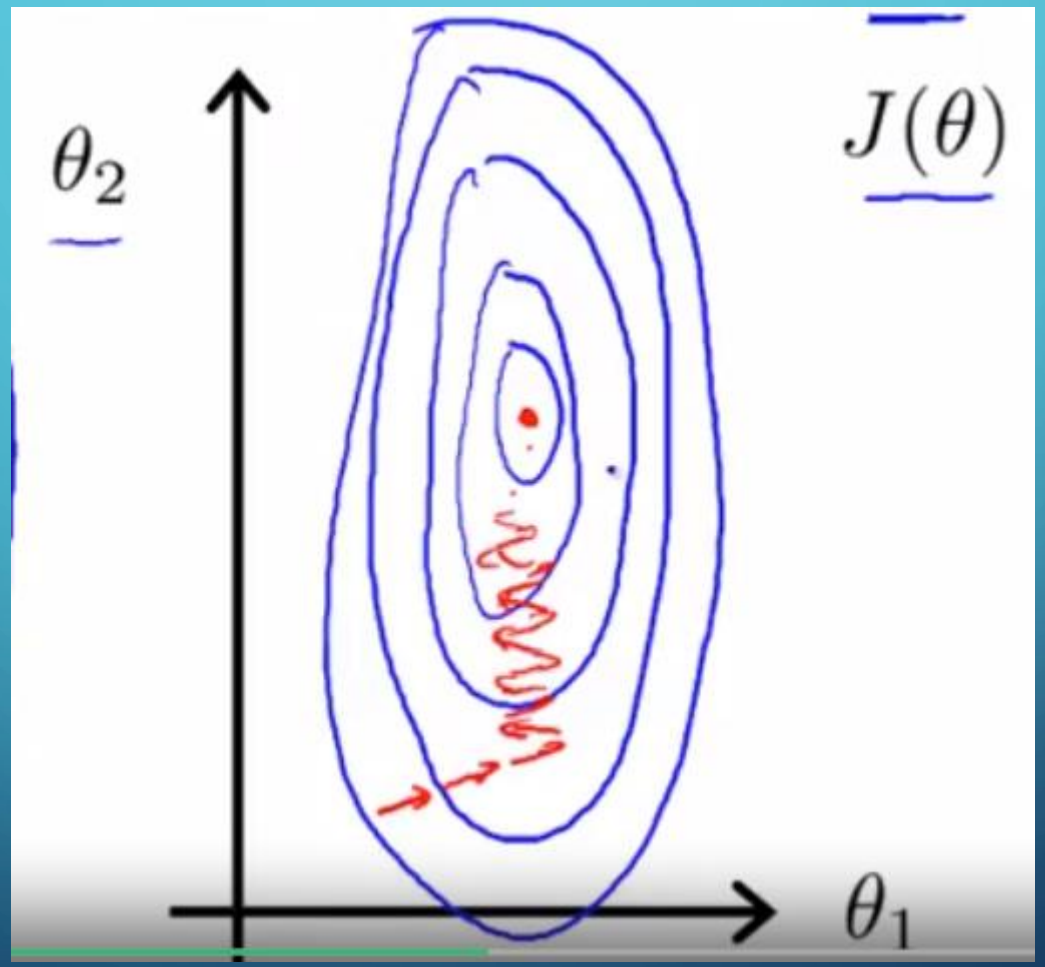
E.g. x_1 = size (0-2000 feet^2)

x_2 = number of bedrooms (1-5)

- Contour Plot without feature scaling:



- Number steps taken to reach minima will be too high

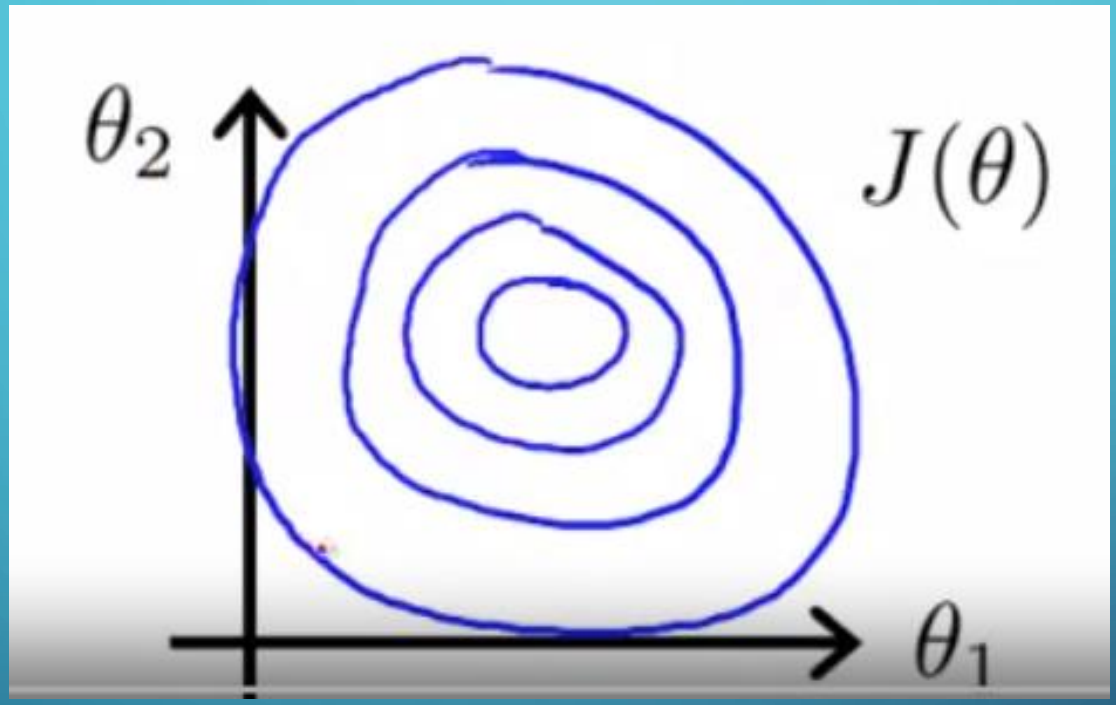


SCALING THE FEATURE VALUES:

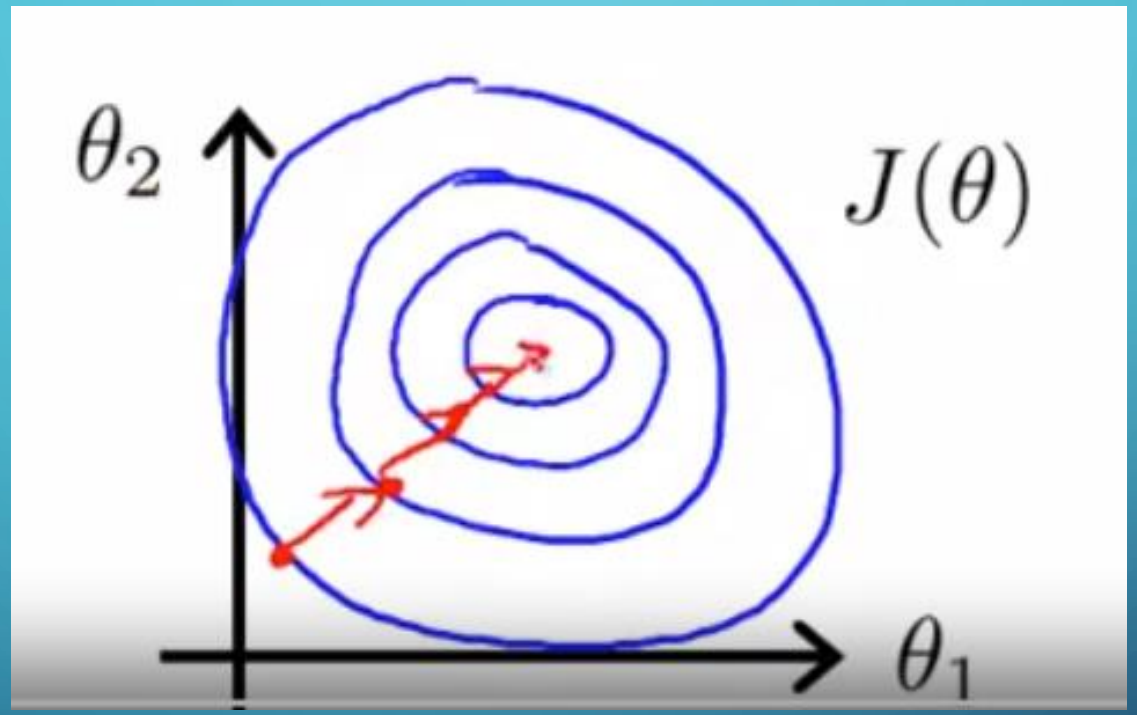
$$x_1 = \frac{\text{size (feet}^2\text{)}}{2000}$$

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

- Contour plot for scaled feature:



- Minima will be reached with lesser steps and more efficiently



THUMB RULE

- It is fine if feature values are in the range of -3 to +3
- In case of decimal: -1/3 to +1/3

Feature Scaling

Get every feature into approximately a $-1 \leq x_i \leq 1$ range.

MEAN NORMALIZATION

- μi here denotes the average value of a given feature's data
- Denominator has the range of the values for a feature i

Mean normalization

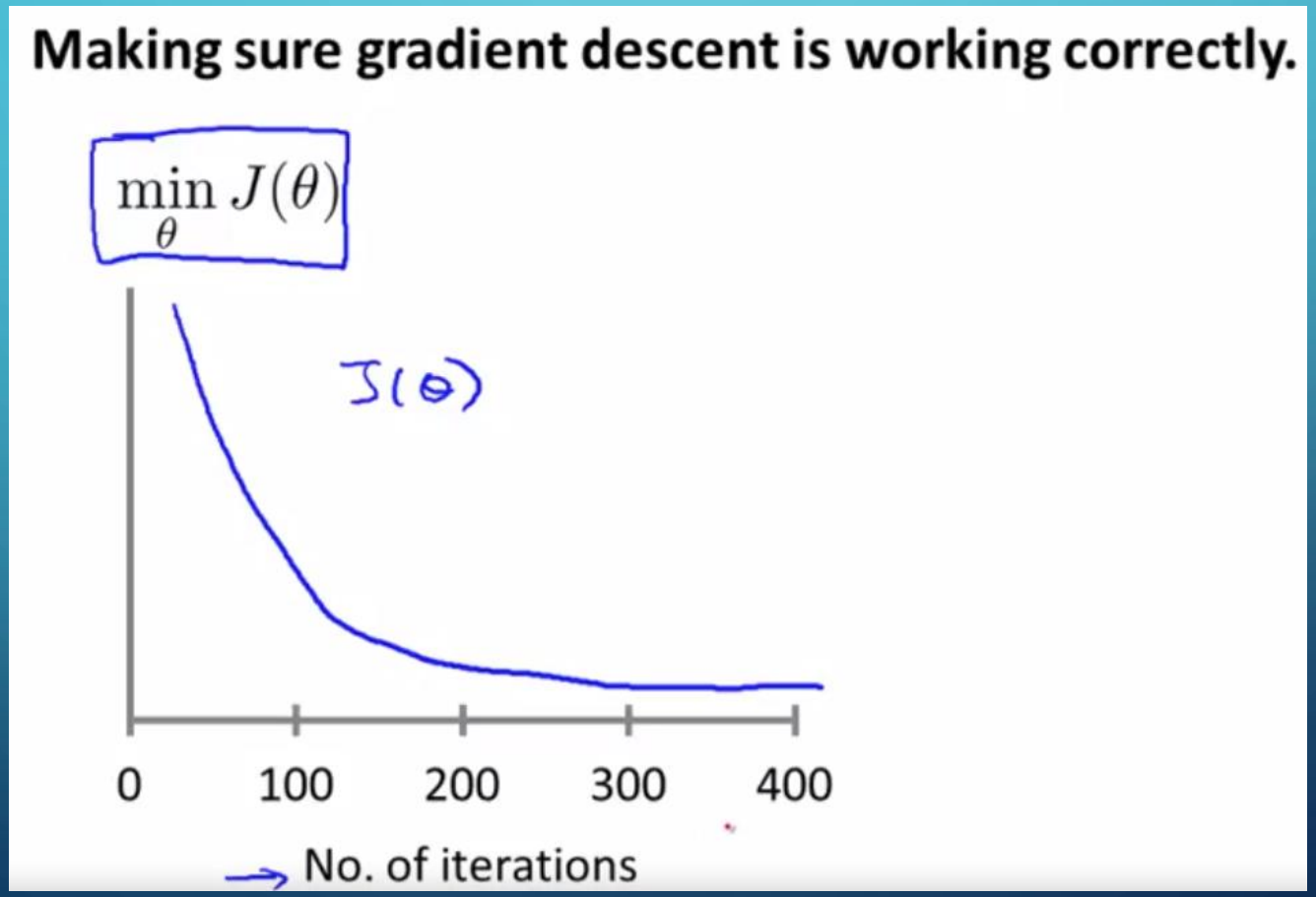
Replace x_i with $x_i - \mu_i$ to make features have approximately zero mean (Do not apply to $x_0 = 1$).

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

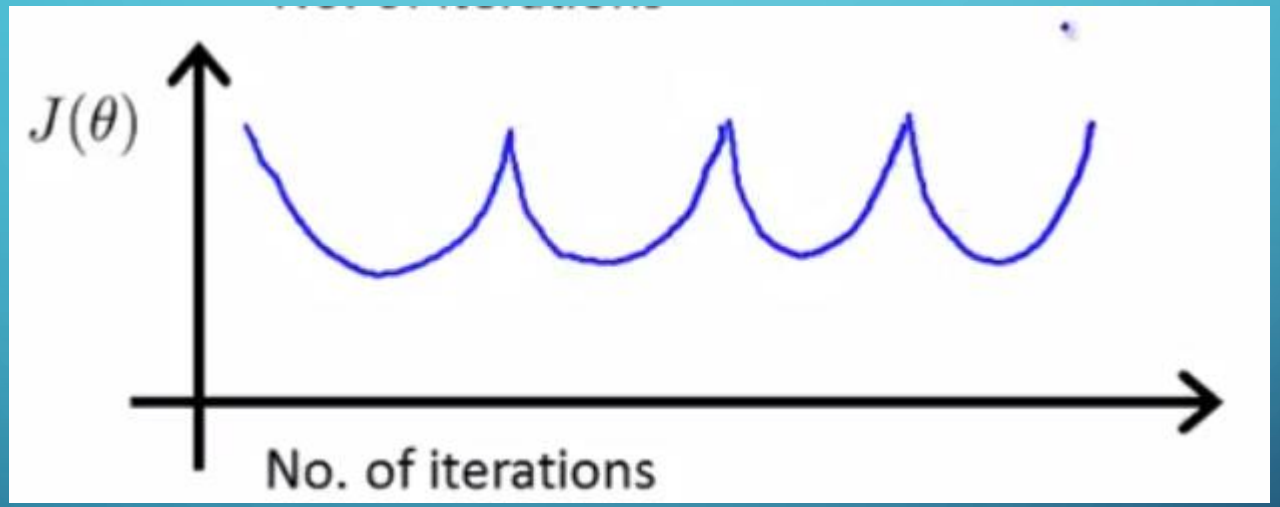
$x_2 = \frac{\#bedrooms - 2}{5}$

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

- Cost function value should decrease as number of iteration increases
- Declare the convergence when cost decreases less than 10^(-3) for an iteration



- Gradient Descent not working correctly. Reason can be the high value of learning rate α



Summary:

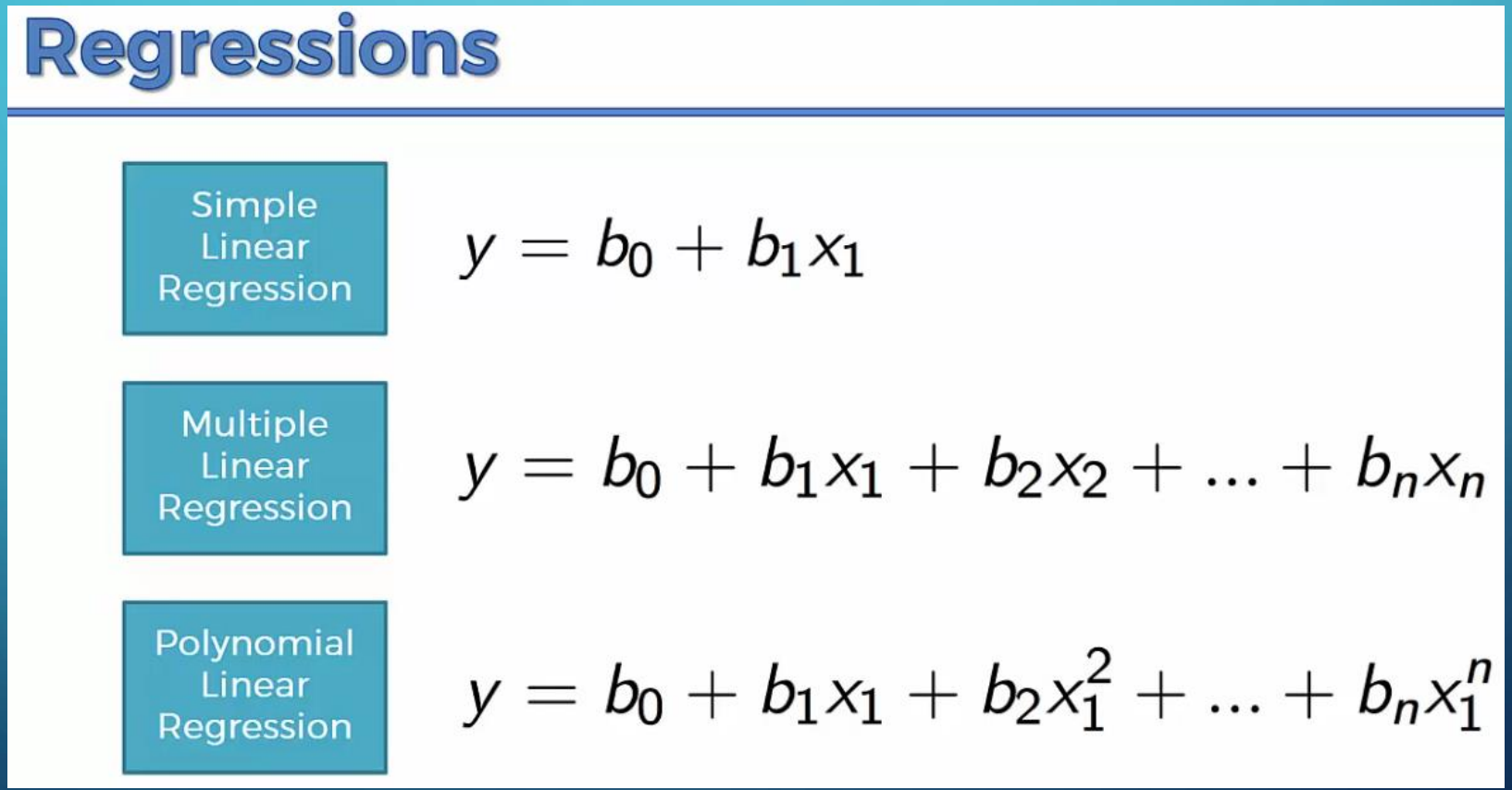
- If α is too small: slow convergence.
- If α is too large: $J(\theta)$ may not decrease on every iteration; may not converge.

To choose α, try

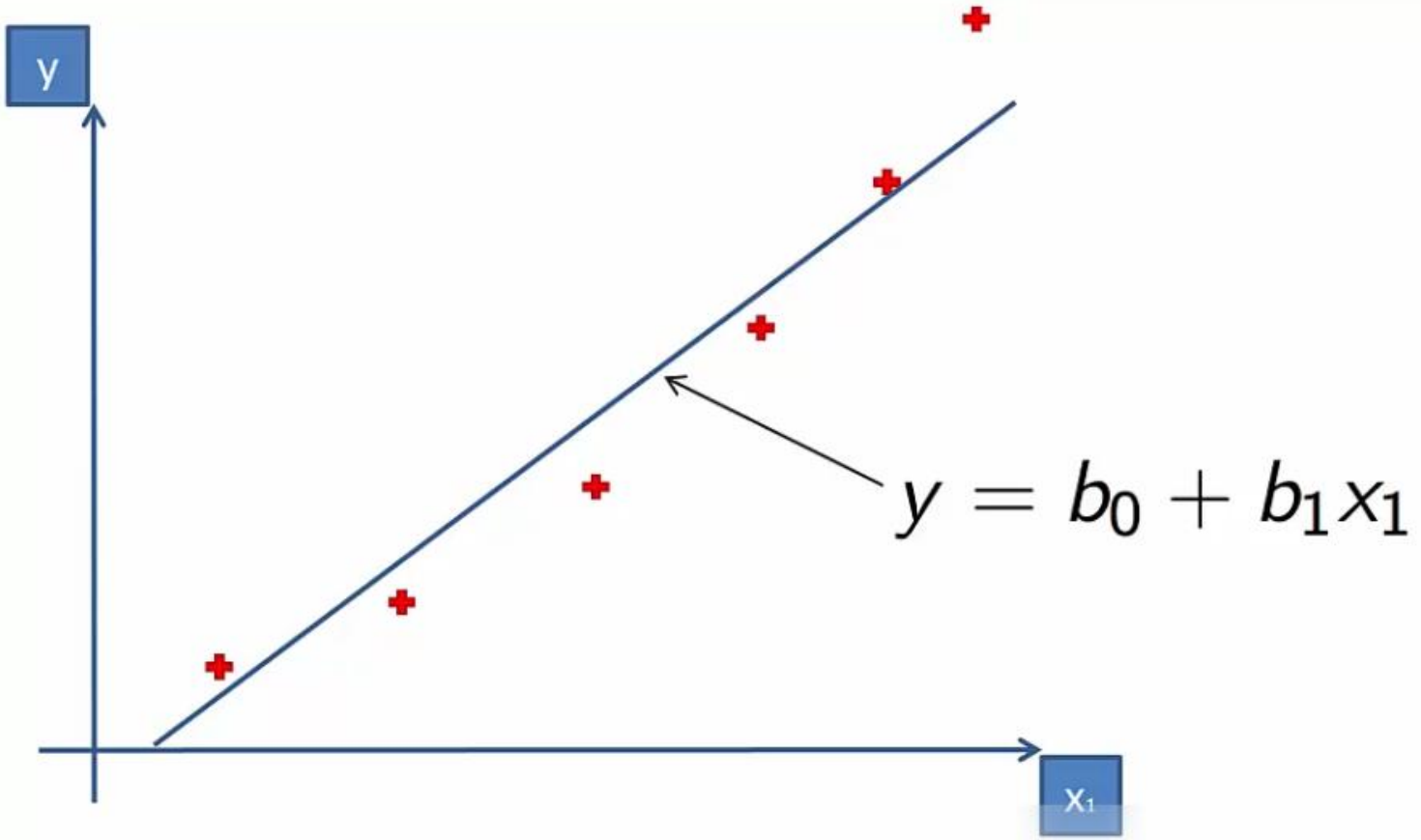
$$\ldots, 0.001, \quad , 0.01, \quad , 0.1, \quad , 1, \ldots$$

POLYNOMIAL REGRESSION

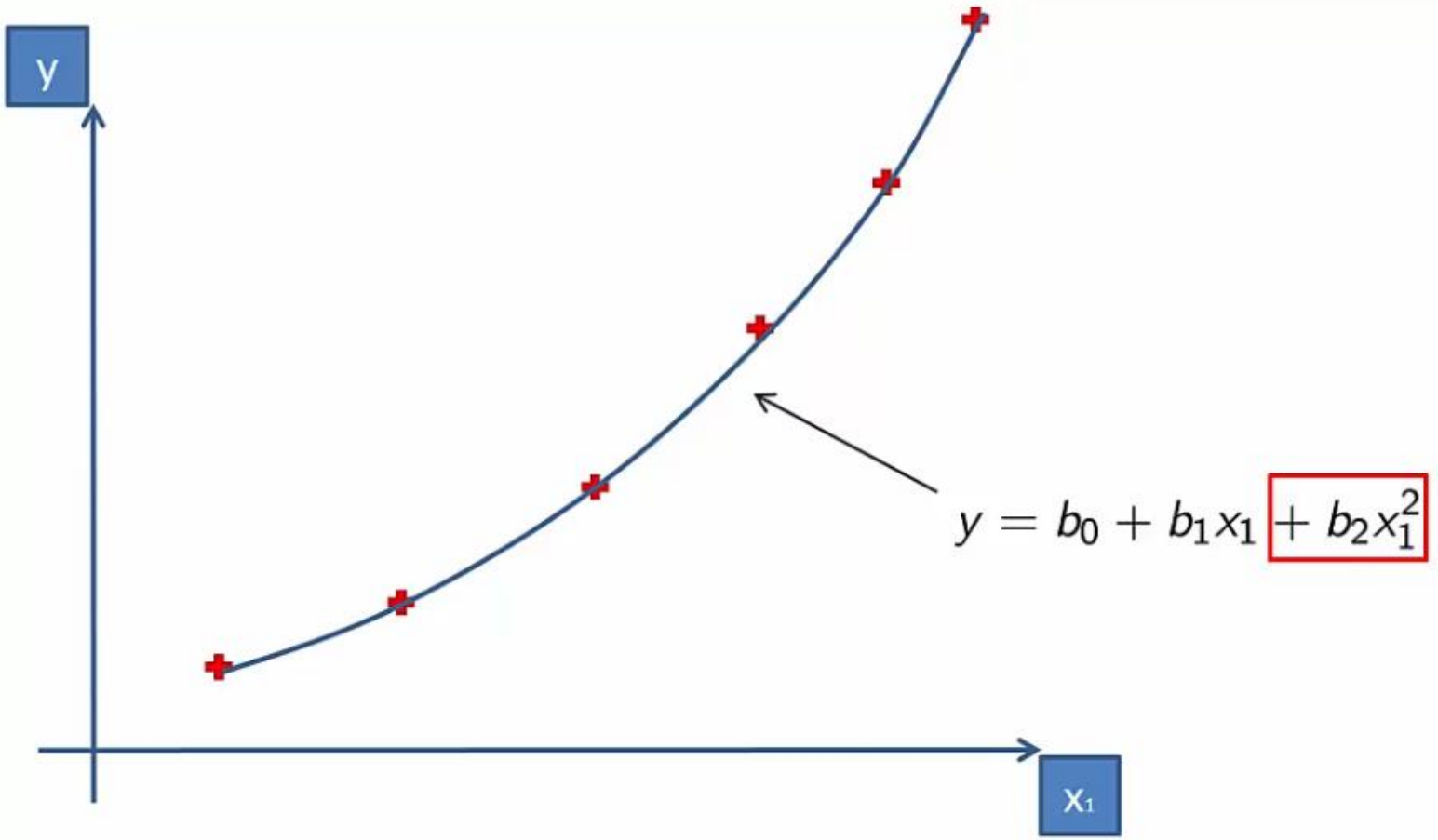
- Polynomial regression has higher degree terms
- Note that if we replace higher degree term with a new variable then it becomes exactly same as Multi Linear Regression



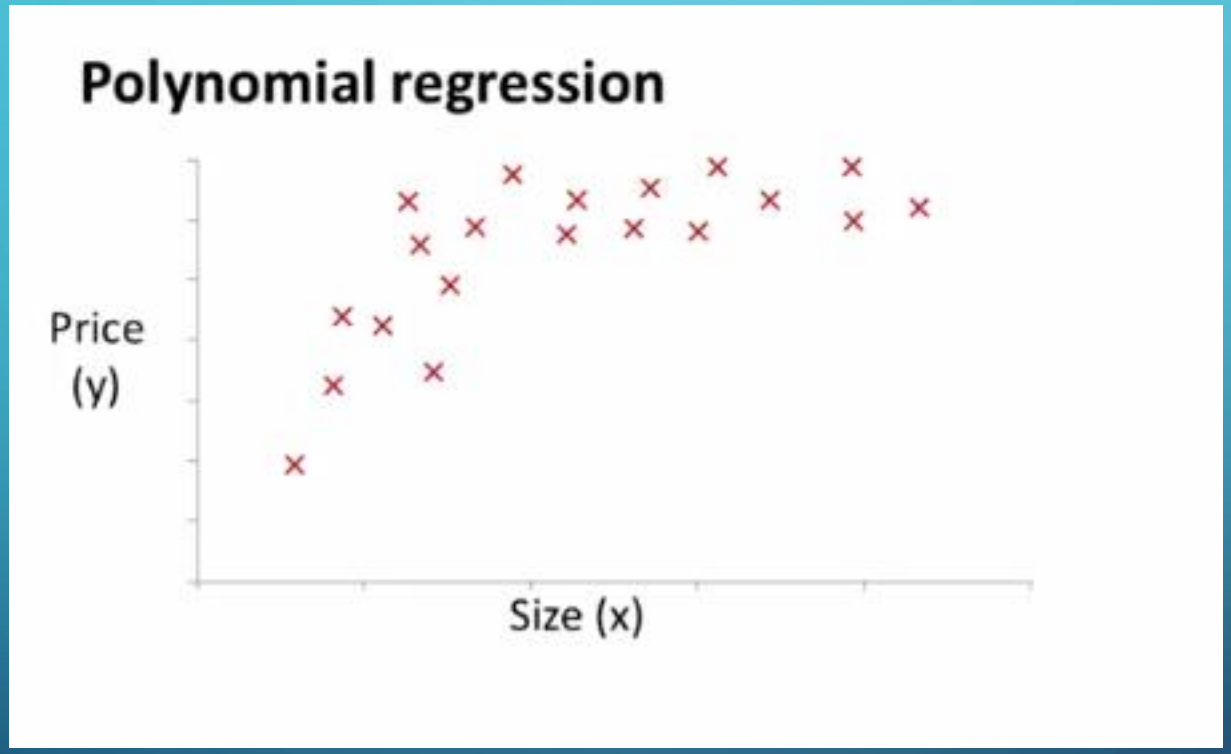
Simple Linear Regression



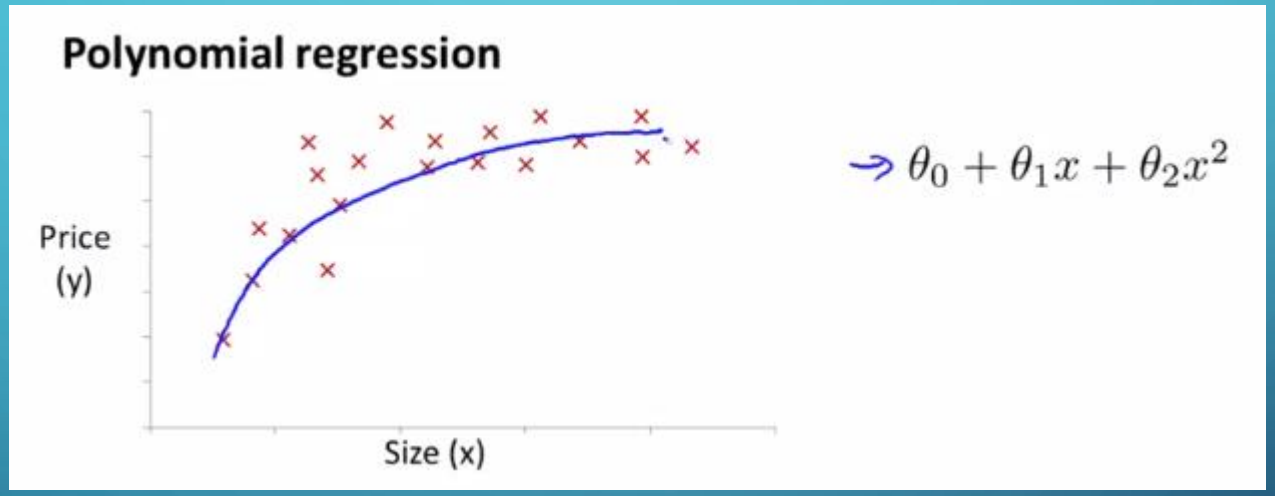
Polinomial Regression



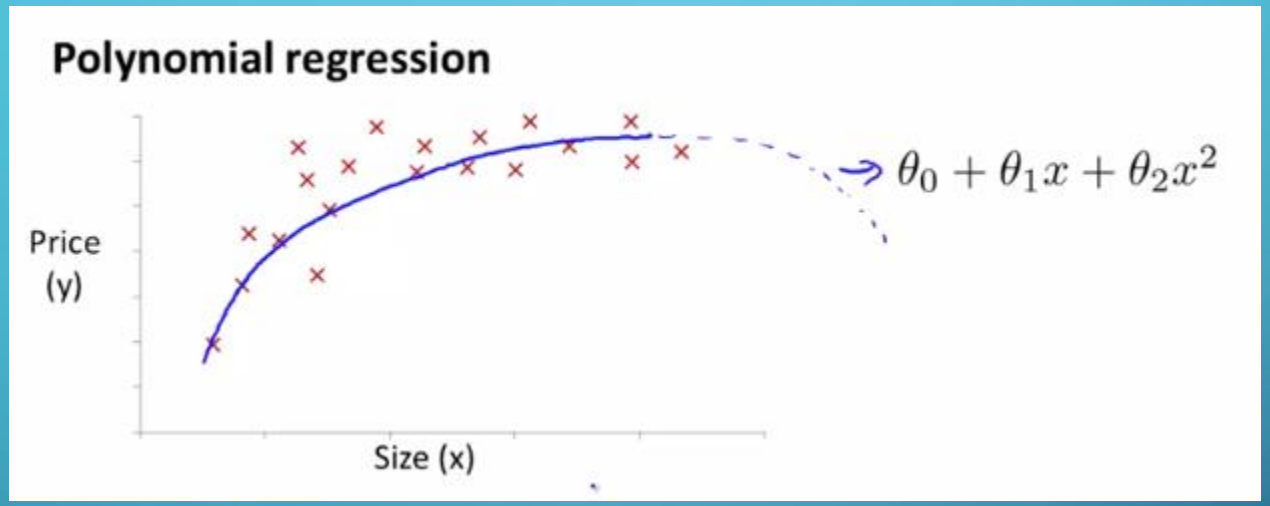
EXAMPLE:



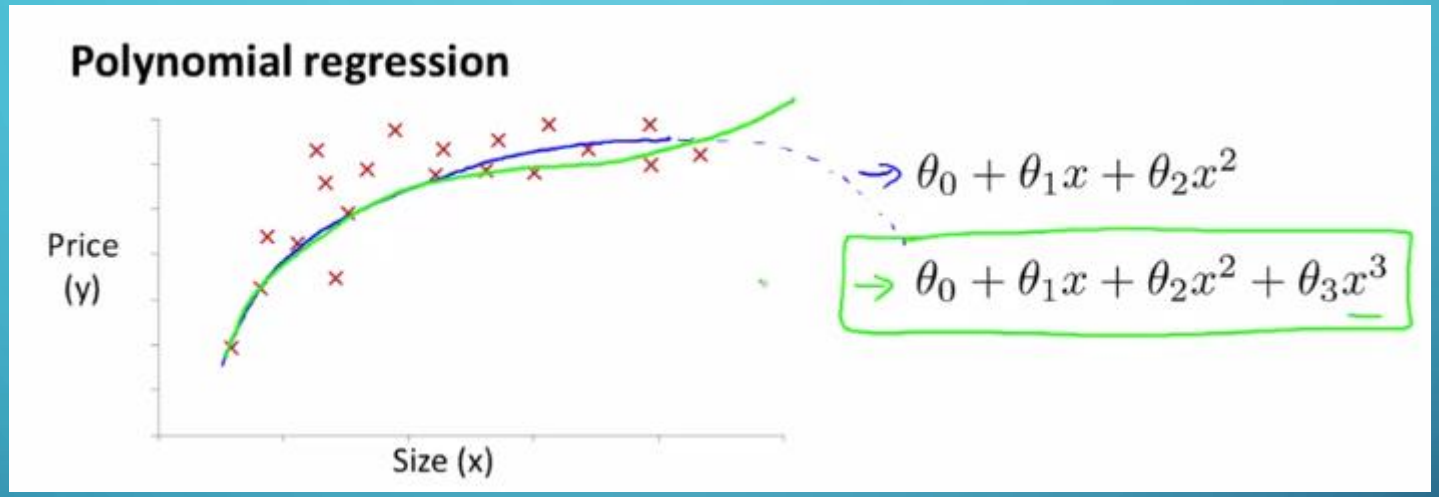
- A line wont be enough to predict
- Introduce curves using higher degree terms



- But 2nd degree wont work:



- Increase the degree further:



- Polynomial Regression is same as Multivariate Regression
- As we can replace higher degree terms with new features, this algorithm is called Polynomial **Linear** Regression. Note: Even with polynomial terms, it is called Linear

$$
\begin{aligned}
h_\theta(x) &= \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 \\
&= \theta_0 + \theta_1 (size) + \theta_2 (size)^2 + \theta_3 (size)^3
\end{aligned}
$$

Importance Feature Scaling in Polynomial Regression

- We are taking higher degree of input features and treating them as new features
- So the range of values will be different between the features
- Eg:

size →1-1000

size^2 → 1-1000000

size^3 → 1- 10^9

THANK YOU

Now lets get our hands dirty. Launch Python!!!

NEXT WEEK

- Types of Gradient Descent
- Logistic Regression
- K-Nearest Neighbours
- Naïve Bayes
- Decision Trees
- Random Forest
- Bias and Variance problem