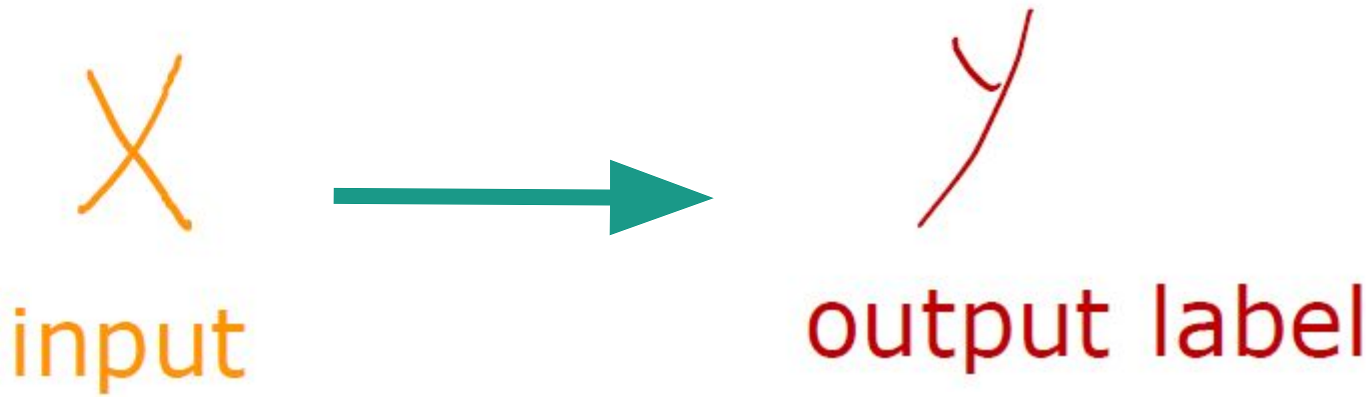
Supervised Learning
Linear regression with one variable

Ali Aburas, PhD

Slide credit:Andrew Ng

Supervised Learning

Learns from being given “Right Answers”



Supervised Learning

Input (X)	Output (Y)	Application
email →	spam? (0/1)	spam filtering
audio →	text transcripts	speech recognition
English →	Spanish	machine translation
ad, user info →	click? (0/1)	online advertising
image, radar info →	position of other cars	self-driving car
image of phone →	defect? (0/1)	visual inspection

Linear Regression (example-1)

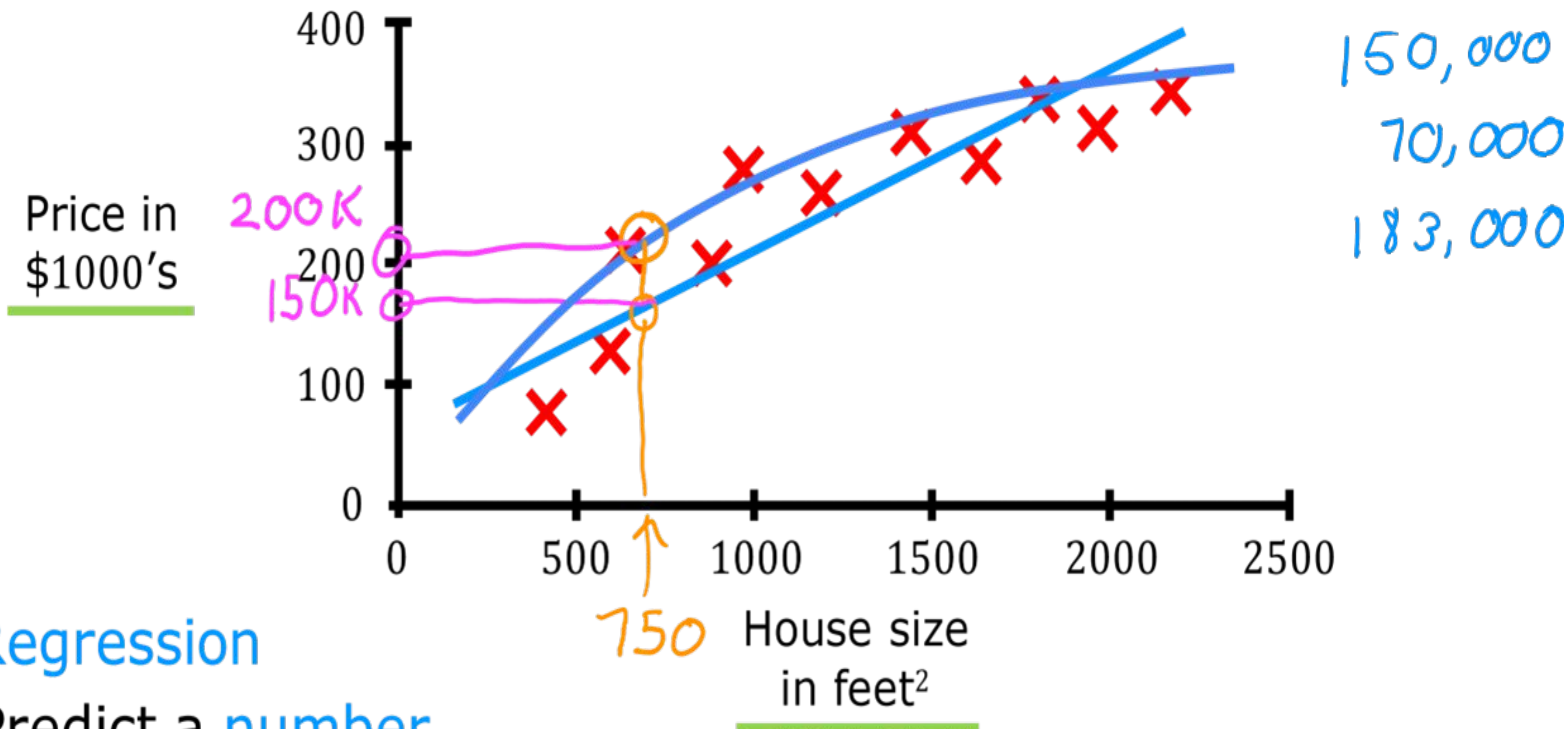
- Home prices in Tripoli Libya

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

Given these home prices find out prices whose area is,

3300
5000

Regression: Housing price prediction



Regression

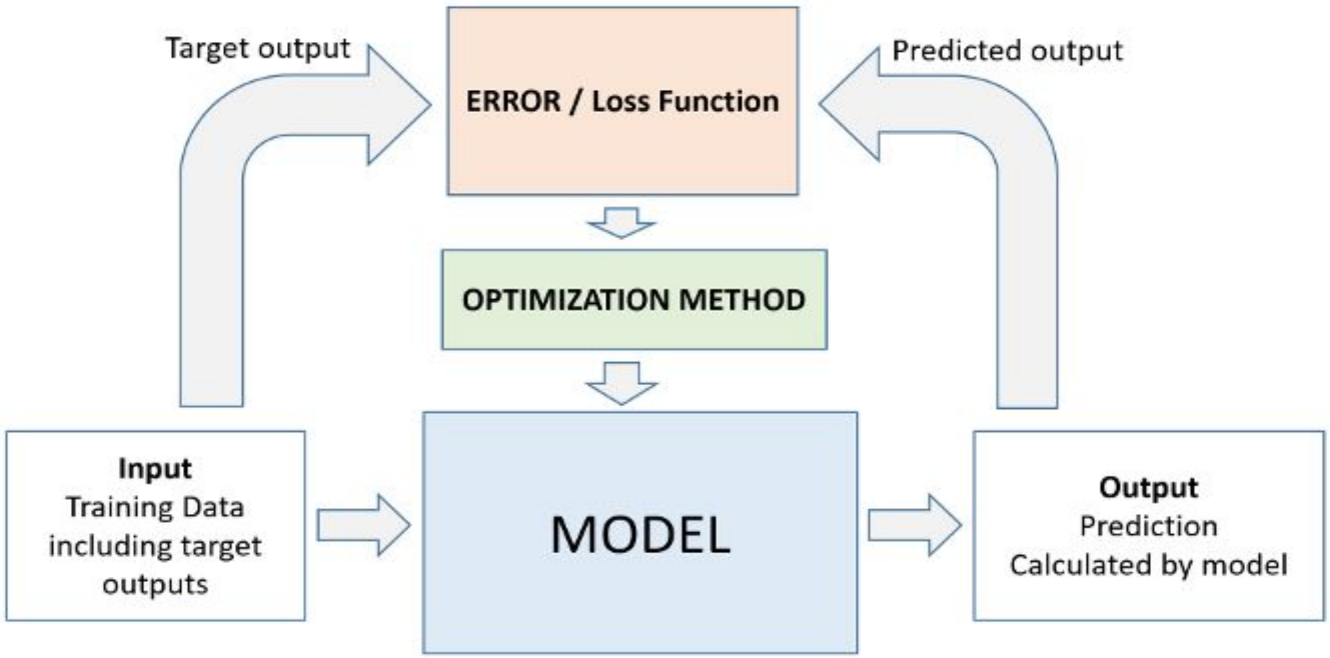
Predict a number

Machine Learning

Every machine learning problem has three components:

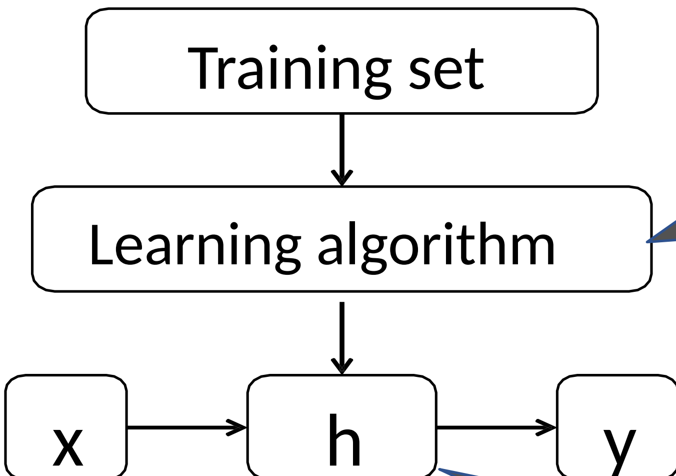
1. **Model**
2. **Cost Function**
3. **Optimizer**

We'll look at several examples of each of the above in future classes. Here's how the relationship between these three components can be visualized:



Model Representation

A **classifier/model** is a function that takes feature values as input and outputs a label

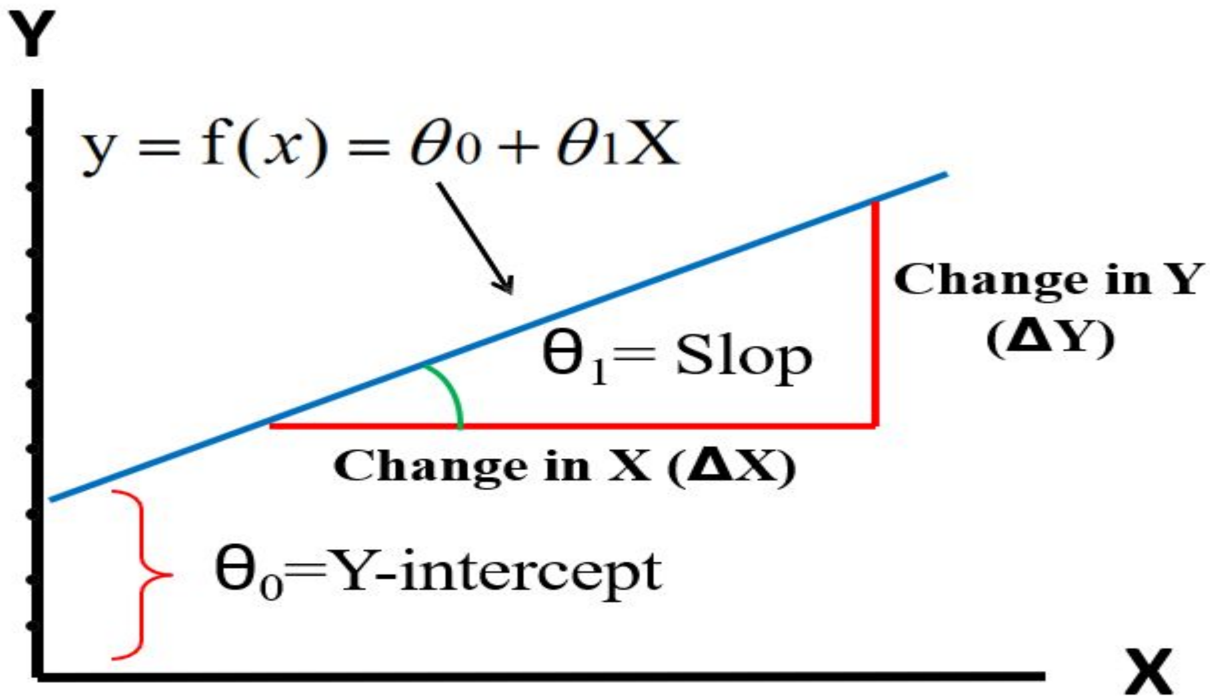


the job of a learning algorithm to output a function is usually denoted lowercase ***h*** and ***h*** stands for hypothesis

the job of a hypothesis function is taking the value of x and it tries to output the estimated value of y. So *h* is a function that maps from x's to y's

How do we represent *h* ? (model/classifier)

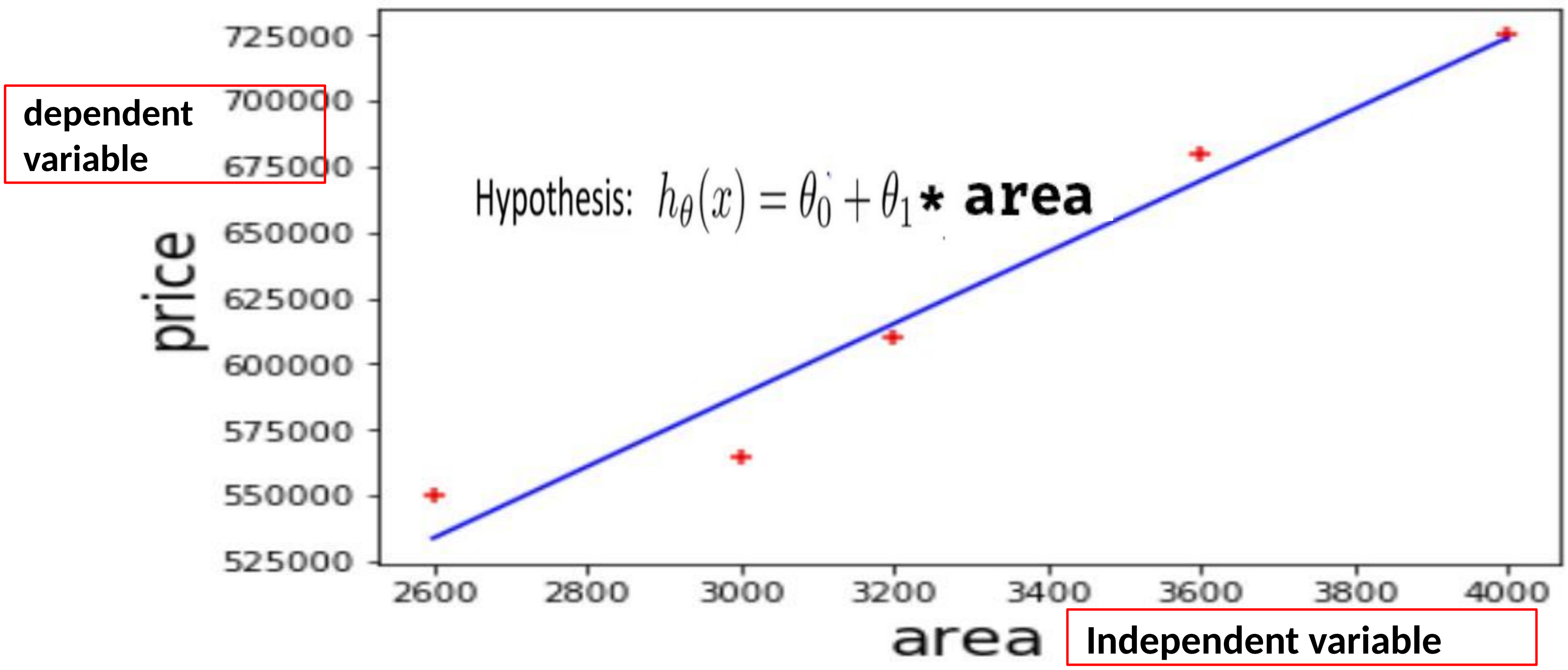
Hypothesis: $h_\theta(x) = \theta_0 + \theta_1 *$ **area**



- the **slope** basically represents the orientation of the line
- The **intercept** value is the distance between the origin point and from where the line is starting

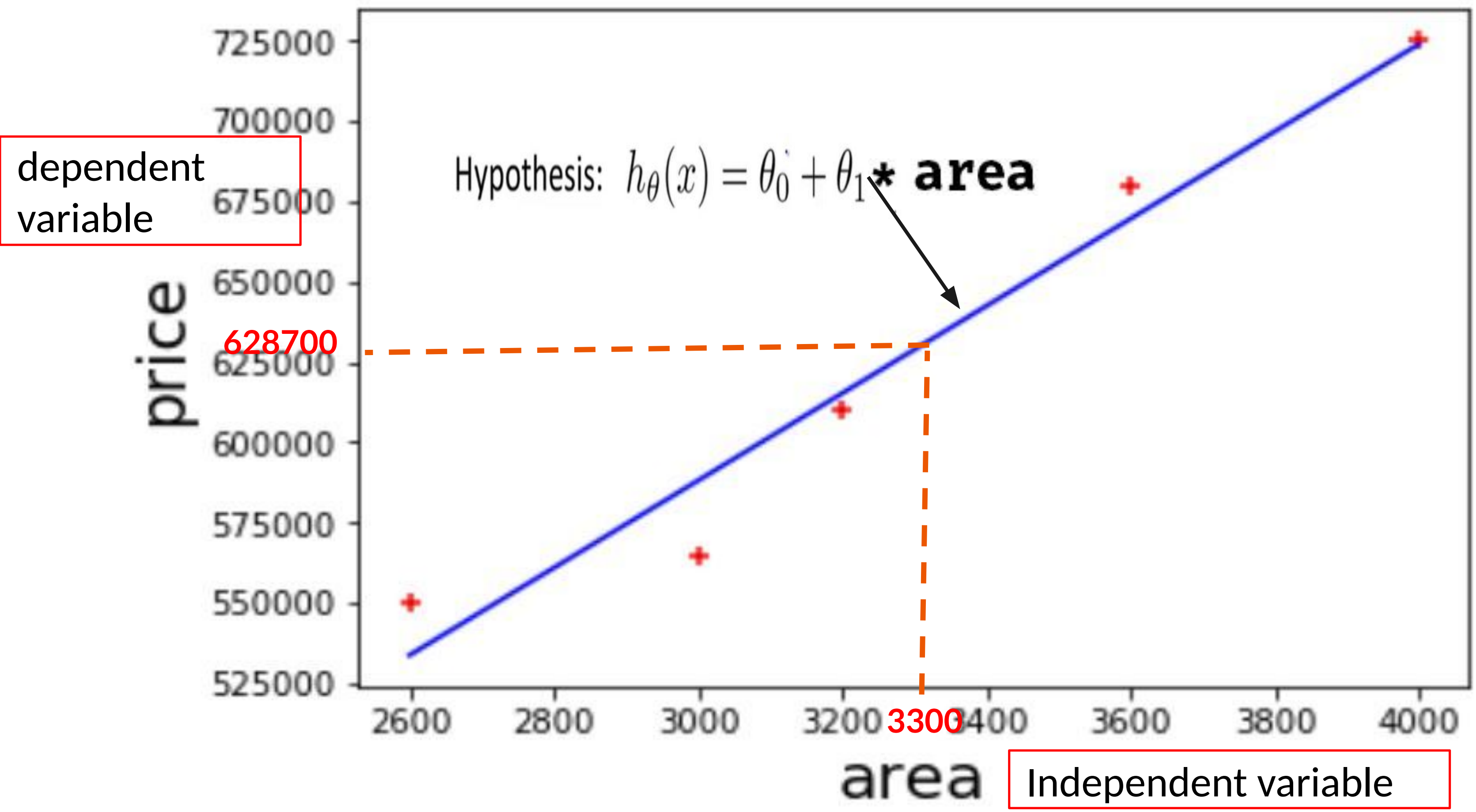
How do we represent *h* ?

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



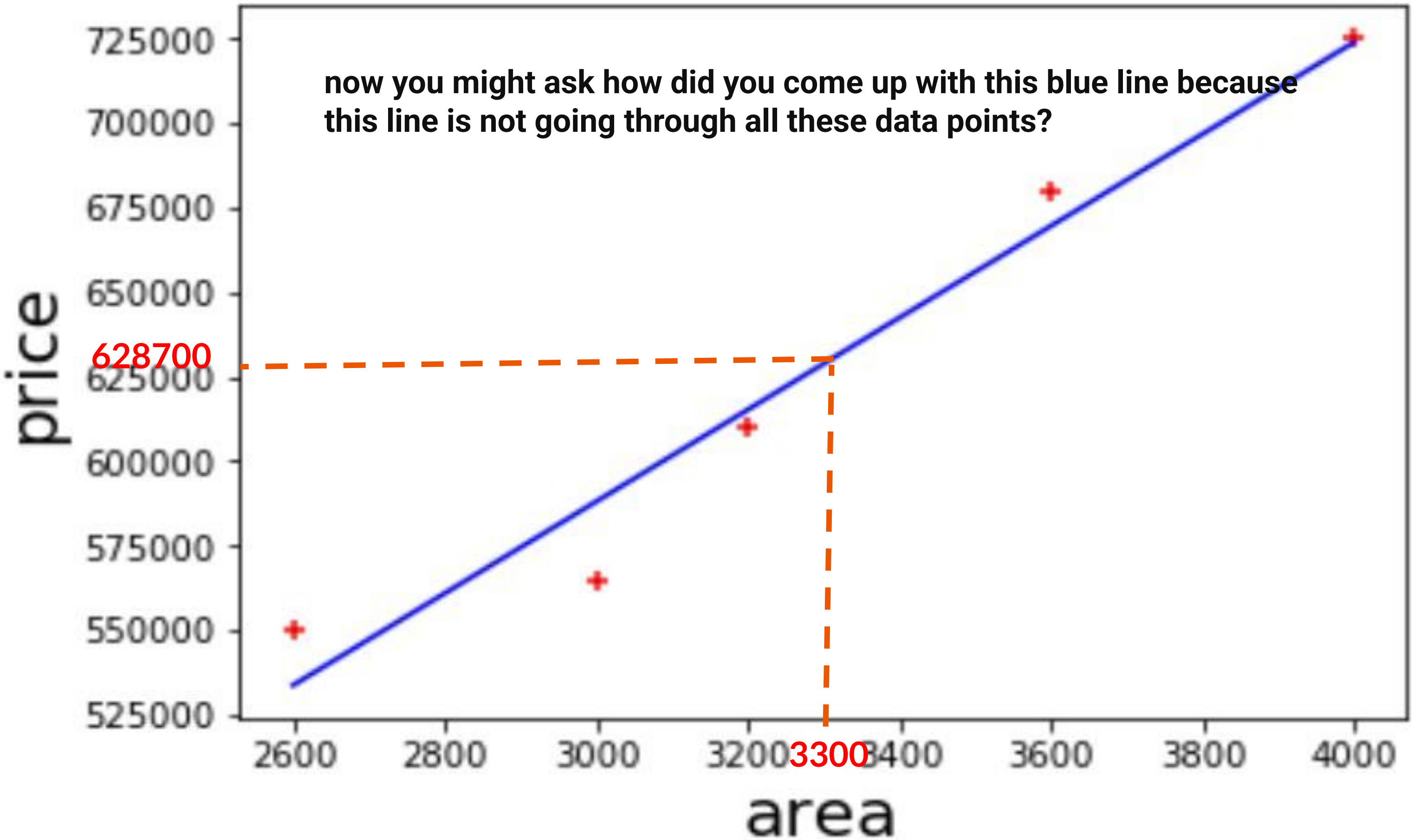
How do we represent *h* ?

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



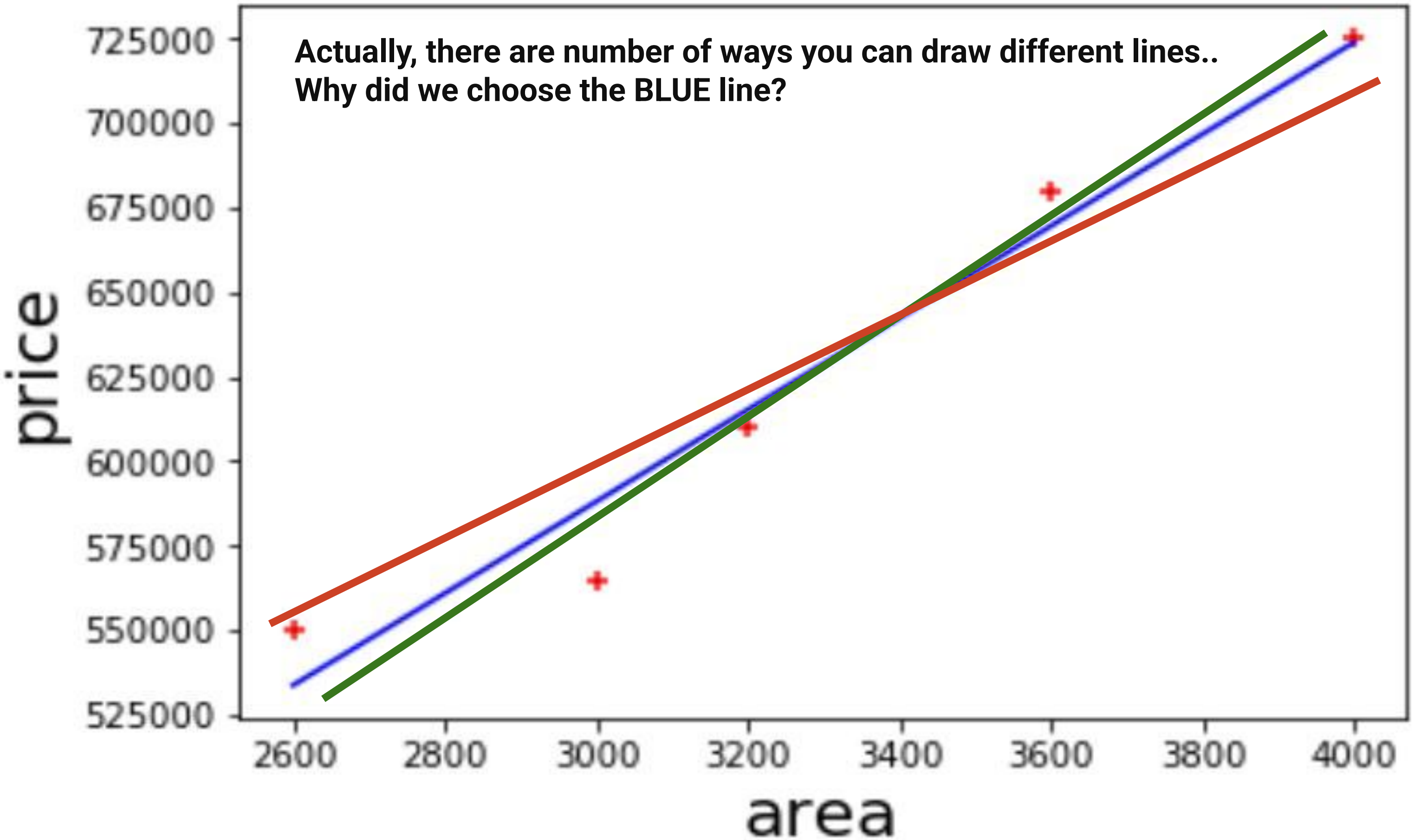
How do we represent *h* ?

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



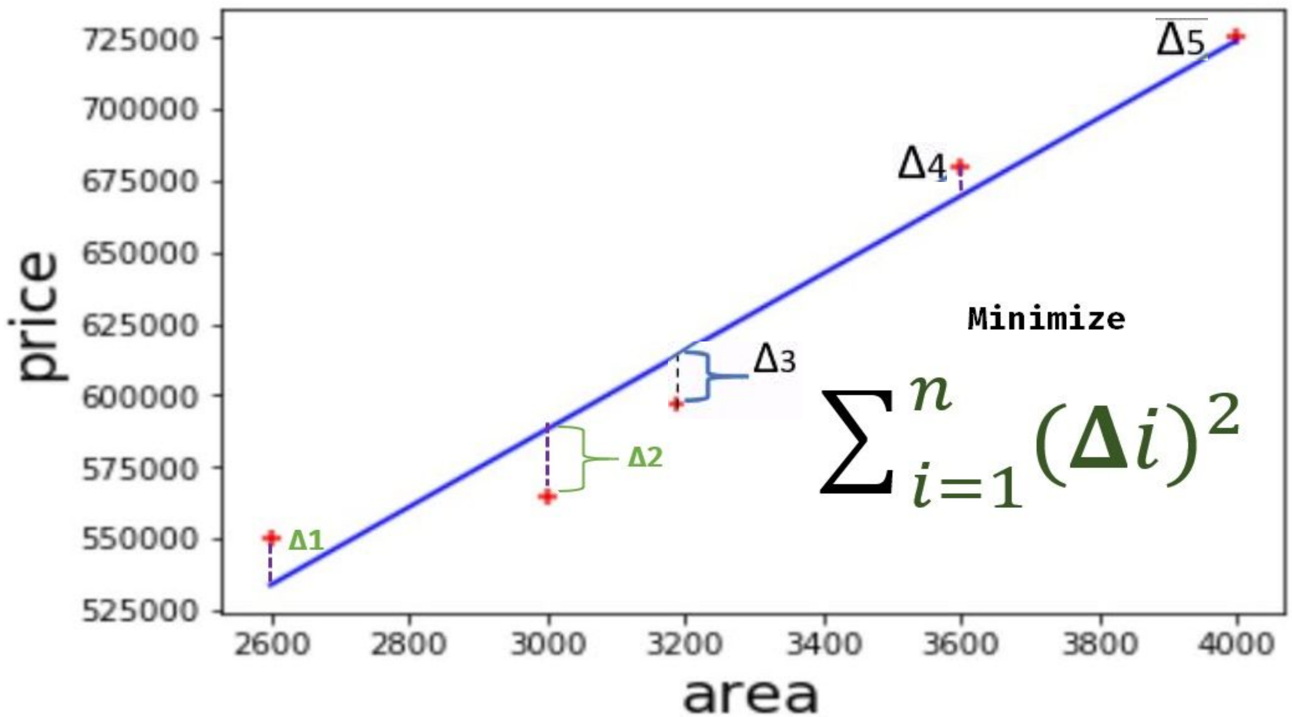
Linear Regression (example-1)

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



How do you determine which line ‘fits best’?

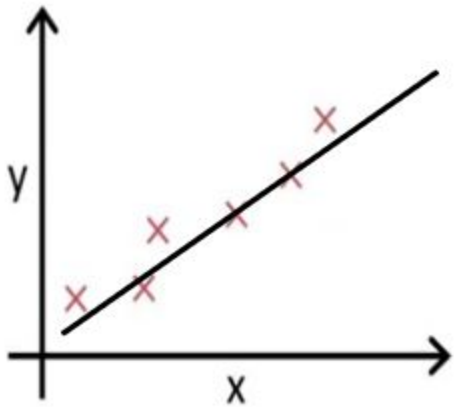
- You calculate the error between the data point and the data point predicted by your line



- The reason you want to square them is these deltas could be negative also and if you don't square them and just add them then the results might be skewed

Mean Squared Error

- Best Fit' Means Difference Between Actual Y Values and Predicted Y Values is a Minimum. So square errors!



Idea: Choose θ_0, θ_1 so that $h_\theta(x)$ is close to y for our training examples (x, y)

Cost/Loss function is helpful to determine which model performs better and which parameters θ_0, θ_1 are better

$$\underset{\theta_0 \quad \theta_1}{\text{Minimize}} \quad \frac{1}{2m}\sum_{i}^{m}(h_\theta(x^i) - y^i)^2$$

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

$h_\theta(x^i)$ predictions on the training set

y^i the actual values

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

$$\underset{\theta_0 \quad \theta_1}{\text{Minimize}} \quad j(\theta_0, \theta_1)$$

m= number of datapoints

Cost function visualization

X	y
1	1
2	2
3	3

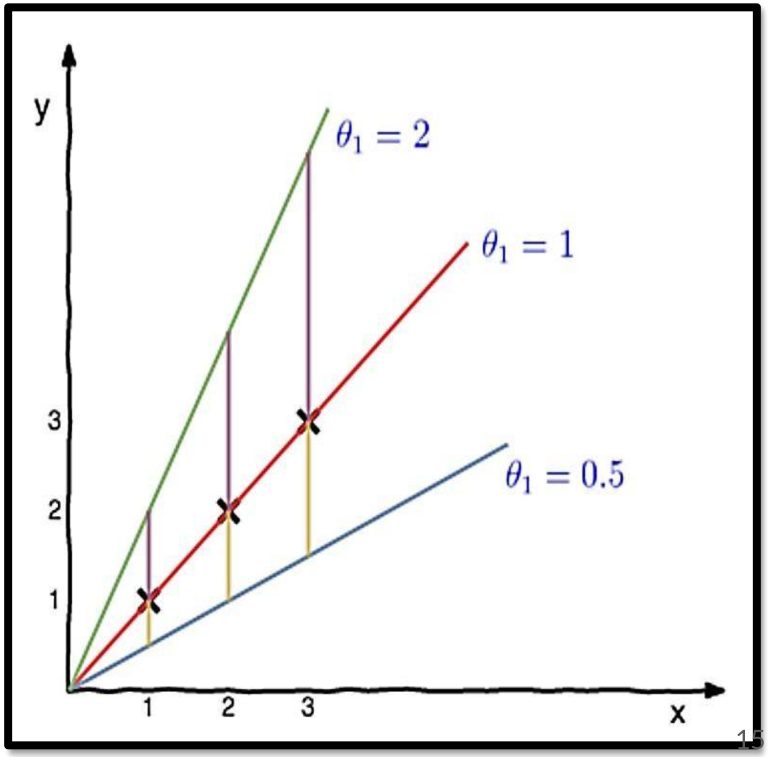
- Consider a simple case of **hypothesis** by setting θ_0=0, then ***h*** becomes : $h_\theta(x)=\theta_1 x$
  Each value of θ_1 corresponds to a different hypothesis as it is the **slope** of the line which corresponds to different lines passing through the origin as shown in plots below as y-intercept i.e. θ_0 is nulled out.

$$J(\theta_1) = \frac{1}{2m}\sum_{i=1}^{m}\left(\theta_1 x^{(i)} - y^{(i)}\right)^2$$

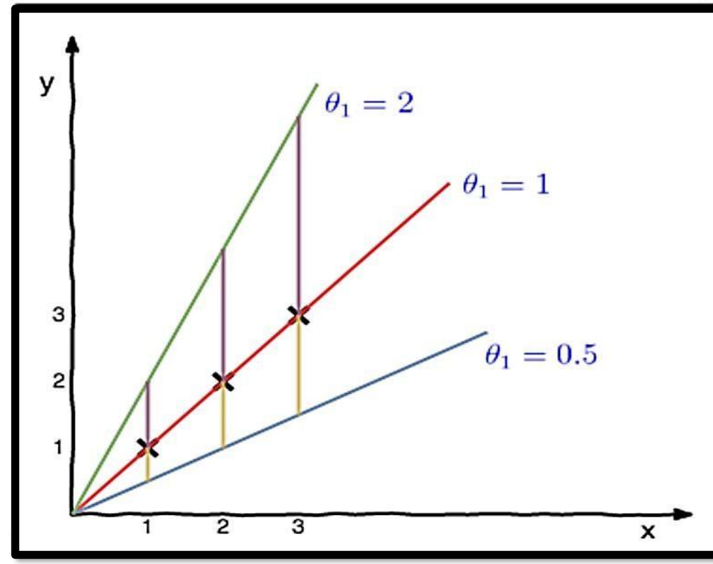
At θ_1=2, $J(2) = \frac{1}{2*3}(1^2+2^2+3^2) = \frac{14}{6} = 2.33$

At θ_1=1, $J(1) = \frac{1}{2*3}(0^2+0^2+0^2) = 0$

At θ_1=0.5, $J(0.5) = \frac{1}{2*3}(0.5^2+1^2+1.5^2) = 0.58$



$$J(\theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(\theta_1 x^{(i)} - y^{(i)} \right)^2$$

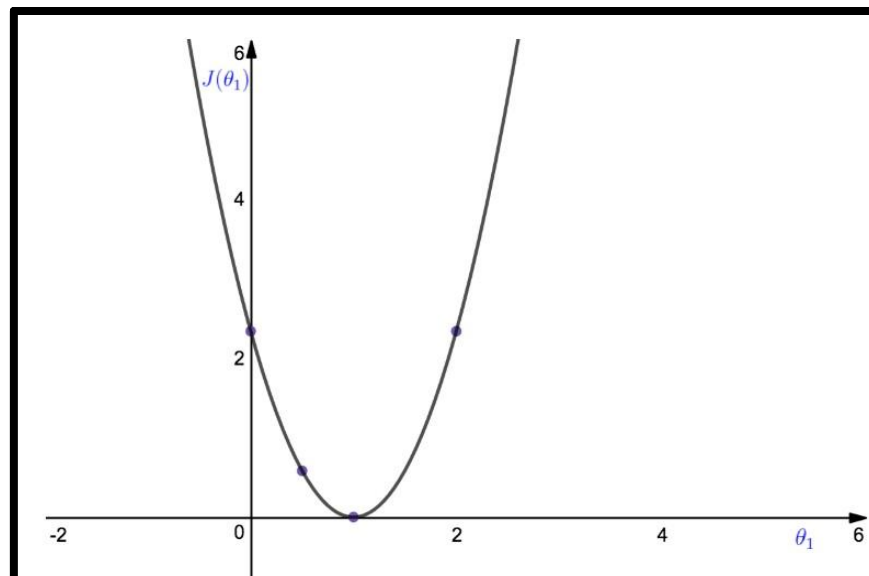


At θ_1=2, $J(2) = \frac{1}{2*3}(1^2 + 2^2 + 3^2) = \frac{14}{6} = 2.33$

At θ_1=1, $J(1) = \frac{1}{2*3}(0^2 + 0^2 + 0^2) = 0$

At θ_1=0.5, $J(0 = \frac{1}{2*3}(0.5^2 + 1^2 + 1.5^2) = 0.58$

On **plotting points** like this further, one gets the following graph for the cost function which is dependent on parameter $\boldsymbol{\theta_1}$.

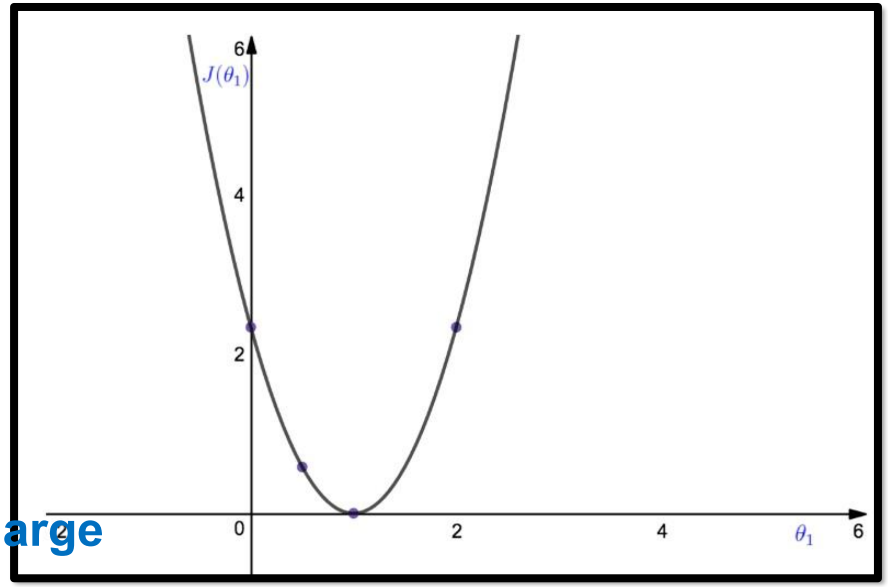


plot each value of $\boldsymbol{\theta_1}$ corresponds to a different hypothesizes

Cost function visualization

What is the optimal value of θ_1 that minimizes $J(\theta_1)$?

It is clear that best value for $\theta_1 = 1$ as $J(\theta_1) = 0$, which is the minimum.



How to find the best value for θ_1 ?

Plotting ?? Not practical especially in high dimensions?

The solution :

1. Analytical solution: not applicable for large datasets
2. Numerical solution: ex: Gradient descent .

Linear Regression Equation

Hypothesis:

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

Parameters:

$$\theta_0, \theta_1$$

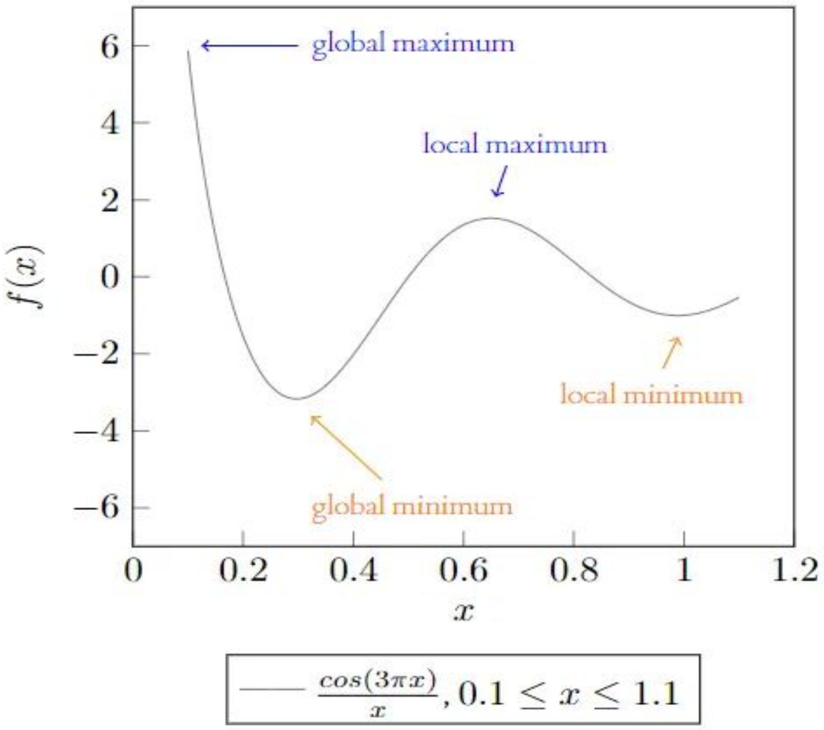
Cost Function:

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

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

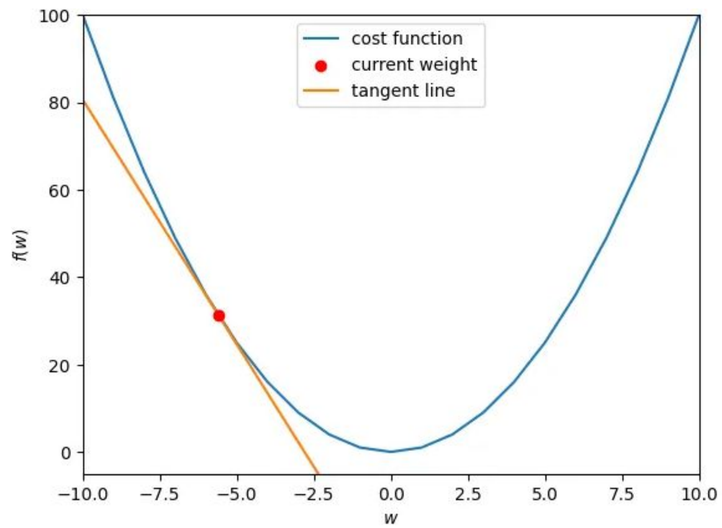
Optimization Algorithms

- In machine learning, **optimization** is the process of finding the ideal parameters, or weights, to maximize or minimize a cost or loss function.
- The **global maximum** is the largest value on the domain of the function, whereas the **global minimum** is the smallest value.
- While there is only one global maximum and/or minimum, there can be many **local maxima** and **minima**.
- The **global minimum or maximum** of a cost function indicates where a model's parameters generate predictions that are close to the actual targets.
- The **local maxima and minima** can cause problems when training a model, so their presence should always be considered.



Gradient Descent in One Dimension

- **Gradient descent** is a first-order, iterative optimization algorithm used to minimize a cost function.
- By using partial derivatives, a direction, and a learning rate, gradient descent decreases the error, or difference, between the predicted and actual values.
- The idea behind gradient descent is that the derivative of each weight will reveal its direction and influence on the cost function.
- In the image below, the cost function is $\mathbf{f(w) = w^2}$.
- The minimum is at (0,0), and the current weight is **w**=-5.6. The current loss is **f(w)**=31.36, and the line in orange represents the derivative which is -11.2 (2w)
- This indicates the weight needs to move "downhill" — or become more positive — to reach a loss of 0. This is where gradient descent comes in.



Gradient Descent

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

Want $\min\limits_{\theta_0, \theta_1} J(\theta_0, \theta_1)$

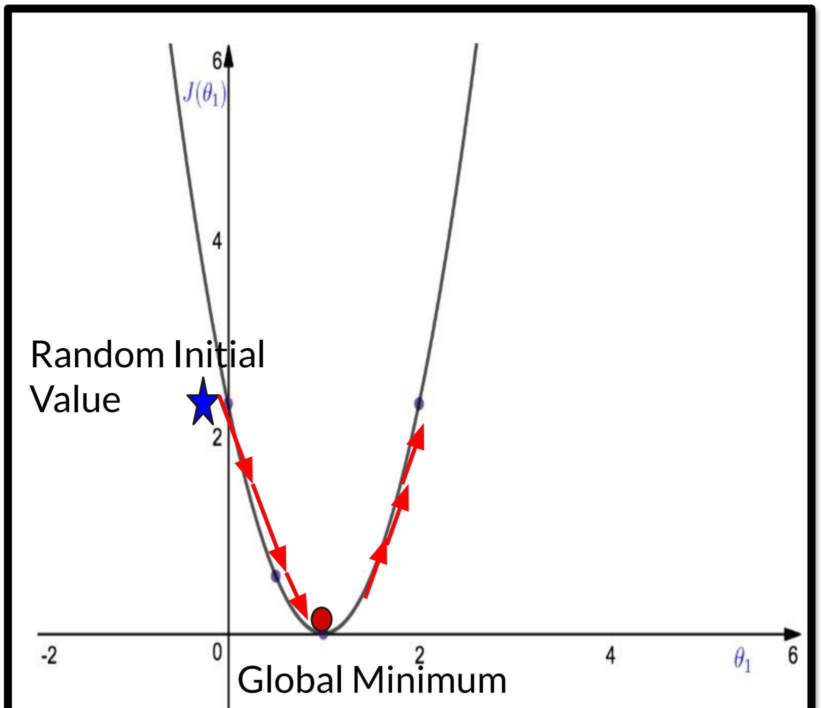
Outline:

- 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

Gradient Descent Equation

- **Gradient descent** is an optimization algorithm used for minimizing the loss function in various machine learning algorithms so it is not only used in linear regression. It is used for updating the parameters of the learning model.



$$\frac{d}{d\theta_j} j(\theta_0,\theta_1) = \frac{d}{d\theta_j}\frac{1}{2m}\sum_{i=1}^{m}\left(h\theta(x_i)-Y_i\right)^2$$

$$\frac{d}{d\theta_j} j(\theta_0,\theta_1) = \frac{d}{d\theta_j}\frac{1}{2m}\sum_{i=1}^{m}\left(\theta_0+\theta_1(x_i)-Y_i\right)^2$$

$$j=0: \frac{d}{d\theta_0} j(\theta_0,\theta_1) = \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x_i)-Y_i\right)$$

$$j=1: \frac{d}{d\theta_1} j(\theta_0,\theta_1) = \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x_i)-Y_i\right)\bullet x_i$$

Gradient descent Algorithm

$$y = f(x) = \theta_0 + \theta_1 X$$

$$\text{repeat until convergence}\{\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) \forall j \in \{0,1\}\}$$

- Where
  - $:=$ is the assignment operator
  - α is the **learning rate** which basically defines how big the steps are during the descent
  - $\frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$ is the **partial derivative** term
  - j = 0, 1 represents the **feature index number**

Also the parameters should be **updated simulatenously**, i.e. ,

$$temp_0 := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$$

$$temp_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$$

$$\theta_0 := temp_0$$

$$\theta_1 := temp_1$$

Gradient descent algorithm

repeat until convergence {

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

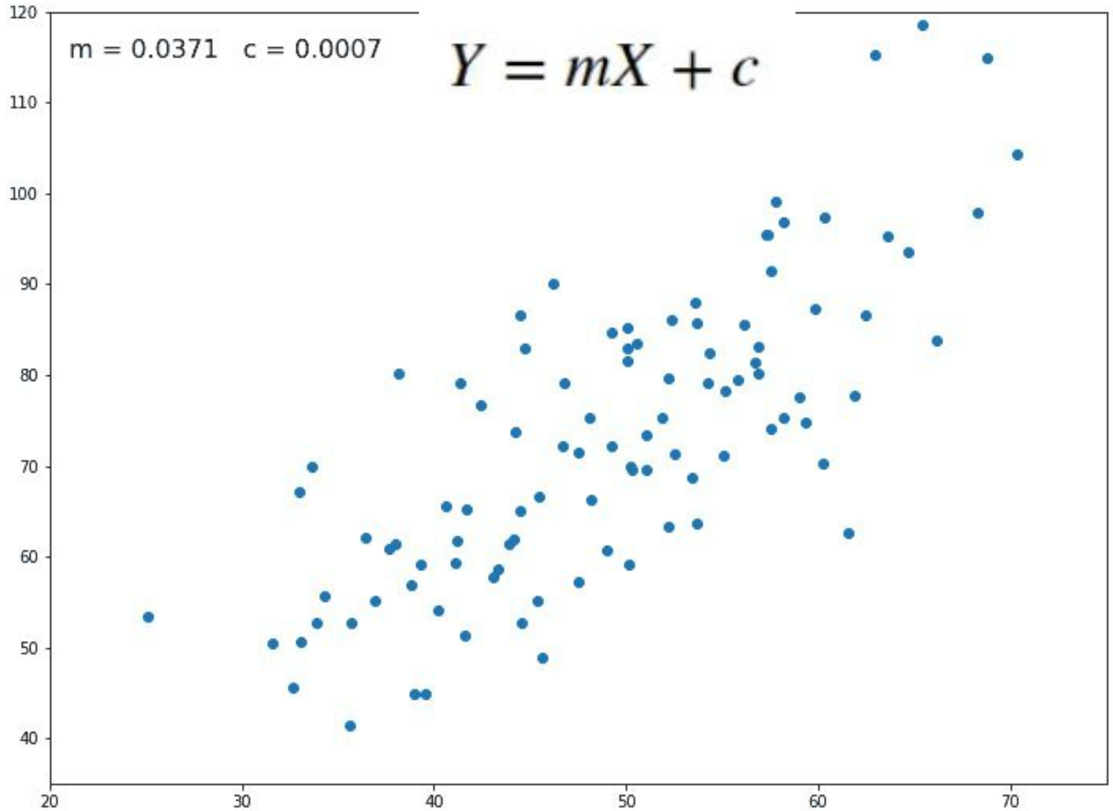
$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right) \cdot x^{(i)}$$

}

Gradient Descent

$$y = f(x) = \theta_0 + \theta_1 X$$

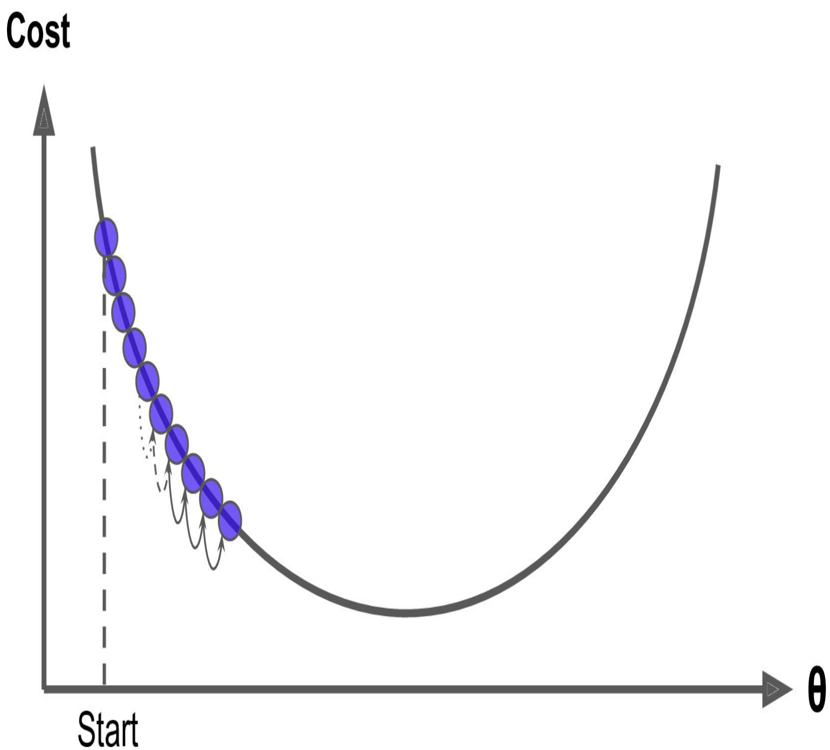
$$Y = mX + c$$



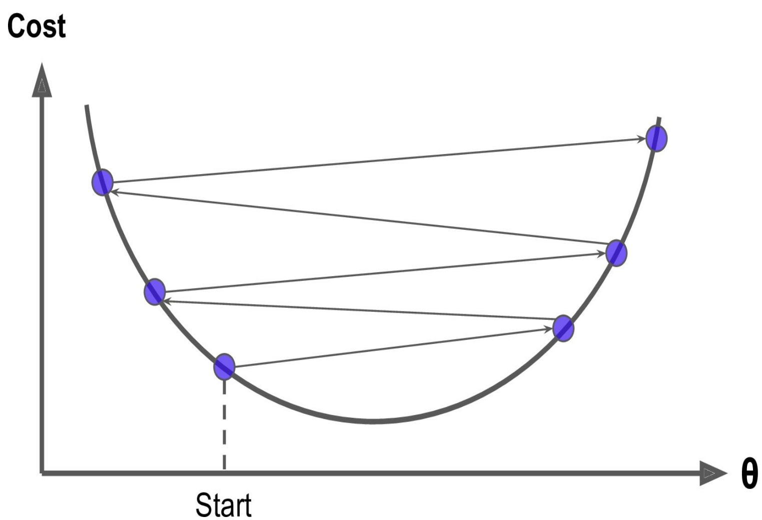
The values of c as theta0 and **m as theta1 are updated at each iteration to get the optimal solution**

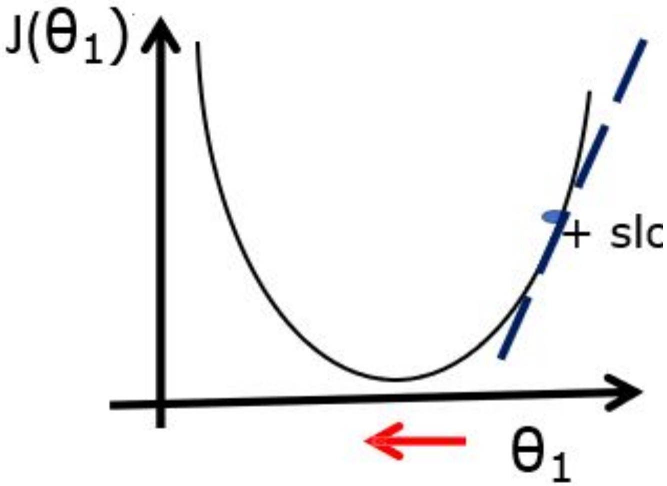
An important parameter in Gradient Descent

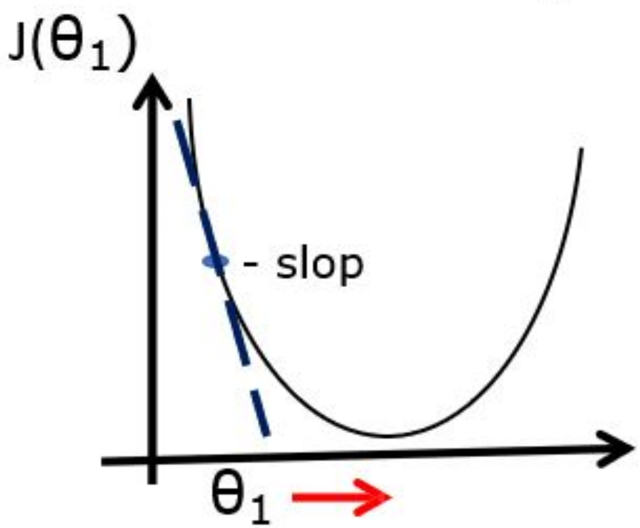
If the **learning rate** is too small, then the algorithm will have to go through many iterations to converge, which will take a long time.



if the learning rate is too high, you might jump across the valley and end up on the other side, possibly even higher up than you were before. This might make the algorithm diverge, with larger and larger values, failing to find a good solution







$$\boxed{\theta_1 = \theta_1 - \alpha \frac{d}{d\theta_1} j(\theta_1)}$$

$\theta_1 = \theta_1 - \alpha(+ve)$

$\theta_1 = \theta_1 - \alpha(-ve)$

Example:

Question : Find the local minima of the function $y=(x+5)^2$ starting from the point $x=3$

Step 1 : Initialize x =3. Then, find the gradient of the function, dy/dx = 2*(x+5).

Step 2 : Move in the direction of the negative of the gradient. But wait, how much to move? For that, we require a learning rate. Let us assume the **learning rate → 0.01**

Step 3 : Let's perform 2 iterations of gradient descent

Initialize Parameters :

$$X_0 = 3$$

$$Learning\ rate = 0.01$$

$$\frac{dy}{dx} = \frac{d}{dx}(x+5)^2 = 2 * (x+5)$$

Iteration 1 :

$$X_1 = X_0 - (learning\ rate) * (\frac{dy}{dx})$$

$$X_1 = 3 - (0.01) * \left(2 * (3+5)\right) = 2.84$$

Iteration 2 :

$$X_2 = X_1 - (learning\ rate) * (\frac{dy}{dx})$$

$$X_2 = 2.84 - (0.01) * \left(2 * (2.84+5)\right) = 2.6832$$