Linear Regression

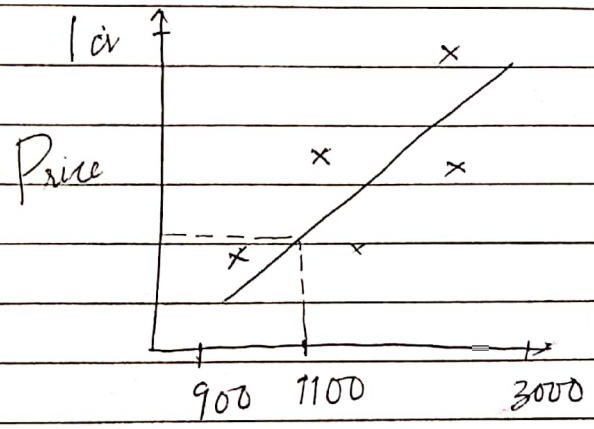
Problem – House Price Prediction

$x = 1$ → input features

Input → Size (Sqft) | Price (lakhs) → Output / Target Variable

Size (Sqft)	Price (lakhs)
900	12
1200	20
1500	30
1800	40
2400	50

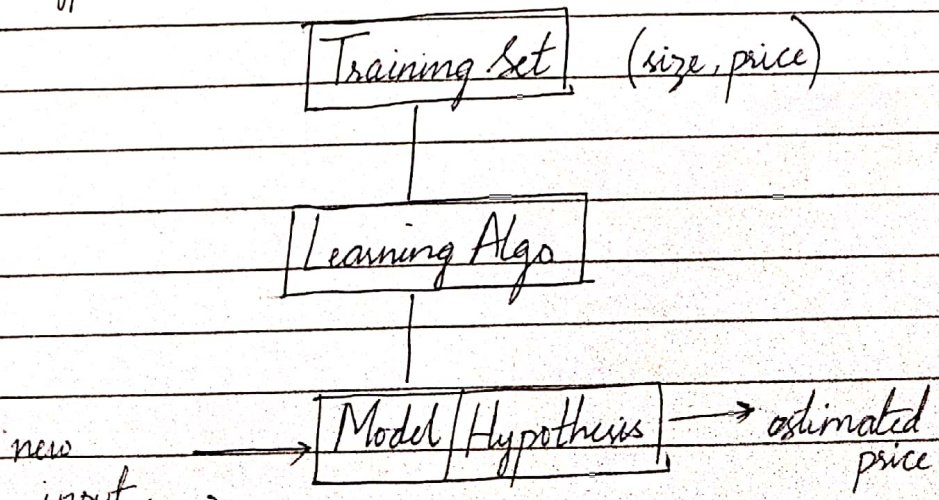
$m = 5$ training examples



Trying to find the best fit line

Notations: Training set (x, y)
input / features (x)
output / target variable (y)
m = # training examples

Hypothesis / Model



Representation of our Hypothesis

$y = mx + c$ (equation of line)

slope, intercept

$h(x) = w_0 + w_1 x_1$ → linear for single variable

$h(x) = w_0 x_0^{1} + w_1 x_1$ $\quad w$ = Parameters / Weight (initialized to 0 / random value)

$h(x_1, x_2) = w_0 + w_1 x_1 + w_2 x_2$

↓ in case of two input features (size, BHK)

Generalized → $\boxed{h(X) = \sum_{i=0}^{n} w_i x_i \quad x_0 = 1}$

↑ vectorized form of input features

$$w = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix} \quad X = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix}$$

Job of learning algorithm is to find best value of Parameters / Weights

→ find / choose W such that $h(x) \approx y$ for the given training set

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

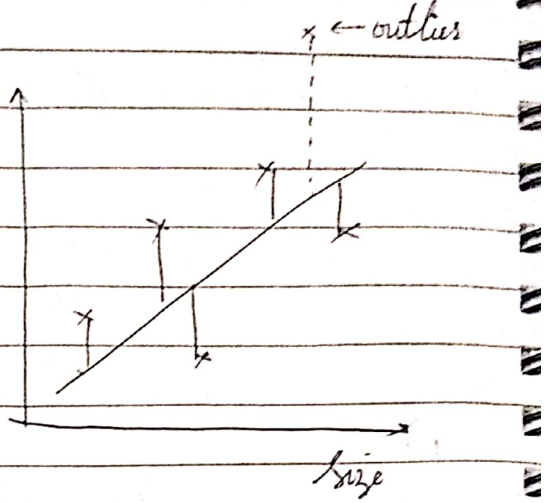
Cost Function

↳ purpose: Identify quality of your result

loss → single data point / example

cost → over all the data points / batch

Loss function



$h(x)$ → predicted value

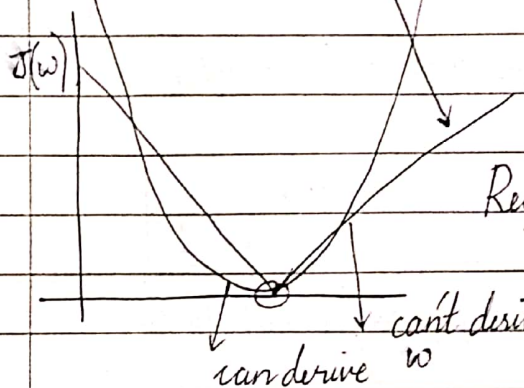
y → ground truth / original value

$h(x) - y$ — margin

$|h(x) - y|$ — absolute loss.

$(h(x) - y)^2$ — squared loss

↳ emphasizes distance b/w $h(x)$ and y



Regression loss functions

Loss and Accuracy (difference)

y	predicted o/p ($h(x)$)	
0	0.2	0
1	0.8	1
1	0.3	0
0	0.1	0
1	0.9	1

$T = 0.5$

if $h(x) \geq 0.5$ = 1

else = 0

$4/5 \times 100$ → Accuracy (Classification)

$(0.2 - 0)^2 + (0.8 - 1)^2 + \ldots$ → loss (Regression)

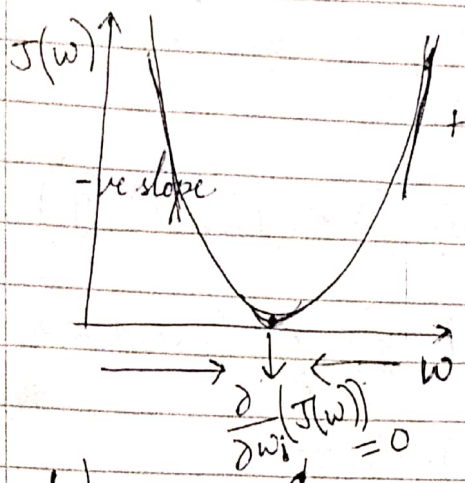
(correctness)

Learning Algo: Gradient Descent

$y = (3x_1 + 2x_2)^2$

$\frac{\partial y}{\partial x_1} = 2(3x_1 + 2x_2)^{2-1} \times 3$

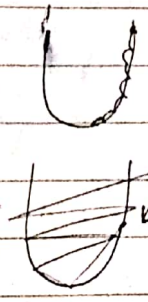
$\frac{\partial y}{\partial x_2} = 2(3x_1 + 2x_2)^{2-1} \times 2$



$w_i = w_i - \alpha \frac{\partial}{\partial w_i}(J(w))$

partial derivative of $J(w)$ slope at that pt

learning rate:
how fast shall we approach the minima.



↳ smaller α → smaller steps

larger α → larger steps

To see if gradient descent is working
$J(\theta)$ → decrease at each iteration
else adjust α

Summary:

$w \leftarrow$ find a best $w \leftarrow [w_0, w_1, w_2 \cdots]$

minimize $J(w) = \frac{1}{2}\sum_{i=1}^{m}\left(h(x)^i - y^{(i)}\right)^2$ ← squared loss

$h(x) = w_0x_0 + w_1x_1 + w_2x_2 \cdots w_mx_m$

$w_i = w_i - \alpha \frac{\partial}{\partial w_i}(J(w))$

Gradient Descent Update Rule

$W = [w_0, w_1, w_2 \dots]$

$$J(w) = \frac{1}{2}(h_w(x) - y)^2$$

$$\frac{\partial J(w)}{\partial w_i} = 2 \times \frac{1}{2}(h_w(x) - y)\frac{\partial}{\partial w_i}(h_w(x) - y)$$

$$= (h_w(x) - y) \cdot \frac{\partial}{\partial w_i}(w_0 x_0 + w_1 x_1 \cdots w_i x_i \cdots w_n x_n - y)$$

$$\boxed{\frac{\partial J(w)}{\partial w_i} = (h_w(x) - y) \cdot x_i}$$

$$w_0 \longrightarrow w_0 = w_0 - \frac{\partial}{\partial w_0}(J(w))$$

$$w_0 = w_0 - (h_w(x) - y) \cdot x_0.$$

Generalizing →

$$w_i = w_i - \alpha(h_w(x) - y)x_i$$

$$J(w) = \frac{1}{2}\sum_{i=0}^{m}(h_w(x)^{(i)} - y^{(i)})^2.$$

$$w_i = w_i - \alpha \sum_{i=1}^{m}(h_w x^{(i)} - y^{(i)}) \cdot x_j$$

n → size BHK y

m = # training examples - no. of rows.

n = # features - no. of columns

m { - - - - - }

ML Life Cycle:

Data → Preprocess Data

↓

Split Data (Train Test Validation)

↓

Decide Problem Type (Regression/Classification)

↓

Decide loss function

↓

Decide Hypothesis (Model Structure) → LA (gradient descent)

Train Model

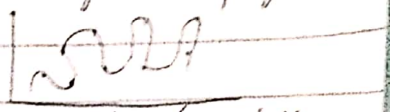
↓

Check for Testing set

↓

Satisfied? — No → (back to Train Model) ; Yes → Finish!

higher-order polynomial



Regularization → addresses high variance problem (overfitting problem)

L_1 → Lasso Regression → adds 'absolute value of magnitude'

L_2 → Ridge Regression → adds 'squared magnitude'

training accuracy ↑

testing accuracy

$$Loss(s) = \underbrace{\sum_i Loss(\hat{y}_i, y_i)}_{\text{minimize}} + \lambda \sum_j |w_j|$$

λ: regularization parameter (not important) ↓

w: updates weights ↑

$|w|$: +ve, -ve, 0 → close to

L_1 norm $\|w\|_1$

$(|w_1| + |w_2| + \cdots |w_n|)$

L_2 norm $\|w\|_2$

$(|w_1|^2 + |w_2|^2 \cdots |w_n|^2)^{1/2}$

→ Vanilla (Batch) G.D → slower but smooth

$$w_i = w_i - \alpha \frac{\partial}{\partial \text{ w.r.t } w_i} J(\theta)$$

$$\frac{1}{m} \sum_{i=1}^{m} (\hat{y}_i - y_i) x_i$$

→ Stochastic G.D.

→ faster but uneven (zig-zag) noise

```
for i in range(m):
    w_i = w_i - α(ŷ(i) - y(i)) x_i
```

→ Mini Batch Gradient Descent

```
for i in range(0, m, 1000):        # if mini batch size = 1000
    w_i = w_i - α Σ_{i=0}^{p-1000} (h_w(i) - y(i)) x_i
```

instead of single sample in SD → we take n data points in each iteration