Week_1 Machine_learning

Day-1 **Machine Learning**

↳ Field of study that gives computers the ability to learn without being explicitly programmed

Arthur Samuel (1959)

- **Supervised learning** $x \rightarrow y$

input ⇝ output label

learn from being given "right answers"

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

Regression :- predict numbers or values {House price}

Classification:- predict categories {Breast cancer}
small number of outputs {0, 1, 2}

- **Unsupervised learning**

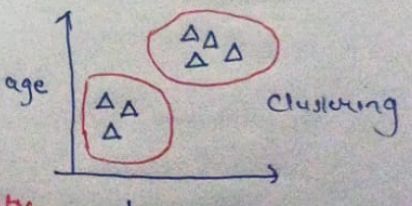
finding something interesting in unlabelled data.

↳ Basically we don't know our data belongs from which part for e.g.

Clustering: Google news.

& Data only comes with inputs x, but not output label y.

Algorithm has to find structure in data

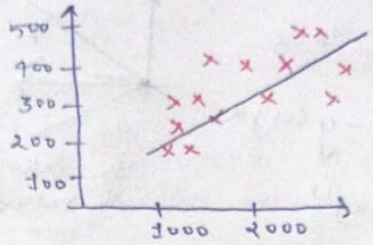


Anomaly detection
Find unusual data points

Dimensionality reduction
Compress data using fewer numbers

Regression Model

Predict Numbers



Linear Regression

Terminology

Training set: Data used to train model.

x = "input" variable features

y = "output" or target variables

m = number of training ex

(x, y) ⇒ single training example

$(x^{(i)}, y^{(i)})$ ⇒ i^{th} train example

Training set → Learning Algo → f

x → f → $\hat{y}$

features — hypothesis function

How to represent f?

$f_{w,b}(x) = wx + b$

$F(x)$

size → F → price (estimated)

Cost Function

Q. Find w, b:

$\hat{y}^{(i)}$ is close to $y^{(i)}$ for all $(x^{(i)}, y^{(i)})$.

$$J(w,b) = \frac{1}{2m} \sum_{i=1}^{m} (\hat{y}^{(i)} - y^{(i)})^2$$

error — or squared error cost function

$$J(w,b) = \frac{1}{2m} \sum_{i=1}^{m} (f_{w,b}(x^{(i)} - y^{(i)})^2$$

goal: minimize $J(w,b)$

cost function

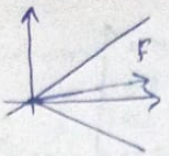
Simplified Model

$f_w(x) = w(x)$ $\quad b = 0$

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

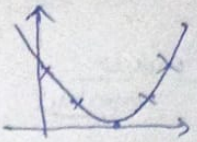
$w(x)^{(i)}$

minimize $J(w)$
w



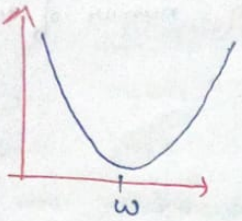
By taking values

there $b = 0$



$J(w)$

J (function of w, b)



shape similar to soup bowl in 3-D

Gradient descent

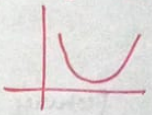
Have function $J(w,b)$. → cost function for linear Regression

Want $\min_{w,b} J(w,b)$

outline: Start with some w, b (set $w=0, b=0$)

- Keep changing w, b to reduce $J(w,b)$
- Until we settle at or near a minimum

J not always



may have > 1 minimum

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

α → Learning rate

(Here = is Assignment)

derivative term help us for in which direction we should take baby step to come down till.

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

✗ Correct: simultaneous update

$tmp\_w = w - \alpha \frac{\partial}{\partial w} J(w,b)$

$tmp\_b = b - \alpha \frac{\partial}{\partial b} J(w,b)$

$w = tmp\_w$

$b = tmp\_b$

Incorrect:

$tmp\_w = w - \alpha \frac{\partial}{\partial w} J(w,b)$

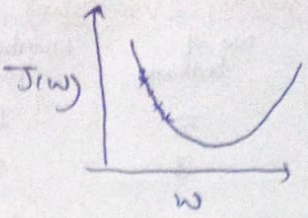
$w = tmp\_w$

$tmp\_b = b - \alpha \frac{\partial}{\partial w} (J(w,b)$

$b = tmp\_b$

Learning Rate

$$w = w - \alpha \frac{d}{dw} J(w)$$



If α is too small ...
Then we are taking very minute baby step until you finally approach minimum
But it will be very slow.

If α is too large ...
It will take huge step and this will create worse condition
Gradient descent may:
- overshoot, never reach minimum
- fail to converge, diverge

Calculus Part-

$$\frac{\partial}{\partial w} J(w,b) = \frac{\partial}{\partial w} \frac{1}{2m} \sum_{i=1}^{m} (f_{w,b}(x^{(i)}) - y^{(i)})^2 = \frac{\partial}{\partial w} \frac{1}{2m} \sum_{i=1}^{m} (wx^{(i)} + b - y^{(i)})^2$$

$$= \frac{1}{2m} \sum_{i=1}^{m} (wx^{(i)} + b - y^{(i)}) x^{(i)} = \frac{1}{m} \sum_{i=1}^{m} (f_{w,b}(x^{(i)}) - y^{(i)}) x^{(i)}$$

$$\frac{\partial}{\partial b} (J(w,b)) = \frac{\partial}{\partial b} \frac{1}{2m} \sum_{i=1}^{m} (f(w,b)(x^{(i)} - y^{(i)})^2 = \frac{\partial}{\partial b} \frac{1}{2m} \sum_{i=1}^{m} (wx^{(i)} + b - y^{(i)})^2$$

$$= \frac{1}{2m} \sum_{i=1}^{m} (wx^{(i)} + b - y^{(i)}) \cdot 2 = \frac{1}{m} \sum_{i=1}^{m} (f_{w,b}(x^{(i)}) - y^{(i)})$$

Multiple features: (Variables)

ex

	Size in feet²	No. of bedrooms	Number of floor	Age of home	Price $(1000)
$x^{(1)}$	2104	5	1	45	460
$x^{(2)}$	1416	3	2	40	232
	x_1	x_2	x_3	x_4	

x_j = j^{th} feature

n = number of features

$x^{(i)}$ = features of i^{th} training example

Model: $f_{w,b}(x) = w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + b$

Ex: $f_{(w,b}(x) = 0.1x_1 + 4x_2 + 10x_3 + -2x_4 + 80$ ↘ base price

$\vec{w} = [w_1\ w_2\ w_3 \cdots w_n]$ parameters of Model

b is a number

$\vec{x} = [x_1, x_2, x_3 \cdots x_n]$

$$f_{\vec{w},b}(\vec{x}) = \vec{w} \cdot \vec{x} + b$$

- Vectorization: This make code shorter and run more smoothly

Parameters and features

$\vec{w} = [w_1\ w_2\ w_3]$ $n=3$

b is a number

$\vec{x} = [x_1\ x_2\ x_3]$

Code:

```
w = np.array([1.0, 2.5, -3.3])
b = 4
x = np.array([10, 20, 30])
```

(without Vectorization)

```
f = 0
for j in range(0, n)
    f = f + w[j] * x[j]
f = f + b
```

$$f_{w,b}(\vec{x}) = \left(\sum_{j=1}^{n} w_j x_j\right) + b$$

Vectorization $f_{w,b}(\vec{x}) = \vec{w} \cdot \vec{x} + b$ ∴

```
f = np.dot(w,x) + b
```

Gradient descent for Multiple feature

$$w_1 = w_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} (f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)}) x_1^{(i)}$$

Normal Equation

- only for linear regression
- Solve for w, b without iterations

- Feature Scaling

$\hat{price} = w_1 x_1 + w_2 x_2 + b$

x_1: size, x_2: # bedrooms

House:- $x_1 = 2000$, $x_2 = 5$, price = $500k

Here value is large, So model will choose small value of w_1

w_2 → will be large

If the cost function is increasing, we know that gradient descent is diverging, so we need a lower learning rate or vice-versa

Feature engineering.

$f_{w,b}(\vec{x}) = w_1 x_1 + w_2 x_2 + b$ } creating a new features

x_1: frontage, x_2: depth

area = frontage × depth

$x_3 = x_1 x_2$

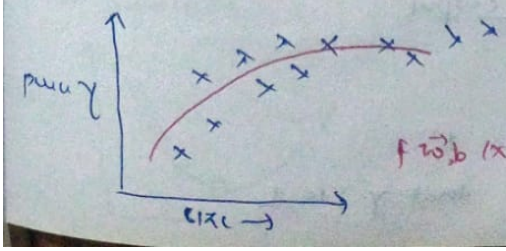
new feature

$f_{\vec{w},b}(\vec{x}) = w_1 x_1 + w_2 x_2 + w_3 x_3 + b$

Polynomial regression

$f_{\vec{w},b}(x) = w_1 x + w_2 x^2 + w_3 x^3 + b$

size, size², size³



$f_{\vec{w},b}(x) = w_1 x + w_2 x^2 + b$

size, size²