Houses prices

Size (ft^2)	# bedrooms	# Age of home	# floors	price
x_1	x_2	x_3	x_4	y
—	—	—	—	—
—	—	—	—	—
—	—	—	—	—

$\underbrace{x_1 \quad x_2 \quad x_3 \quad x_4}_{n}$

m

$n =$ No. of features

$m =$ No. of training examples

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

$$h_\theta(x) \begin{bmatrix} h_1(x) \\ h_2(x) \\ \vdots \\ h_m(x) \end{bmatrix} = \begin{bmatrix} x_0^{(1)} & x_1^{(1)} & x_2^{(1)} & \cdots & x_n^{(1)} \\ x_0^{(2)} & x_1^{(2)} & x_2^{(2)} & \cdots & x_n^{(2)} \\ x_0^{(3)} & x_1^{(3)} & x_2^{(3)} & \cdots & x_n^{(3)} \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix}$$

$m * 1$ $\quad m \times n+1 \quad$ $n+1 \times 1$

$$h_\theta(X) = X\theta$$

* Multiple Features

$$h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$$

$x_0 = 1$

$$= \underbrace{[\theta_0 \quad \theta_1 \quad \cdots \quad \theta_n]}_{1\times n+1} \underbrace{\begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix}}_{n+1\times 1}$$

$$= \theta^T x$$

Multivariate Linear Regression

* Gradient Descent for Multiple Features

repeat until Convergence: {

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

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

$\vdots$

}

* Feature Scaling

→ To speed up gradient descent by modifying the range of our inputs. to work on a smaller scale.

$$x_1 = \frac{\text{size}}{2000} \qquad x_1 \, (0 - 2000)$$

$$x_2 = \frac{\text{No. of rooms}}{5} \qquad x_2 \, (1 - 5)$$

\# Divide by the range

Max. Value − Min. Value

So, $\boxed{0 \leq x_i \leq 1}$

* Mean Normalization (Z-Score Normalization)

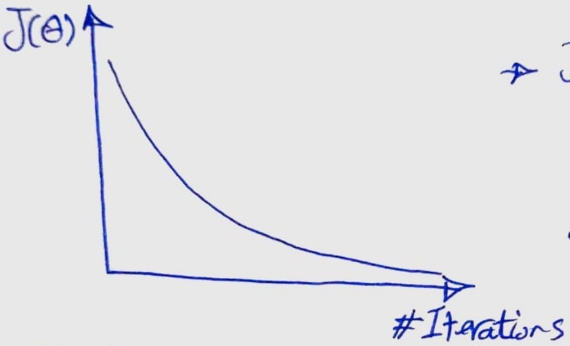
→ It is another method to reduce the steps of gradient descent by subtracting the feature average value from the input and dividing the result by the ~~std~~ deviation

$$x_i := \frac{x_i - \mu_i}{s_i}$$

μ_i → avg. value of x

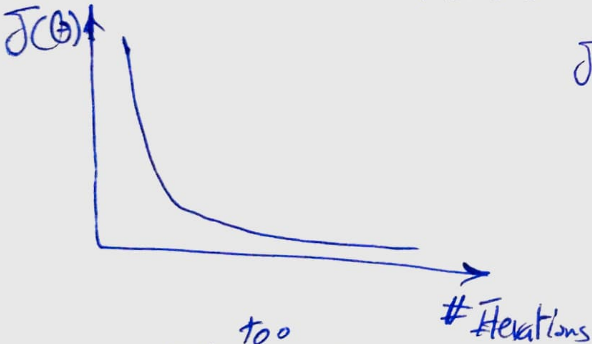
s_i → Standard deviation ~~(Range = Max - Min)~~

* Learning Rate

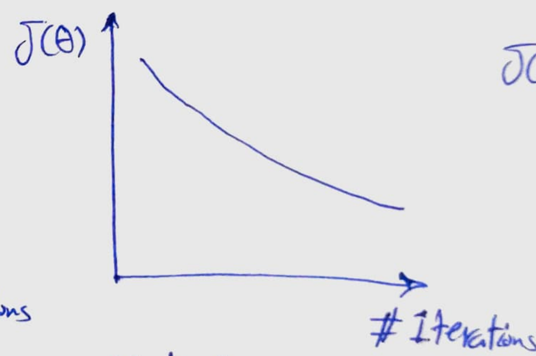


→ J(θ) should decrease every new iterations

→ The speed of J(θ) decreasing depends on α



α is too small
(Slow convergence)



α is small
(more speed)

α is too big

→ But, if α is too small, gradient descent can be slow to converge.

* Features Reformation [Feature Engineering]

→ We can Reform our features to improve Regression

$h(x) = \theta_0 + \theta_1 \underset{\text{Width}}{x_1} + \theta_2 \underset{\text{length}}{x_2} \Rightarrow h(x) = \theta_0 + \theta_1 \underset{\text{Area} = x_1 x_2}{x}$

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

To minimize $J(\theta)$ we take the derivative $\frac{d J(\theta)}{\partial \theta}$ and set it to zero to solve for θ

but if $J(\theta_1, \theta_2, \ldots \theta_m) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$

$$\frac{\partial}{\partial \theta_j} J(\theta) = \cdots \overset{set}{=} 0 \quad \text{(for every } j\text{)}$$

Solve for $\theta_0, \theta_1, \ldots \theta_n$

$\Rightarrow X\theta = y$ <u>where</u> $X \in \mathbb{R}^{m \times n+1}$

$X^T X \theta = X^T y$ $\quad \theta \in \mathbb{R}^{n+1}$

$\quad y \in \mathbb{R}^m$

$$\theta = (X^T X)^{-1} X^T y$$

Let $h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 + \cdots + \theta_n x_n$

We want to minimize the least-square cost function

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

$h_\theta(x) = \theta^T X$

Matrix Notation

$$\text{so, } J(\theta) = \frac{1}{2m} (X\theta - y)^T (X\theta - y)$$

we can simply throw $\frac{1}{2m}$ part away as we will make $\frac{\partial}{\partial \theta} J(\theta) = 0$

$$\text{so, } J(\theta) = (X\theta)^T X\theta - (X\theta)^T y - y^T X\theta + y^T y$$

$$= \theta^T X^T X\theta - 2(X\theta)^T y + y^T y$$

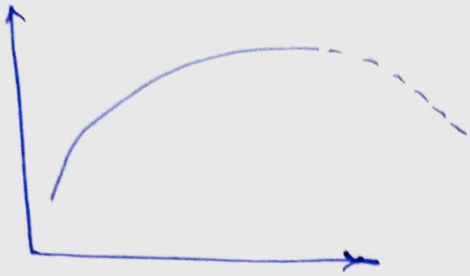
$$\frac{\partial J(\theta)}{\partial \theta} = 0 \Rightarrow \quad = \theta^2 X^T X - 2 \cancel{X^T} y \theta^T + y^T y$$

$$2X^T X\theta - 2X^T y = 0 \Rightarrow X^T X\theta = X^T y \rightarrow \boxed{\theta = (X^T X)^{-1}}$$

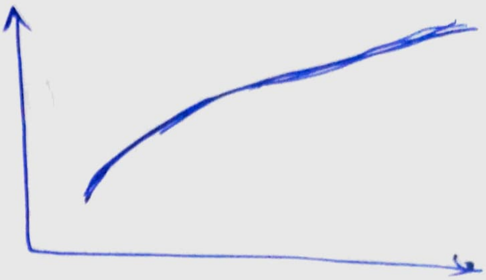
Polynomial Regression

* Linear Regression is not the best to fit the data well.
* We need to change the hypothesis to be a quadratic, Cubic, or Square root ....or any other form

$$h(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_1^2$$

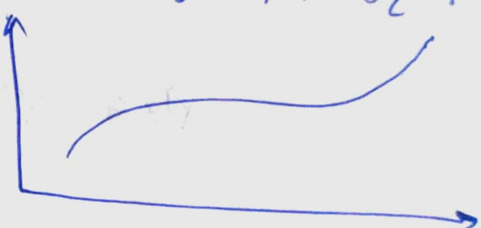


$$h(x) = \theta_0 + \theta_1 x_1 + \theta_2 \sqrt{x_1}$$



* It goes down after awhile, So it is not good for our example (house pricing)

$$h(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \theta_3 x_1^3$$



* Using the square root function may simplify the work more than the Cubic function as the later needs more variables

* The range will be changed according to the new function

if $0 < x < 10$ $\rightarrow$ $0 < x^2 < 100$

$\rightarrow$ $0 < \sqrt{x} < \sqrt{10}$

$0 < x^3 < 1000$

Normal Equation

- To solve for θ analytically

$$\theta = (X^T X)^{-1} X^T y$$

- No need for feature scaling

Design Matrix

$$X = \begin{array}{c} x_0 \\ \begin{bmatrix} 1 & x_1 & x_2 & \cdots \\ 1 & & & \\ 1 & & & \\ \vdots & & & \end{bmatrix} \end{array}, \quad y = \begin{bmatrix} - \\ - \\ - \\ - \end{bmatrix}$$

$m \times (n+1)$ — m_vector

n → No. of features

m → No. of training examples

Gradient Descent	Normal Equation
Need to choose α	No need to choose α
Need several iterations	No need to iterate
$O(kn^2)$	$O(n^3)$, need to calculate $(X^T X)^{-1}$
Works well when n is large (10,000 ↑↑)	Slow if n is very large

★ What if $(X^T X)$ isn't inversible? [Singular = Non invertible]

- This may occure due to:
  1. Redundant features (linearly dependent)
     - x_1 = area in ft^2
     - x_2 = area in m^2
     - → delete one
  2. Too many features ($m \leq n$)
     - delete some features
     - Regularization

→ For Octave: P-inv → Pseudo inverse willy compute $(X^T X)$ even it is Non invertible