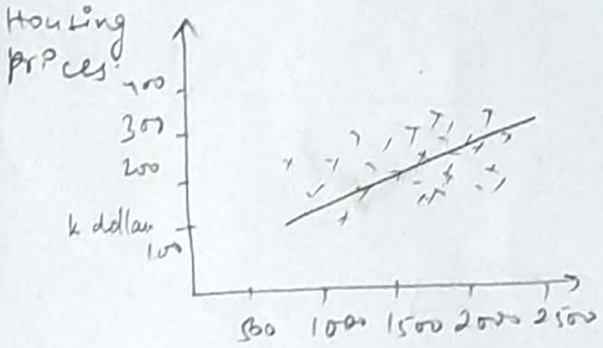
⇒ Model Representation

Supervised learning - Given the 'right answer' for each example in data

↳ Regression problem → Predict real valued output

↳ classification → Discrete valued output



→ supervised learning, we have training set.

ex.

Size (x)	Price (y)	
2104	460	m = 47
1416	232	
1534	315	
852	178	
...	...	

m → No° of training examples

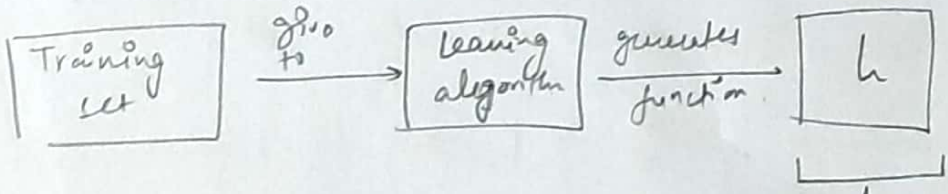
x's → "input" variable / features

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

(x, y) → one training example

$(x^{(i)}, y^{(i)})$ → i^{th} training example. for ex $x^{(2)} = 1416$, $y^{(1)} = 460$ etc.

working



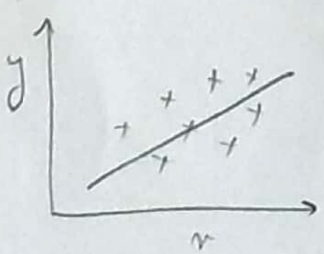
prediction function generated by algorithm

Size of house (x) → h (hypothesis) → estimate price (estimated value of y).

So, h maps x's to y's

→ Representing h.

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



$h(x) = \theta_0 + \theta_1 x$

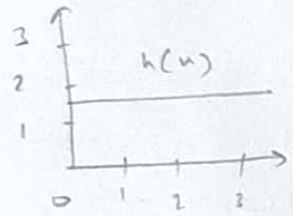
Linear regression with one variable

Univariate linear regression.

⇒ cost function → helps to determine best possible straight line for data

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

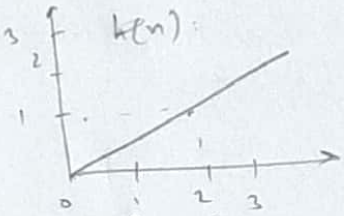
θ_i's → Parameters , How to choose θ_i's ?



$\theta_0 = 1.5$

$\theta_1 = 0$

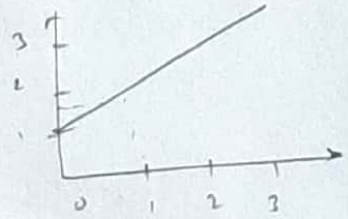
$h(x) = 1.5$



$\theta_0 = 0$

$\theta_1 = 0.5$

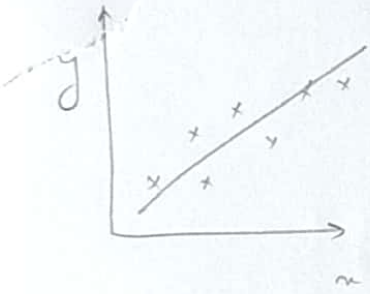
$h(x) = 0.5x$



$\theta_0 = 1$

$\theta_1 = 0.5$

$h(x) = 1 + 0.5x$



↳ we need to find θ_0, θ_1 so that $h_\theta(x)$ is close to y for our training examples (x, y) (determine best fit)

So, we need to solve a minimisation problem

i.e. minimise θ_0, θ_1 such that $(h_\theta(x) - y)^2$ is small.

we need to sum over the training set.

So minimise $\frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$ wrt θ_0, θ_1.

why? ← (2) ↳ average.

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

we need to minimise $J(\theta_0, \theta_1)$ } → cost function

↳ squared error function

$\left(\frac{1}{2}\right)$ is put just as a convenience for computation of gradient descent.

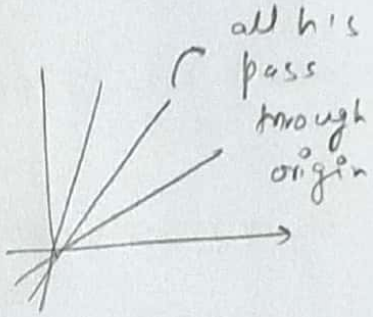
⇒ Cost function Intuition

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

θ_0, θ_1 — parameters

$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$ — cost function.

goal → minimize $(J(\theta_0, \theta_1))$

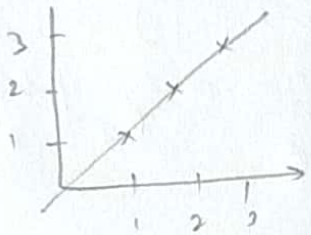


let's take $h_\theta(x) = \theta_1 x$. i.e $\theta_0 = 0$ ∴

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

↓ minimize wrt θ_1.

$h_\theta(x)$ → for fixed θ_1, this is a fn of x



Training set $(1,1), (2,2), (3,3)$

$\theta_1 = 1$. let's calculate $J(\theta_1)$ for $\theta_1 = 1$.

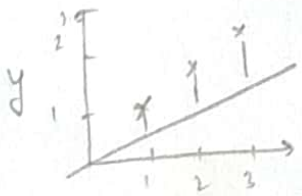
$J(\theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x)^{(i)} - y^{(i)})^2$

$= \frac{1}{2m} \times 0 = 0.$ $\boxed{J(1) = 0}$

so $J(1) = 0$

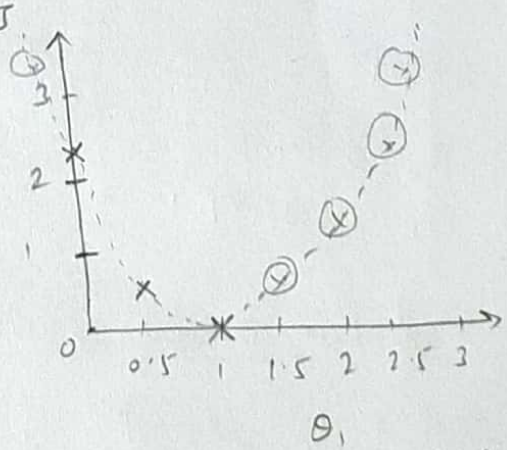
let's plot it over J and θ curve J.

let's take $\theta_1 = 0.5$



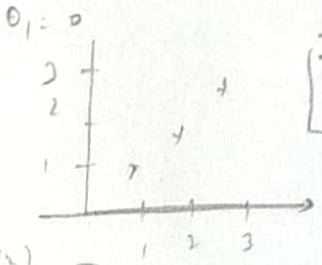
$J(0.5) = \frac{1}{2m}\left((0.5-1)^2 + (1-2)^2 + (1.5-3)^2\right)$

$= \frac{1}{6} \times 3.5 = \boxed{0.58 = J(0.5)}$



$J(0) = \frac{1}{6}(1^2 + 2^2 + 3^2)$

$\frac{14}{6} = \frac{7}{3}$



$\boxed{J(0) = \frac{7}{2}} = 2.333$

So, each value of θ_1 corresponds to a different straight line $h_\theta(x)$ and thus a different J is obtained

Our objective is to minimize $J(\theta_1)$ and find value of θ_1 so that J is minimum.

We can see that $\theta_1 = 1$ gives best possible straight line fit.

⇒ Cost function intuition - 2

$\theta_0 = 1 + \sqrt{1 \cdot 2}$ $\quad 1+\sqrt{2}$

$\theta_1 = 2 + \sqrt{2}$

⇒ Gradient Descent

↳ Here, we will use GD to minimize an arbitrary function J.

Q. we have $J(\theta_0, \theta_1)$ → Outline → start with some θ_0, θ_1

want $\min_{\theta_0, \theta_1} J(\theta_0, \theta_1)$ → keep changing θ_0, θ_1 to reduce $J(\theta_0, \theta_1)$ until we hopefully arrive at minimum

say $(\theta_0 = 0, \theta_1 = b)$.

GD algorithm

$:=$ Assignment operator

$a := a+1$

| Truth assertion
| $a = b$ ✓
| $a = a+1$ ✗

repeat until convergence. {

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

* Simultaneously update θ_0 and θ_1.

} learning rate (defines step size)

correct: simultaneous update

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

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

$\theta_0 := temp0$

$\theta_1 := temp1$

Incorrect

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

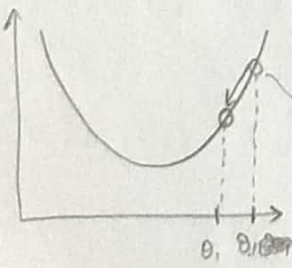
$\theta_0 := temp0$

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

$\theta_1 := temp1$

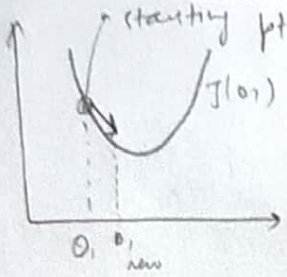
⇒ Gradient Descent Intuition

Let's take example of one parameter θ_1 $J(\theta_1)$ $\theta_1 \in \mathbb{R}$.



$\theta_1 := \theta_1 - \alpha \left[\frac{d}{d\theta_1} J(\theta_1)\right]$ → positive on starting value.

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



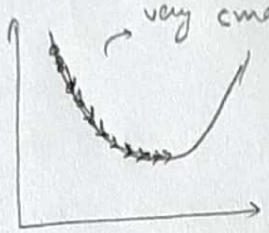
$\theta_1 := \theta_1 - \alpha \frac{dJ(\theta_1)}{d\theta_1}$ → -ve in this case

→ total +ve.

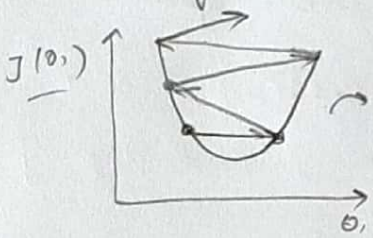
So, θ_1 is greater than previous value.

→ $\theta_1 := \theta_1 - \alpha \frac{dJ(\theta_1)}{d\theta_1}$

If α is too small, gradient descent can be slow

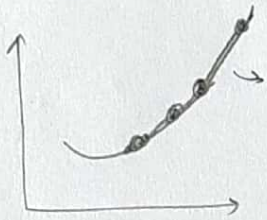


→ If α too large, GD can overshoot minimum. It may not converge or even diverge



→ GD can converge to a local minimum, even with learning rate α fixed.

$$\theta_1 := \theta_1 - \alpha \frac{dJ(\theta_1)}{d\theta_1}$$



→ So, as we approach min, GD automatically takes smaller steps. So we don't need to change α.

⇒ Gradient Descent for Linear Regression

GD algorithm

repeat until convergence {

$$\theta_j := \theta_j - \alpha \frac{dJ(\theta_0,\theta_1)}{d\theta_j}$$

(for $j=1$ and $j=0$)

}

Linear regression model

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

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

We need to figure out the partial derivative term now.

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

$$= \frac{d}{d\theta_j} \frac{1}{2m} \sum_{i=1}^{m} \left(\theta_0 + \theta_1 x^{(i)} - y^{(i)}\right)^2$$

θ_0 : $j=0$; $\frac{dJ(\theta_0,\theta_1)}{d\theta_0} = \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)$

θ_1 : $j=1$; $\frac{dJ(\theta_0,\theta_1)}{d\theta_1} = \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right) \cdot x^{(i)}$

Now, we plug these into GD algo

GD algorithm for linear regression

repeat until convergence {

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

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

} → update θ_0 and θ_1 simultaneously

GD is susceptible to local optima but to our rescue, cost function would always be a convex function, thus, it will only have one global optima.

"Batch" Gradient Descent

'Batch' → Each step of GD uses all the training examples

↳ $\sum_{i=1}^{m}$

⇒ Vector

vector → An $n \times 1$ matrix.

$y = \begin{bmatrix} 460 \\ 232 \\ 315 \\ 178 \end{bmatrix}$ ← $n = 4$ → 4 D vector. $\mathbb{R}^4$

y_i = i^{th} element of vector.

$y_1 = 460$

$y_4 = 178$

1 indexed v/s 0 indexed

We use 1 indexed, unless specified.

$y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix}$ $y = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{bmatrix}$ ← $y[0]$

$y[1]$ →

A, B, C, X → capital for matrices.

a, b, x, y → numbers, variables, vectors

⇒ $A \times x = y$

$\begin{bmatrix} \quad \end{bmatrix} \times \begin{bmatrix} \quad \end{bmatrix} = \begin{bmatrix} \quad \end{bmatrix}$

$m \times n$ matrix · $n \times 1$ matrix = $m \times 1$ matrix

→ House sizes:

2104
1416
1534
852

$h_\theta(x) = -40 + 0.25x$

$\begin{bmatrix} -40 \\ 0.25 \end{bmatrix}$

Datamatrix ↑

$\begin{bmatrix} 1 & 2104 \\ 1 & 1416 \\ 1 & 1534 \\ 1 & 852 \end{bmatrix} \times \begin{bmatrix} -40 \\ 0.25 \end{bmatrix} \Rightarrow \begin{bmatrix} -40 \times 1 + 0.25 \times 2104 \\ \text{Price 2} \\ \text{Price 3} \\ \text{Price 4} \end{bmatrix}$

parameters

Prediction

4×2 · 2×1

So, in Octave

Datamatrix =

Prediction = Datamatrix * Parameters