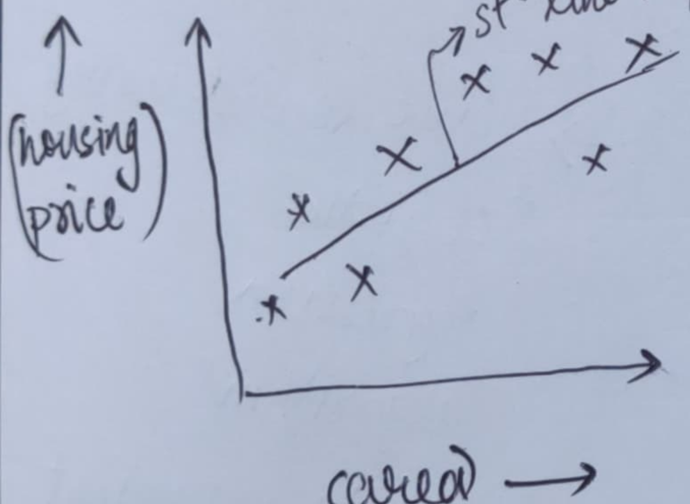
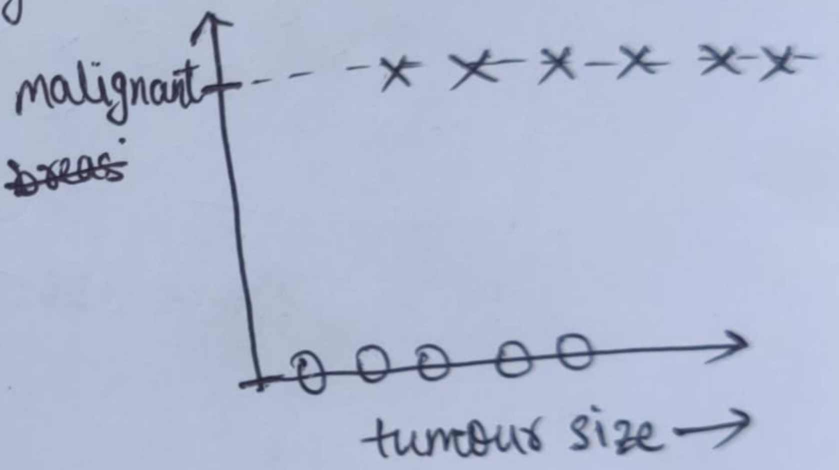
Machine Learning defn: →

program learns from experience E wrt task T and performance measure P, IF its performance on T, as measured by P improves with experience E.

Supervised Learning → most common type of ML.

$\boxed{\text{given } (X, Y) \text{ find } X \to Y}$ ↳ mapping

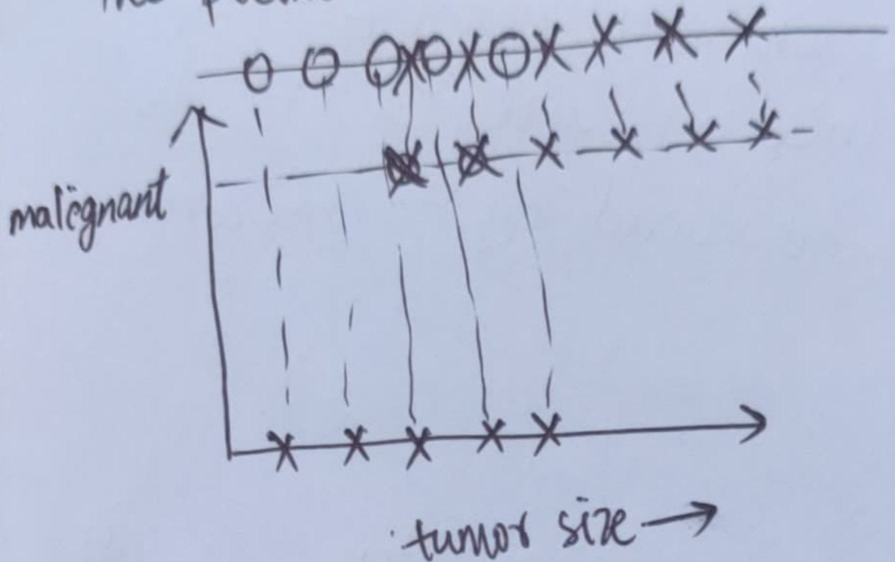




finding the mapping for continous data
↓
"REGRESSION" PROBLEM"

finding mapping for discrete data
↓
"CLASSIFICATION PROBLEM"

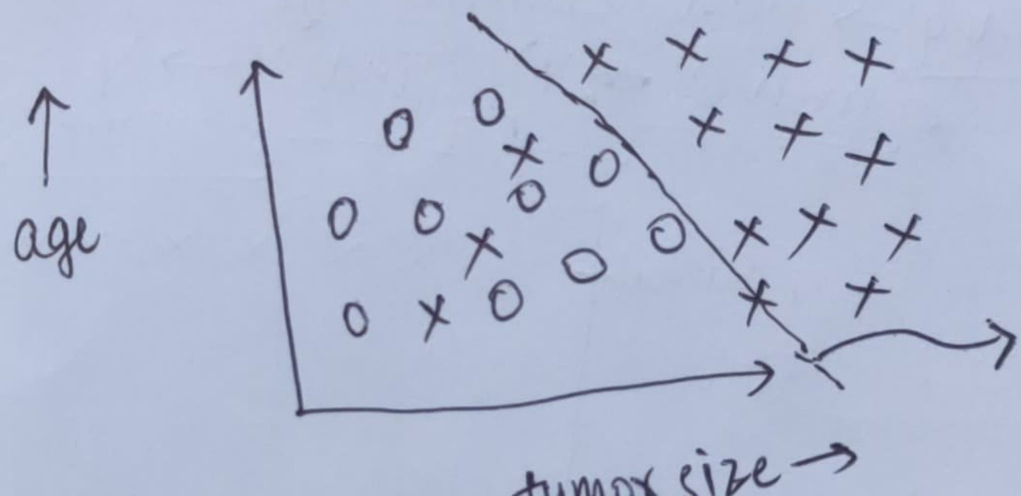
The previous breast cancer problem can be denoted as



x → means +ve

0 → means -ve.

represented in a line.



0 → denotes -ve state

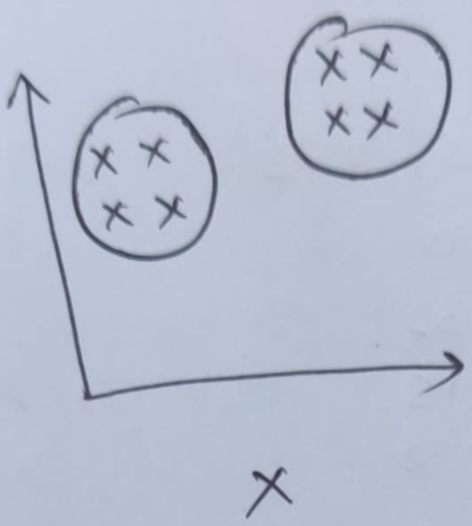
x → +ve state.

Line found using "Logistic Regression algorithm"

"support vector machine" → algorithm to represent infinite dimensional vector to represent a patient.

eg → age, tumor size, cell size .... ∞.

unsupervised learning: → using unlabelled data and finding interesting things about the data



we are given only X and no Y.

we are asked to find something interesting about this data

we find clusters using "K ~~ne~~ mean clustering" algo which is an unsupervised learning algo.

Reinforced learning → favours /rewards a certain behaviour

punishes a bad behaviour.

Linear Regression.

dataset →

size (ft^2)	prices (1000\$)
2104	400
1416	232
1534	315
852	178
⋮	⋮

data plot →

[we have to fit these on a st. line]

"supervised learning" ⇒

$\boxed{\text{TRAINING SET}}$

↓

$\boxed{\text{learning algo}}$.

↓

$\boxed{h}$

"hypothesis" creates a function to predict values

To design a learning algo, we need to know how to represent h?

↓

for linear regression

$$h(x) = \theta_0 + \theta_1 x. \text{ (linear)}.$$

$$h(x) = \sum_{j=0}^{\hat{}} \theta_j x_j \quad \text{(linear)} \quad \text{here } x_0 = 1$$

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix} \qquad X = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix}$$

$\theta \rightarrow$ "parameters"

m = # training samples (# rows in table of houses)

X = "inputs"/features

y = "output"/ target variable

$(x, y) \longrightarrow$ training example.

$(x^{(i)}, y^{(i)}) \rightarrow$ ith training example [not exponent, it is index]

as subscript is taken to denote diff. inputs types

~~for given table~~

we have to choose θ s.t. $h(x) \approx y$ for training examples

$h_\theta(x) = h(x)$.

$\hookrightarrow$ depends on θ too.

minimize by choosing θ s.t. $\frac{1}{2} \times \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2 = J(\theta)$

half because later derivative.

min. $J(\theta)$.

Gradient descent to minm. $J(\theta)$

learning rate

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta). \quad (j = 0, 1, 2 \cdots n)$$

assignment operator notation

$a := a + 1$

Theory: Imagine you are standingon a surface look 360° around you and go in the dirn. with maximum descent. keep repeating till you find minima.

$$\frac{\partial}{\partial \theta_j}(J(\theta)) = \frac{\partial}{\partial \theta_j} \frac{1}{2}(h_\theta(x) - y)^2$$

→ taking only one value instead of n values.

$$= (h_\theta(x) - y) \cdot (x_j)$$

when we sum for alle $\frac{\partial}{\partial \theta_j}(J(\theta)) = \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x^{(i)}$

$$\theta_j := \theta_j - \alpha \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})(x_j^{(i)})$$

$$(\text{for } j = 0, 1, \dots n)$$

if α is too big ↓ might overshoot minima.

if α is too small ↓ takes lots of steps.

↳ known as "batch gradient descent"

↳ as you have to do each step for your entire batch of dataset.

Stochastic gradient descent :→

Repeat {

For $i = 1$ to m {

$$\theta_j = \theta_j - \alpha (h_\theta(x^{(i)}) - y^{(i)})(x^{(i)})$$

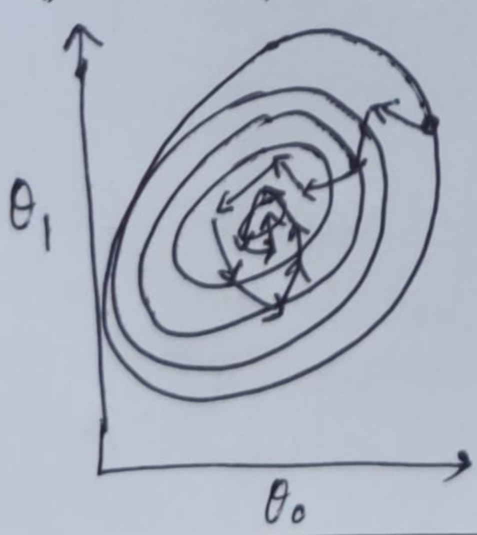
↳ do this step for each $j = 0$ to n for a given i. → batch

↳ pick a random j from 0 to n → stochastic.

}

}

graphically :→



suppose given graph is a contour graph for ~~two~~ parameters.

So, for each data we try to minimise the value of $J(\theta)^{(i)}$ from the random point we chose.

This on avg. reaches near the minima but never converges.

batch and stochastic regression are iterative. However, we can solve linear regression in exactly 1 step using "Normal equations".

some denotions are: $\rightarrow$.

$$\nabla_\theta J(\theta) = \begin{bmatrix} \frac{\partial J}{\partial \theta_0} \\ \frac{\partial J}{\partial \theta_1} \\ \vdots \\ \frac{\partial J}{\partial \theta_n} \end{bmatrix}$$

$\theta \in \mathbb{R}^{n+1}$

let A be a 2×2 matrix

$A \in \mathbb{R}^{2 \times 2}$. $f : \mathbb{R}^{2 \times 2} \mapsto \mathbb{R}$

$f(A) \in \mathbb{R}$ f is some function that maps matrix A to some number.

for ex: $\rightarrow$ $f(A) = A_{11} + A_{12}^2$

if $A = \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}$ $f(A) = 5 + 6^2$

$$\nabla_A f(A) = \begin{bmatrix} \frac{\partial f}{\partial A_{11}} & \cdots & \frac{\partial f}{\partial A_{1n}} \\ \frac{\partial f}{\partial A_{21}} & & \\ \vdots & & \\ \frac{\partial f}{\partial A_{m1}} & \cdots & \frac{\partial f}{\partial A_{mn}} \end{bmatrix}$$

derivative of $f(A)$ wrt A is a matrix of same order as A $(m \times n)$

$$\boxed{\nabla_\theta J(\theta) \overset{\text{set}}{=} \vec{0} \quad \text{and then solve for } \theta}$$

if A is a square matrix $A^{n \times n}$

"$\mathrm{tr}(A)$" = sum of diagonal entries

$$\mathrm{tr}(A) = \mathrm{tr}(A^T)$$

if $f(A) = \mathrm{tr}(AB)$ where B is some fixed matrix.

$$\nabla_A f(A) = B^T.$$

$$\mathrm{tr}(AB) = \mathrm{tr}(BA)$$

$$\mathrm{tr}(ABC) = \mathrm{tr}(CAB)$$ cyclic permutation property

* std. formula.

$$\boxed{\nabla_A \mathrm{tr}\, AA^T C = CA + C^T A}$$

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

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

let
$$X = \begin{bmatrix} (x^{(1)})^T \\ (x^{(2)})^T \\ \vdots \\ (x^{(m)})^T \end{bmatrix}$$

$$X\theta = \begin{bmatrix} (x^{(1)})^T \\ (x^{(2)})^T \\ \vdots \\ (x^{(m)})^T \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix} = \begin{bmatrix} h_\theta(x^{(1)}) \\ \vdots \\ h_\theta(x^{(m)}) \end{bmatrix}.$$

$$\vec{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix} \; ; \; X\theta - y = \begin{bmatrix} h_\theta(x^{(1)}) - y^{(1)} \\ \vdots \\ h_\theta(x^{(m)}) - y^{(m)} \end{bmatrix}$$

$$J(\theta) = \frac{1}{2} (X\theta - y)^T (X\theta - y)$$ as

$$\nabla_\theta J(\theta) = \frac{1}{2}(X\theta - y)^T(X\theta - y)$$

$$= \frac{1}{2}\nabla_\theta(\cancel{*}\theta^T X^T - y^T)(X\theta - y)$$

$$= \frac{1}{2}\nabla_\theta(\theta^T X^T X\theta - y^T X\theta - \theta^T X^T y + y^T y)$$

from formulas written.

$$= \frac{1}{2}[X^T X\theta + X^T X\theta - X^T y - X^T y] = 0.$$

$$\Rightarrow X^T X\theta - X^T y \overset{set}{=} \vec{0}$$

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

if $X^T X$ is non invertible $\Rightarrow$ inputs are linearly dependent $\rightarrow$ redundant data.