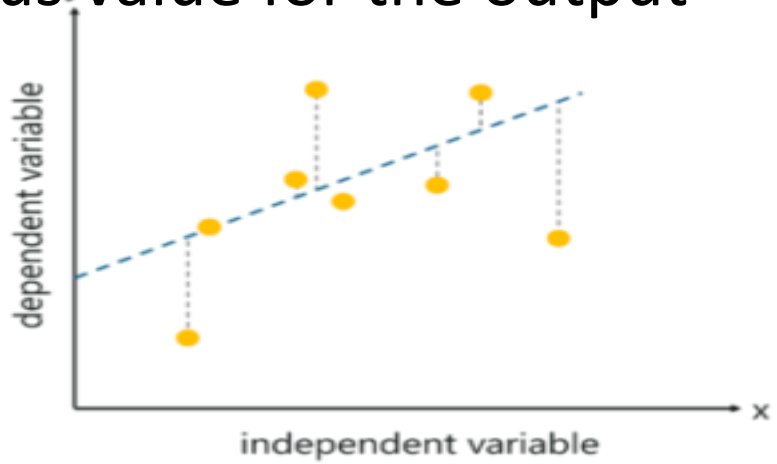
Machine Learning

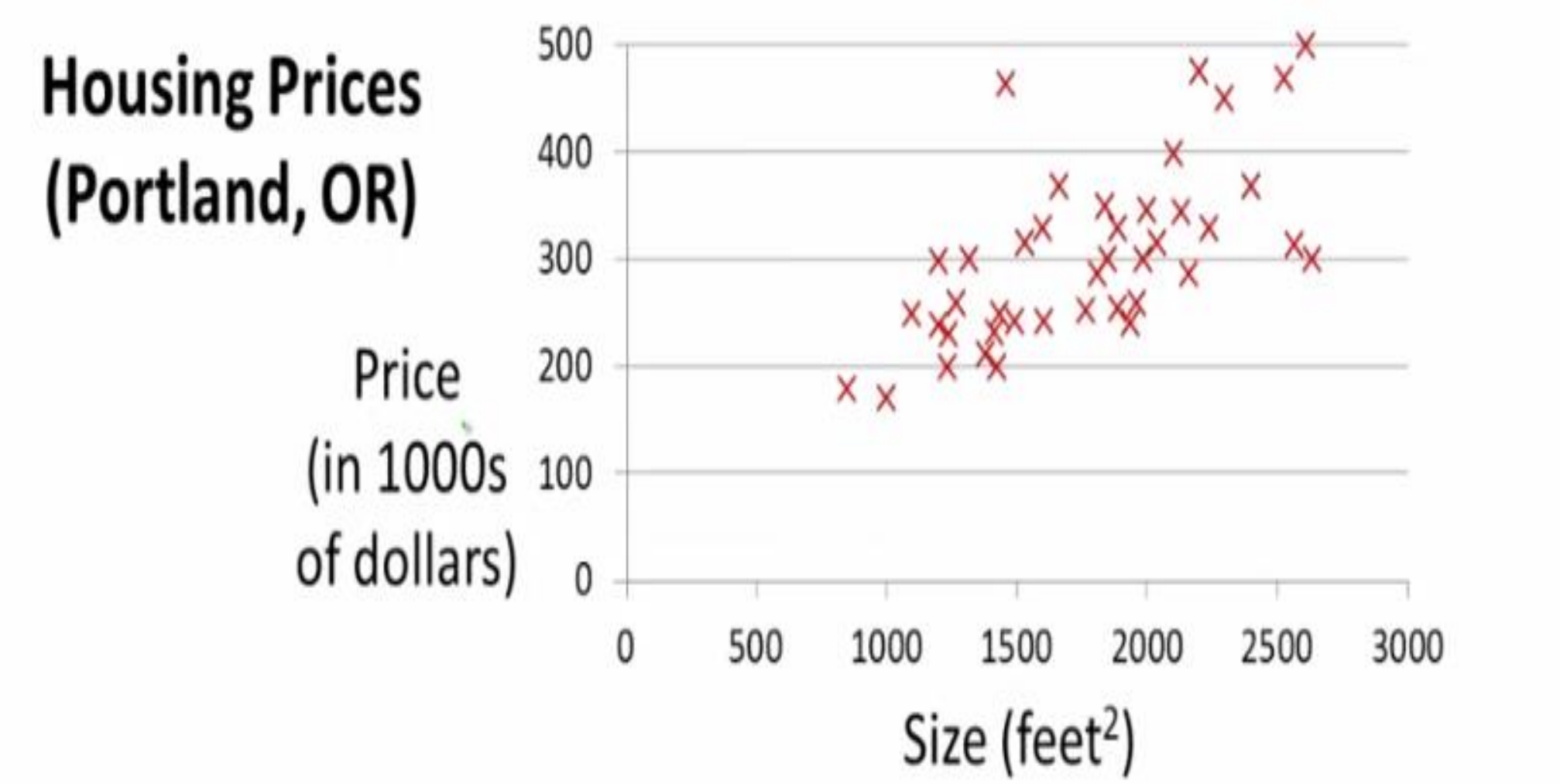
Linear Regression

Linear Regression

- This algorithm assumes that there is a linear relationship between the 2 variables, Input (X) and Output (Y), of the data it has learnt from. The **Input variable** is called the *Independent Variable* and the **Output variable** is called the *Dependent Variable*. When unseen data is passed to the algorithm, it uses the function, calculates and maps the input to a continuous value for the output



Linear Regression



Linear Regression

Training set of housing prices (Portland, OR)

Size in feet^2 (x)	Price ($) in 1000's (y)
2104	460
1416	232
1534	315
852	178
…	…

Notation:

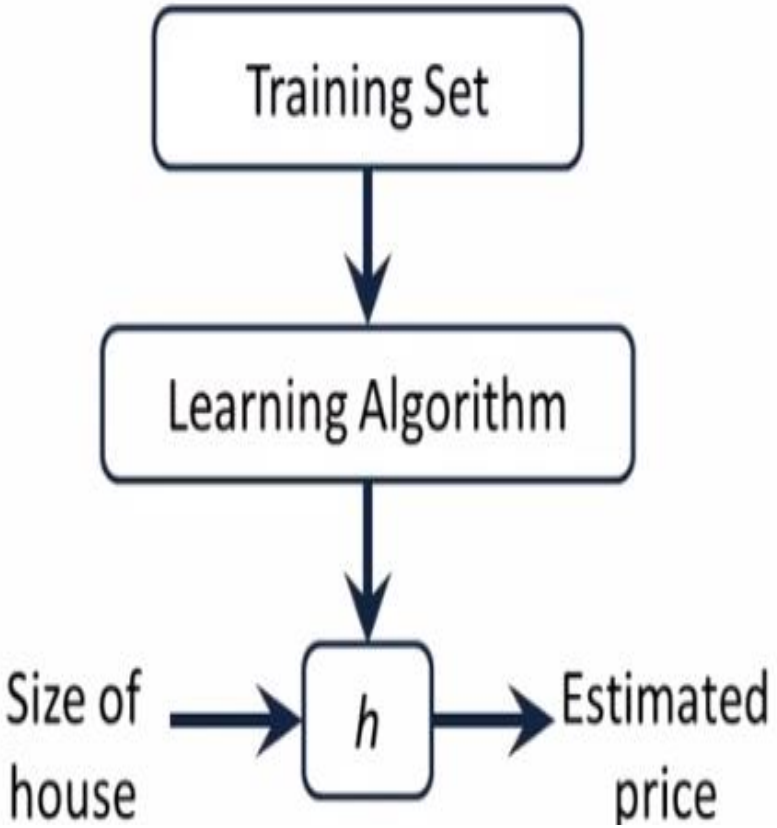
m = Number of training examples

x's = "input" variable / features

y's = "output" variable / "target" variable

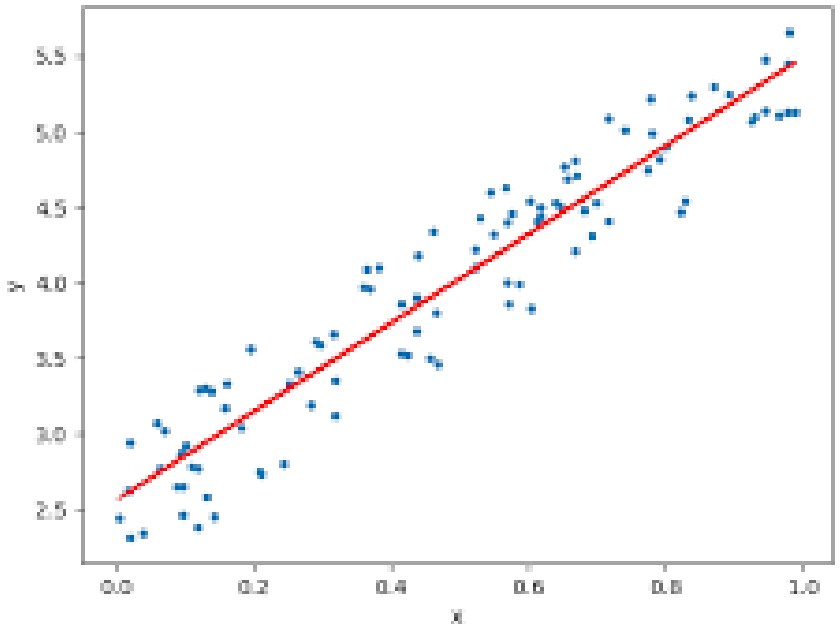
Linear Regression

- a data set of housing prices
- plot data set of a number of houses
- that were of different sizes and
- sold for a range of different prices.
- Eg a house is size of 1250 square
  how much it may be sold for ?
- fit a model ie fit a straight line to this data.
- let's say maybe he can sell the
- house for around $220,000.



Fit a Best Line

- Line Equation : y=c+mx
  - Where m is slope
  - And c is constant (intercept)



- Here in LR model
  - Y is hypothesis
  - X is feature
  - m and c are parameters (Theta0 and Theta 1)

Linear Regression with one variable

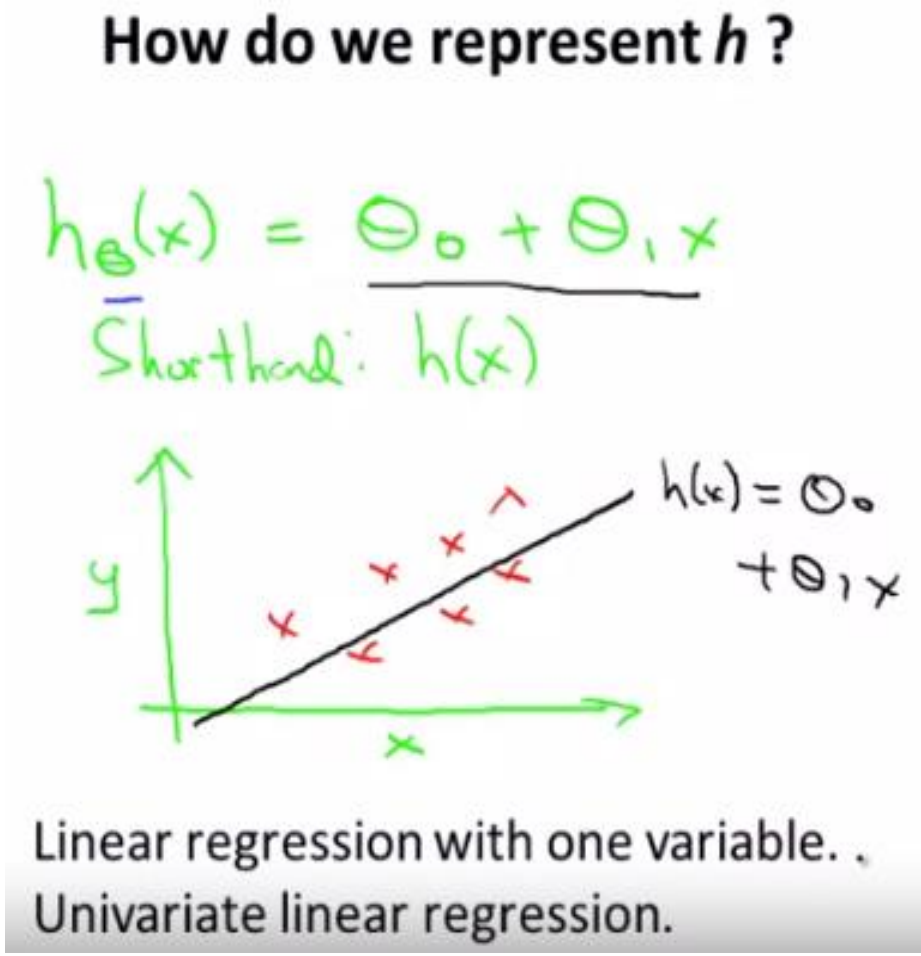
- x^(1) refers to 2104
- Y^(1) refers to 460
- X^(2) refers to 1416 ….

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

- job of a learning algorithm is now to output a
- function which by convention is usually denoted by lowercase h
- h stands for hypothesis
- So h is a function that maps from x's to y's.

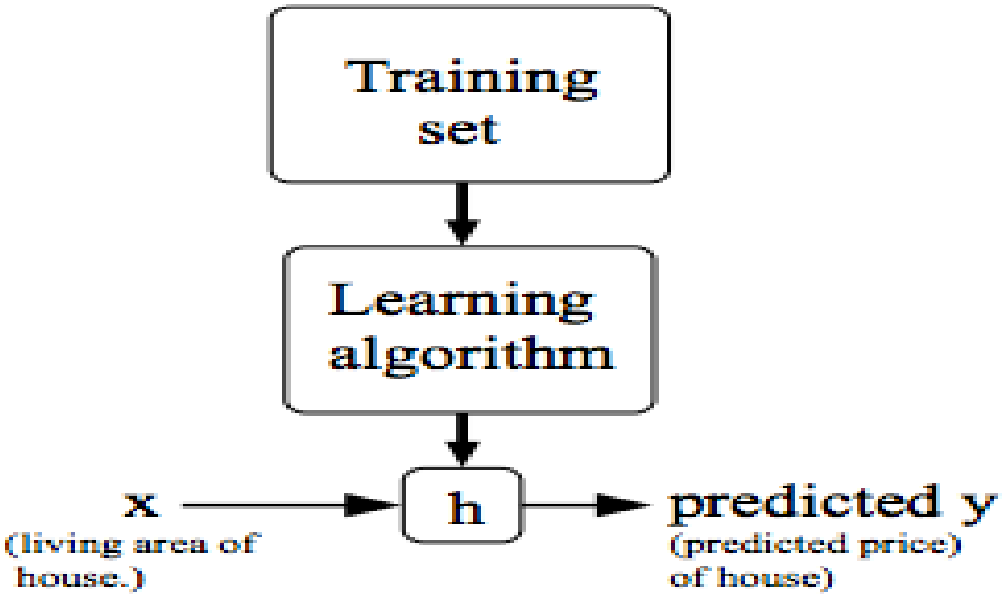
Linear Regression with one variable

- Also called Univariate LR
- the term regression refers to the
- fact that we are predicting a real-valued output
- namely the price.
- to learn from this data how to predict prices
- of the houses
- X denotes the input variable
- Y denotes output variable
- (x,y) denotes a single training example
- (x^(i),y^(i)) is the ith training example



Model Representation

- our goal is, given a training set, to learn a function h : X → Y so that h(x) is a “good” predictor for the corresponding value of y. For historical reasons, this function h is called a hypothesis.



Representing hypothesis h

Function h is predicting that y (which is the price of the house)is some straight line function of x

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

Cost Function

- Cost function determines how to fit the best possible straight line to our data.
- Theta 1 and Theta 0 are the parameters of the model.
- With different choices of the parameter's theta 0 and theta 1, we get different hypothesis

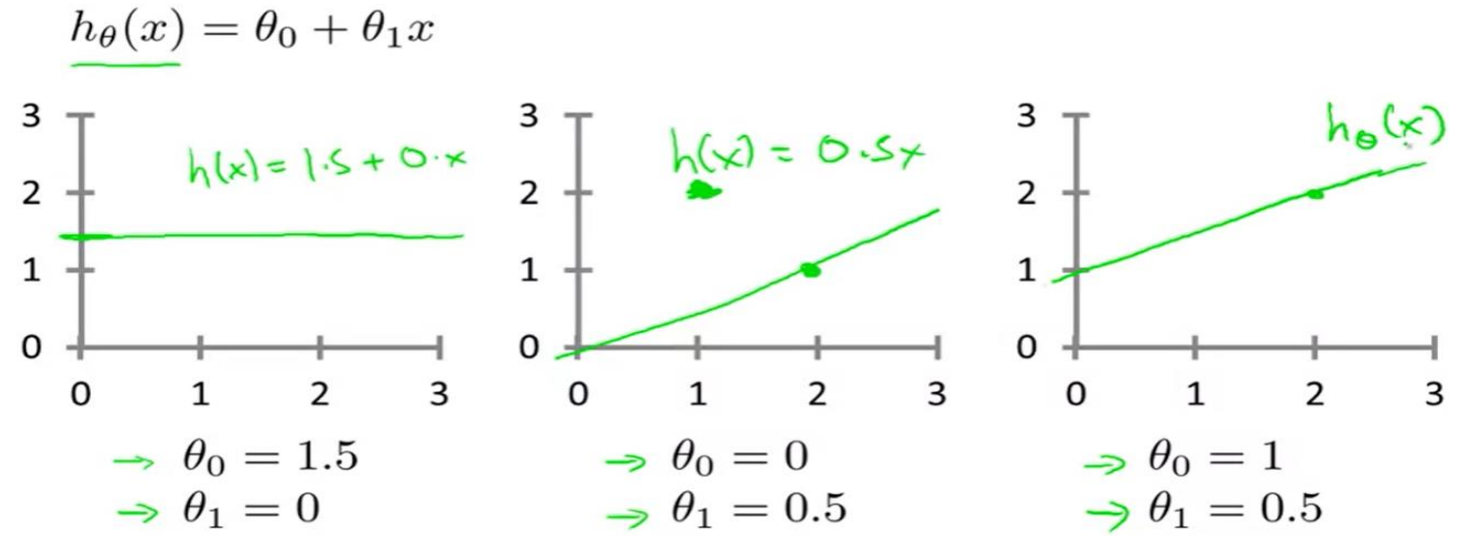
Hypothesis

- Examples : the hypothesis function will look like this with different values. As h(x) is prediction for y (price), therefore

Case 1: for all values of x, h (x) remains constant, ie, 1.5, nothing depends on x, price becomes fixed.

Case 2: if x=1, h(x) or y=.5, if x=2, h(x) or y=1

Case 3: if x=1, h(x) or y = 1.5, if x=2, h(x) or y=2



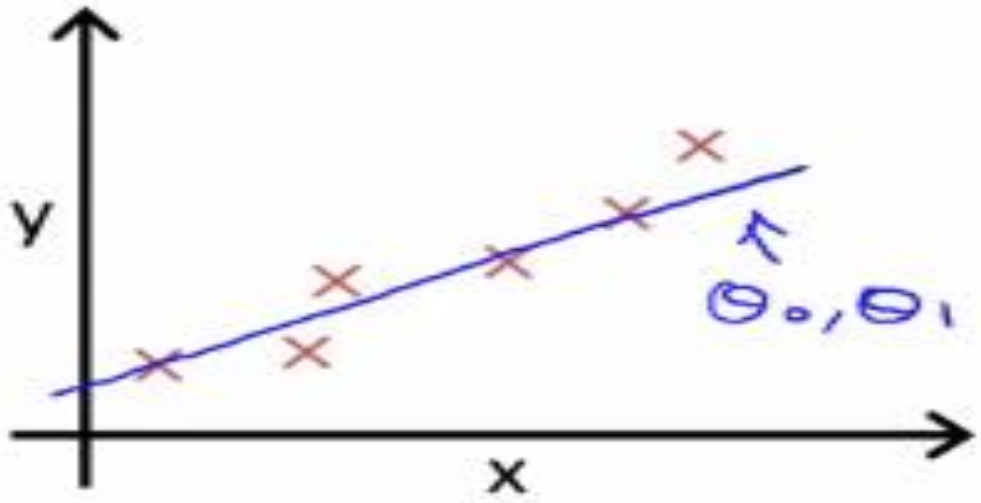
Cost Function

- Choose values for the parameters so that, at least in the training set, given the X in the training set we make good predictions for the Y values.

- We have to minimize the cost function.

- The difference between h(x) and y should be small.

Model Representation



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

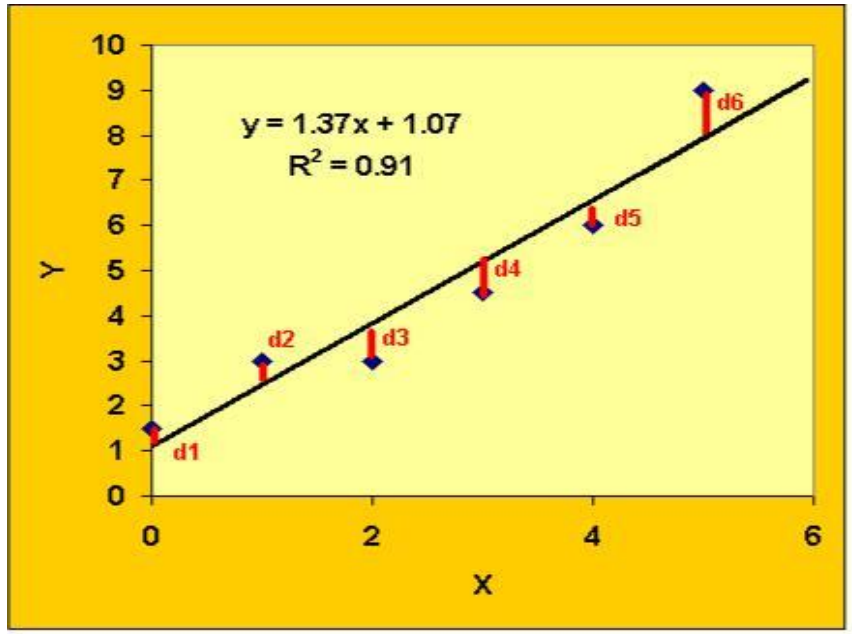
Cost Function

- We can measure the accuracy of our hypothesis function by using a **cost function**. This takes a difference of all the results of the hypothesis with inputs from x's and the actual output y's.

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

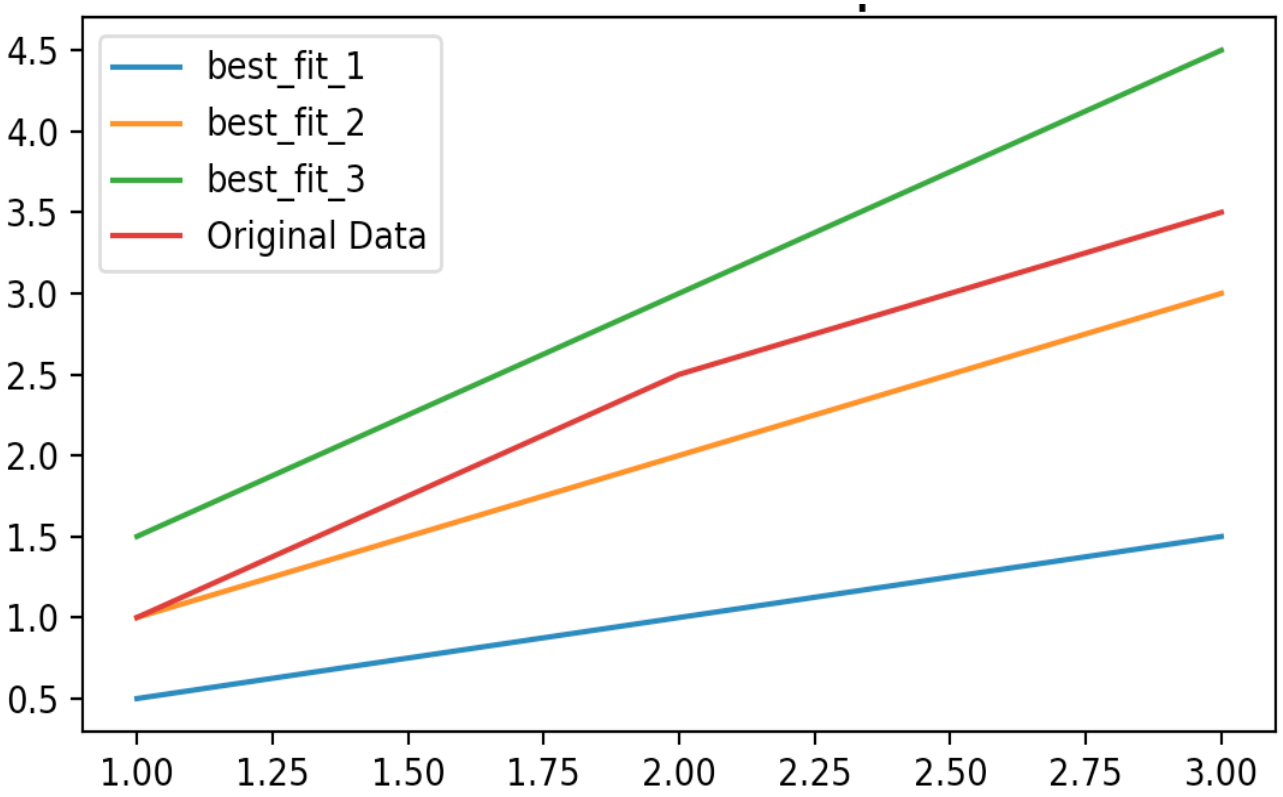
Cost Function

- This function is otherwise called the "Squared error function", or "Mean squared error".
- The mean is halved as a convenience for the computation of the gradient descent, as the derivative term of the square function will cancel out the ½ term.



x	y	h(x) = x-1	h(x)-y	(h(x)-y) squared
4.5	3.15	3.5	3.5-3.15	0.12
6.38	4.46	5.38	5.38-4.46	0.84
8	5.60	7	7-5.60	1.96
10	7	9	9-7	4
12.78	8.94 (value to be predicted))	11.78	11.78-8.94	8.06
				Sum = 14.98

Cost Functions



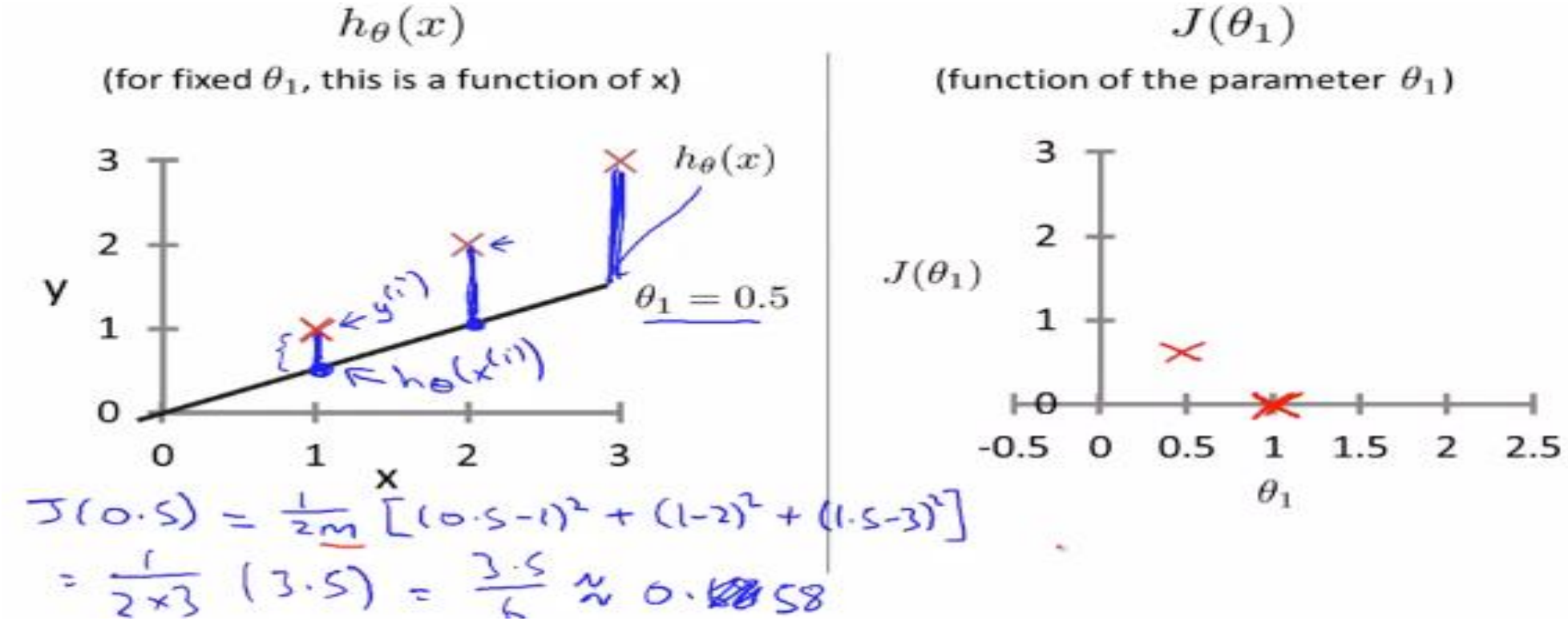
Hypothesis: $h_\theta(x) = \theta_0 + \theta_1 x$

Parameters: θ_0, θ_1

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

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

Example



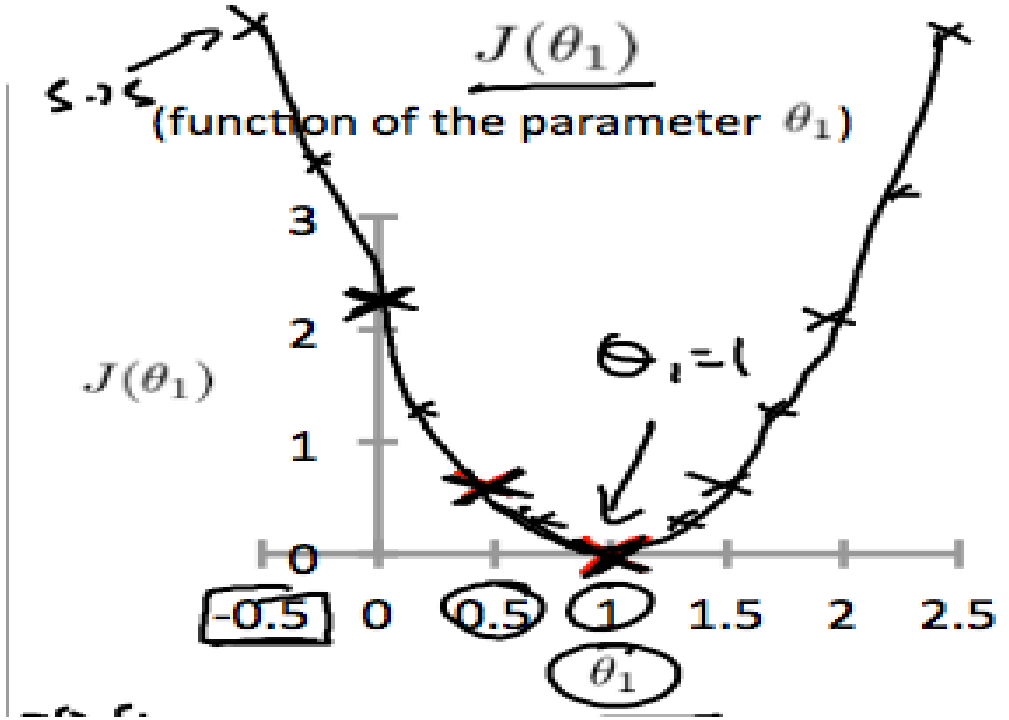
Red crosses are actual y prices, x axis has original house sizes

When x=1, Theta 1=0.5, h(y) derived is 0.5
x=2, Theta 1=0.5, h(y) derived is 1

When Theta=0.5, j (cost function) derived as 0.58
When Theta1=1, j(costs function) dertived as 0, so means error
So the LR line passes perfectly

Cost Function

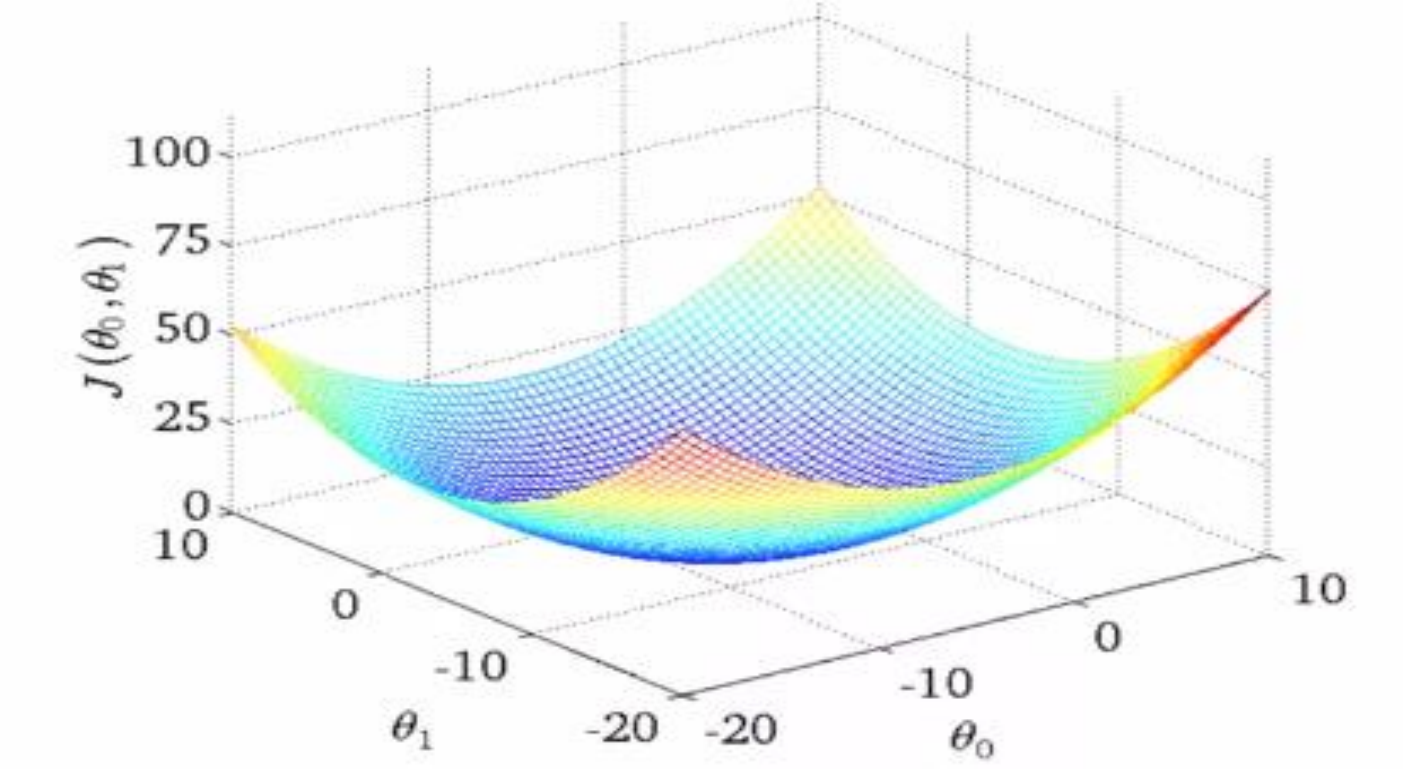
- Plotting several other points yields to the following graph.
- Thus as a goal, we should try to minimize the cost function. In this case, $\vartheta 1=1$ is our global minimum.

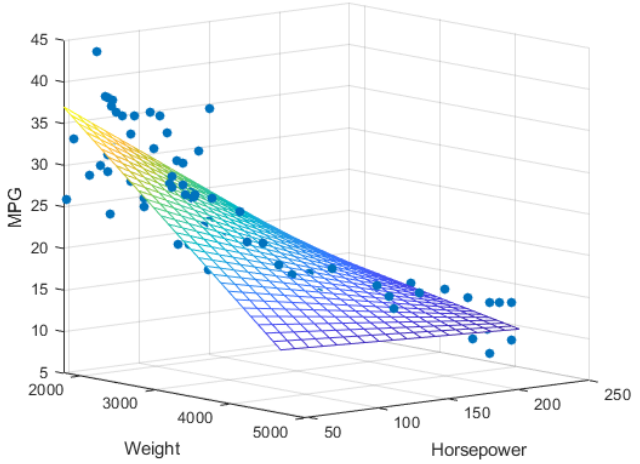


Gradient Descent

- GD is an algorithm for minimizing the cost function J.
- It can be used to minimize other functions also.
- GD used when in a function of J, as theta 0, theta 1,theta 2, up to say some theta n, and you want to minimize theta 0 up to theta n.
- But we will assume one Theta 0 and Theta 1 for better understanding.
- So in GD, we will take any initial values of Theta 0 and Theta 1 and keep on reducing them **a little bit** to minimize J till we find a minimum.

Cost Function in 3D : 2 parameters



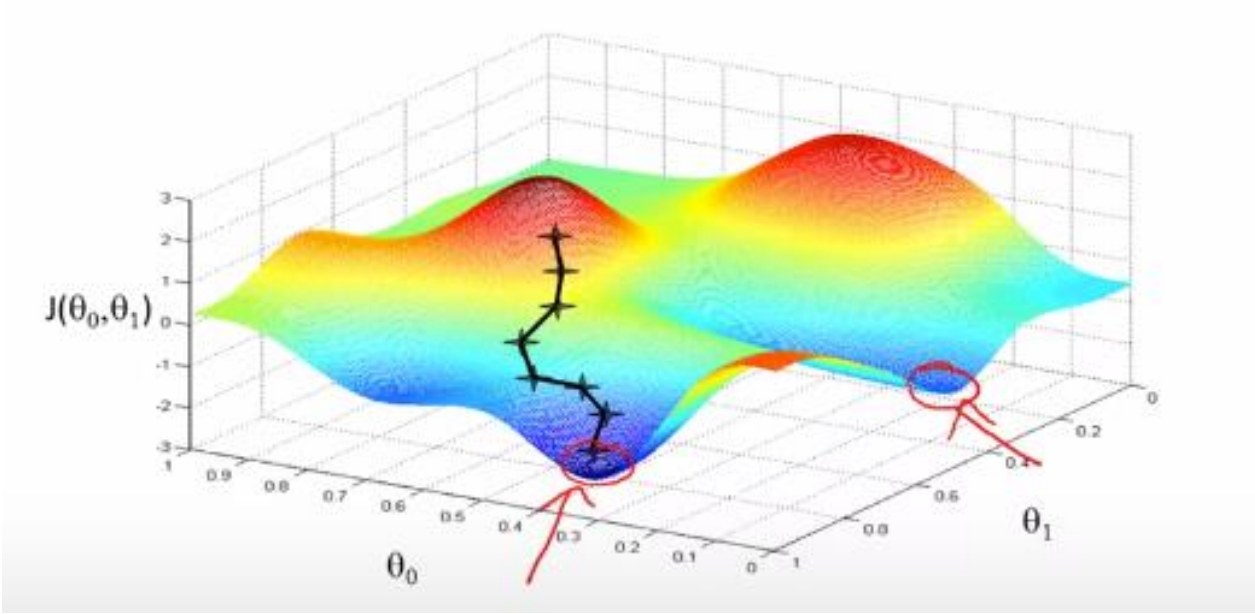


Theta 0 and Theta 1 on the horizontal axes and J is the vertical axis and so the height of the surface shows J and we want to minimize this function. start off with theta 0, theta 1 at some point.

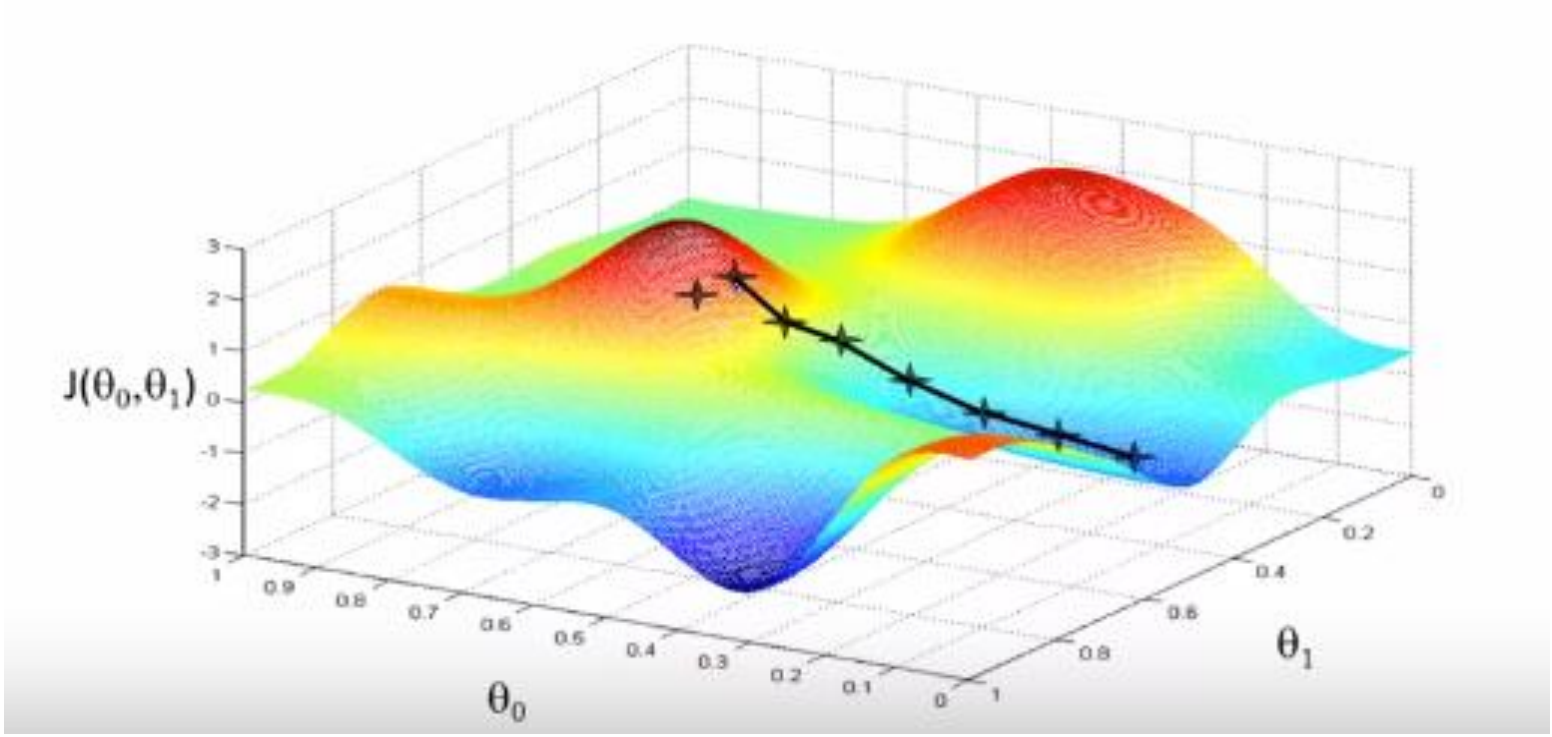
imagine that you are physically standing at any point on the hill

In gradient descent, what we're going to do is we're going to spin 360 degrees to go downhill as quickly as possible,

what direction do I take that little step in?



Gradient Descent - Applied

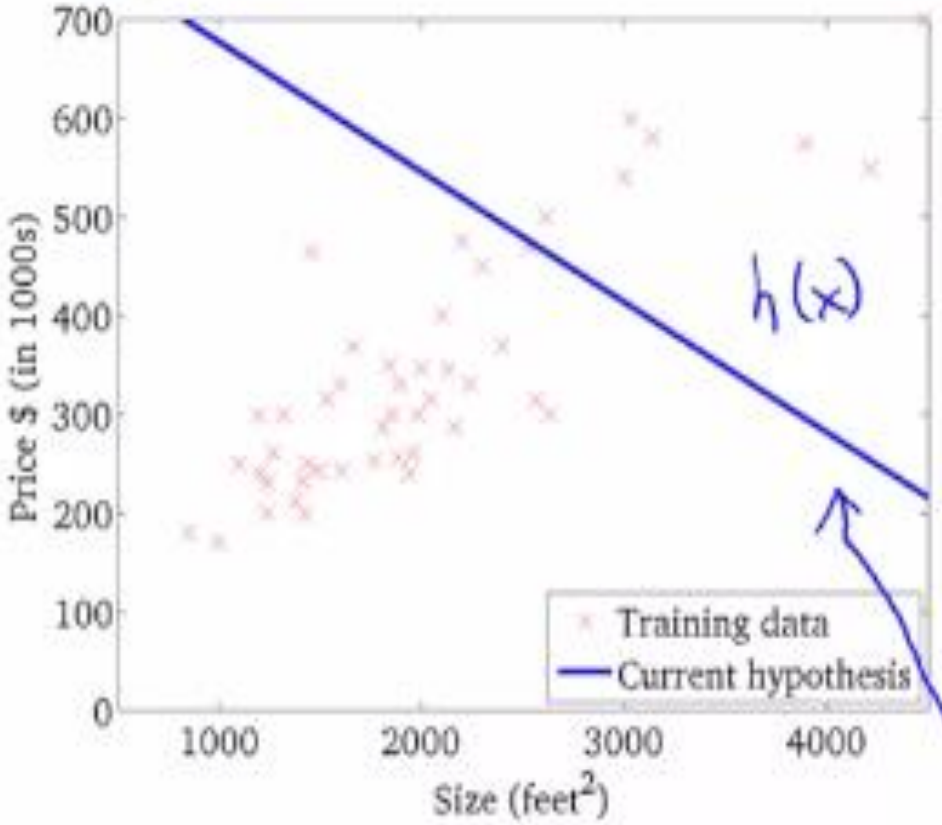


Property of GD

- So if you had started at this first point, you would've wound up at this local optimum,
- But if you started just at a slightly different location, you would've wound up at a very different local optimum.
- And this is a property of gradient descent
- We're going to just repeatedly do this until convergence

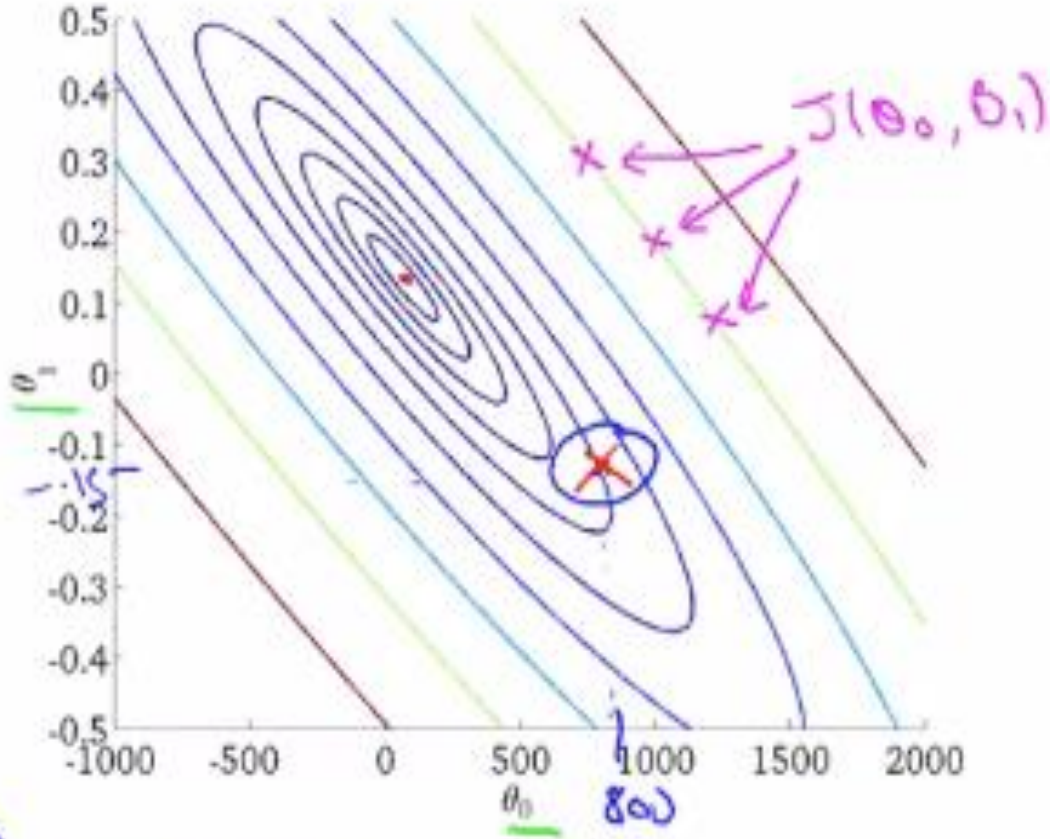
$h_\theta(x)$

(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)



Gradient Descent Algorithm

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

Gradient Descent Algorithm

This alpha here is a number that is called the learning rate. Alpha controls how big a step we take downhill with gradient descent.

repeat until convergence {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) \qquad (\text{for } j = 0 \text{ and } j = 1)$$

}

Correct: Simultaneous update

$\text{temp0} := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$

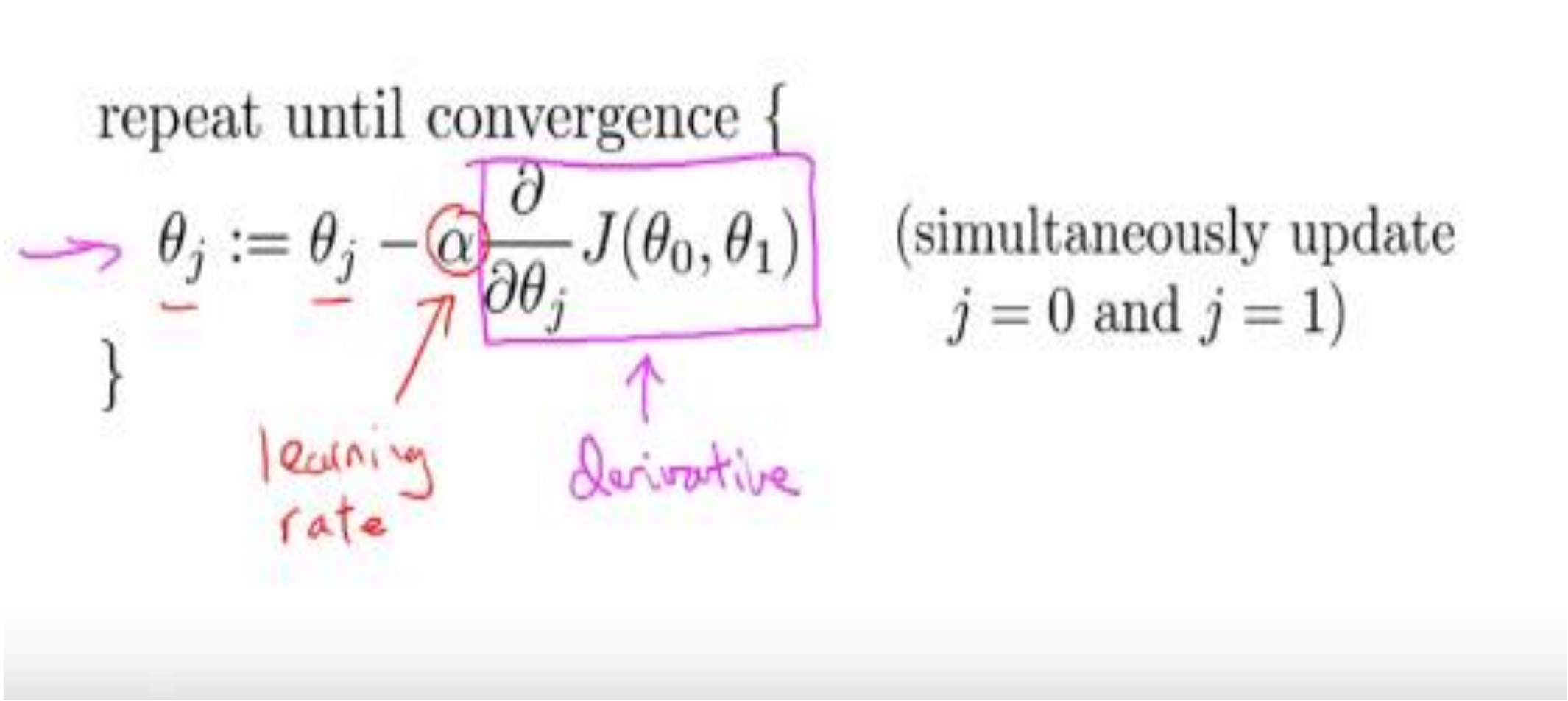
$\text{temp1} := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$

$\theta_0 := \text{temp0}$

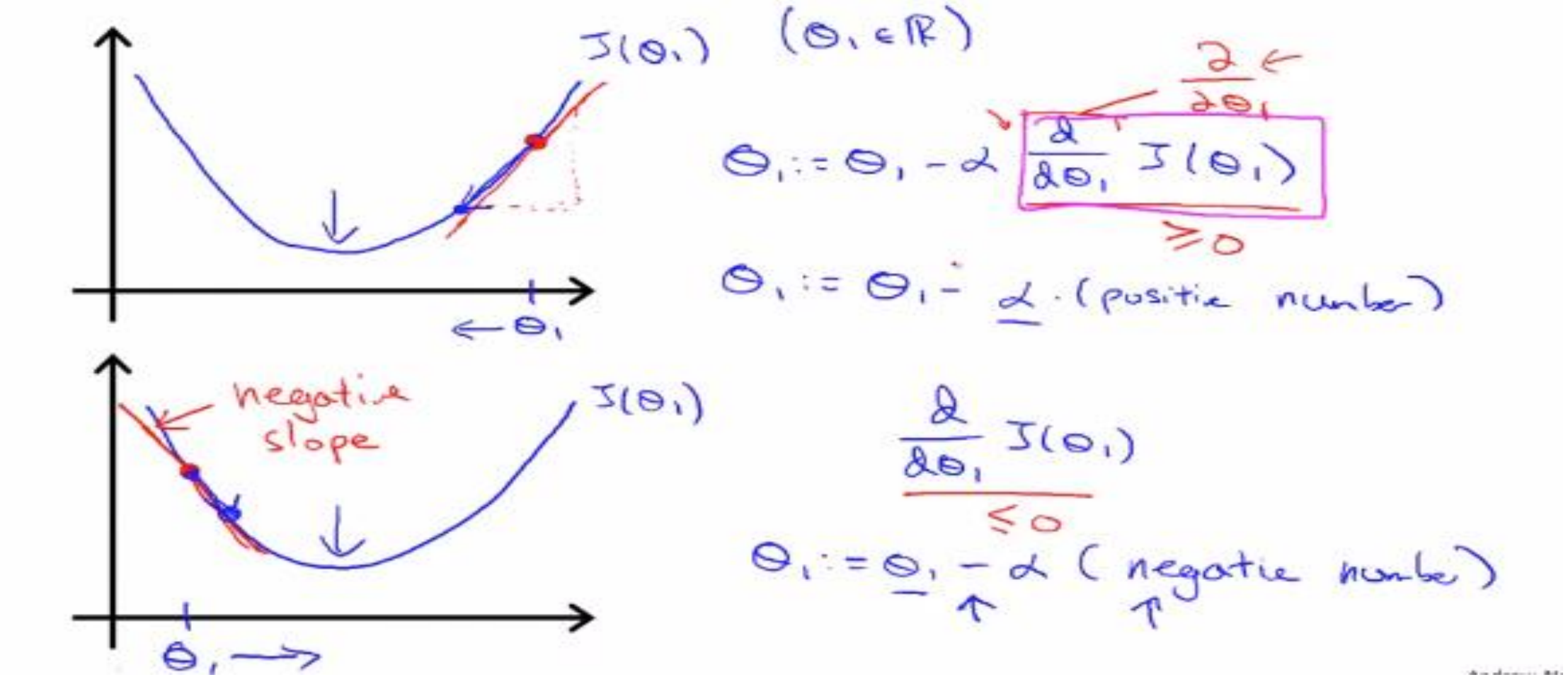
$\theta_1 := \text{temp1}$

Gradient descent algorithm

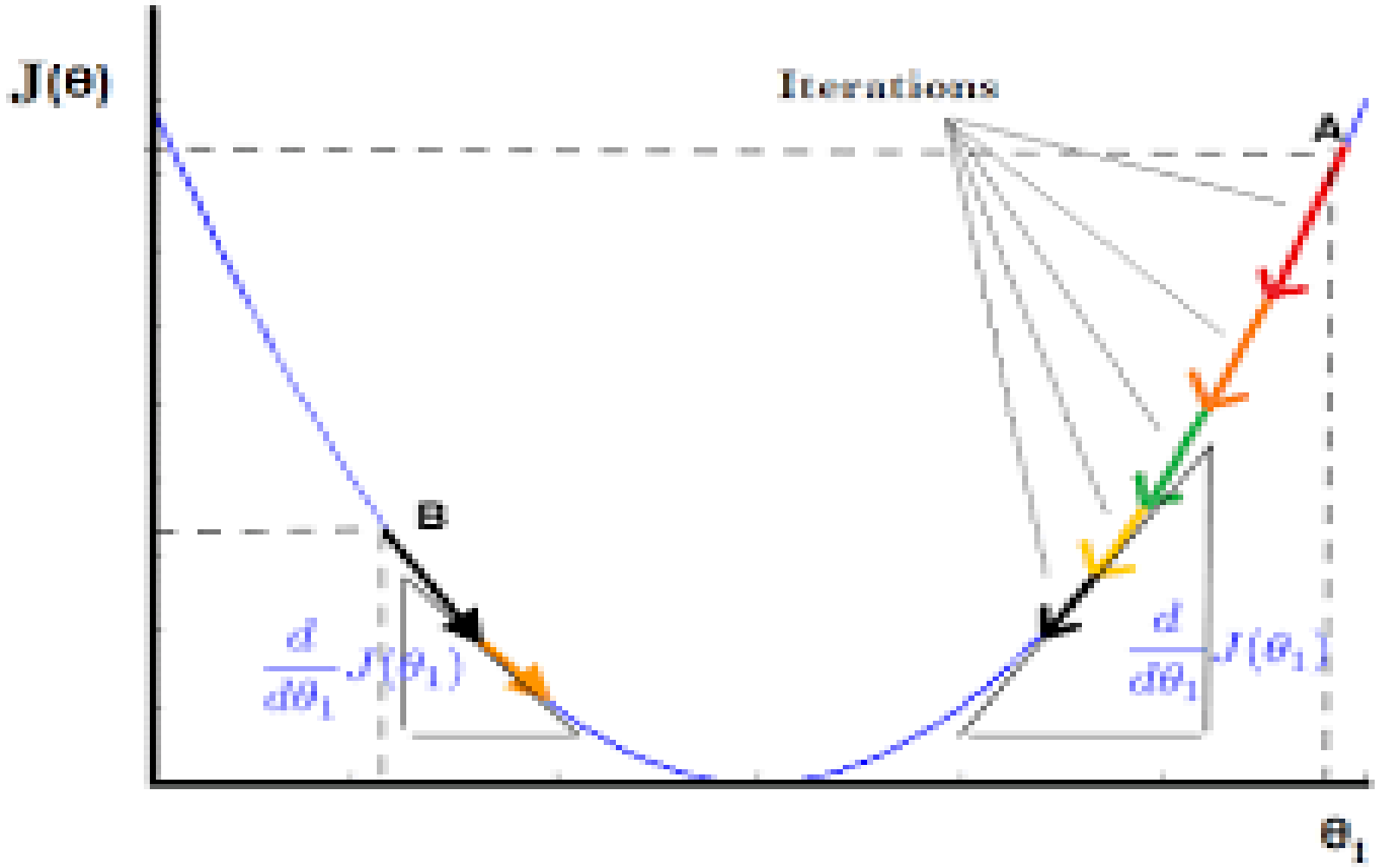
Alpha is a number that is called the learning rate.



Learning Rate Optmization

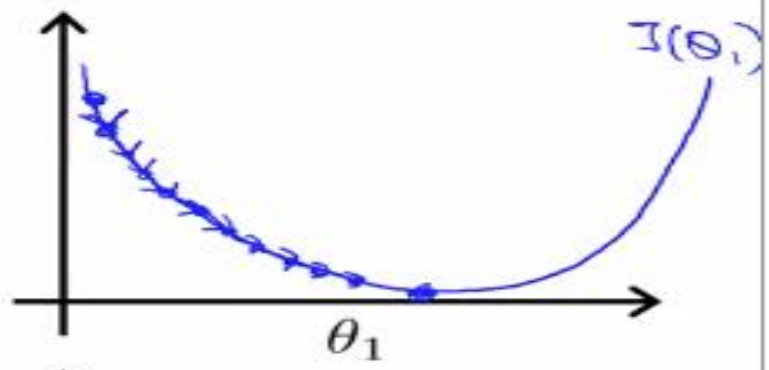


Learing Rate : Values of Alpha



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

If α is too small, gradient descent can be slow.



If α is too large, gradient descent can overshoot the minimum. It may fail to converge, or even diverge.