Supervised Learning

Data sets can be of two types.

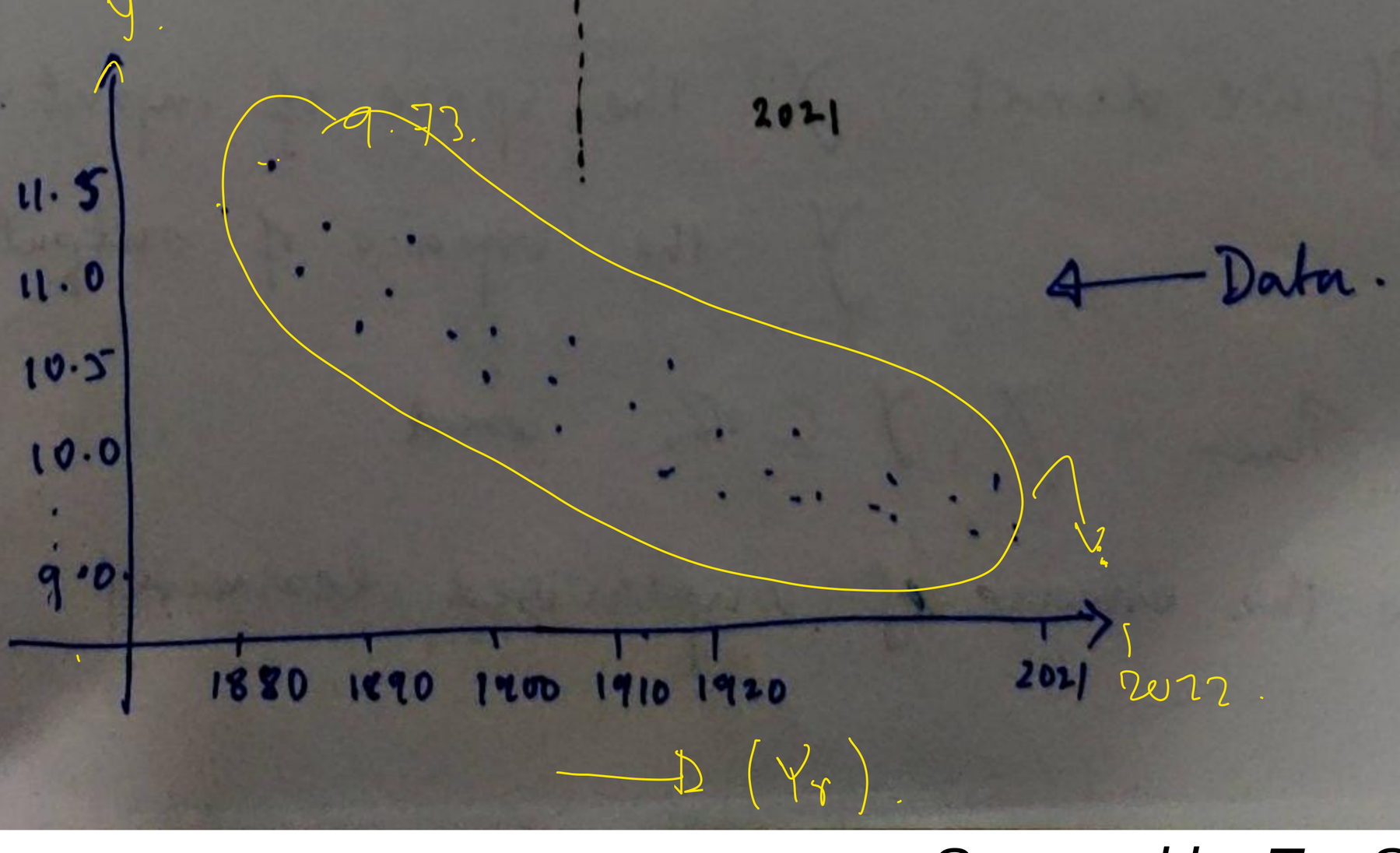
i) Regression

ii) Logistic regression / classification.

Regression: Where, target variables are continuous in nature.

Lets say we have data for winning men's 100 m times at summer Olympics. since, 1896. Two world wars interrupted in 1914, 1940

Time (s)	Years
11.3	1896
11.2	1897
11.25	1898
10.7	1899
10.85	⋮
⋮	⋮
	2021



Given this data we want to find the relation or predict the 100m winning time in the future.

Notations: x^i → input variables (features)

y^i → Output / target variable.

in the above example

$x^{(i)}$ is the year.

$y^{(i)}$ is the time (winning) in seconds.

year, time.

A pair (x^i, y^i) → training example.

$D = \{(x^i, y^i)\}_{i=1}^{i=40}$ → is the training set.

$= \{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, x^{(40)}, y^{(40)})\}$

1896, 11.5 ... 2021, 9.93

If we denote X the space of input values

Y the space of output values

Then $X, Y \subseteq \mathbb{R}$.

Now the essence of supervised learning

$D = \{(x^{(i)}, y^{(i)})\}_{i=1}^{40}$.

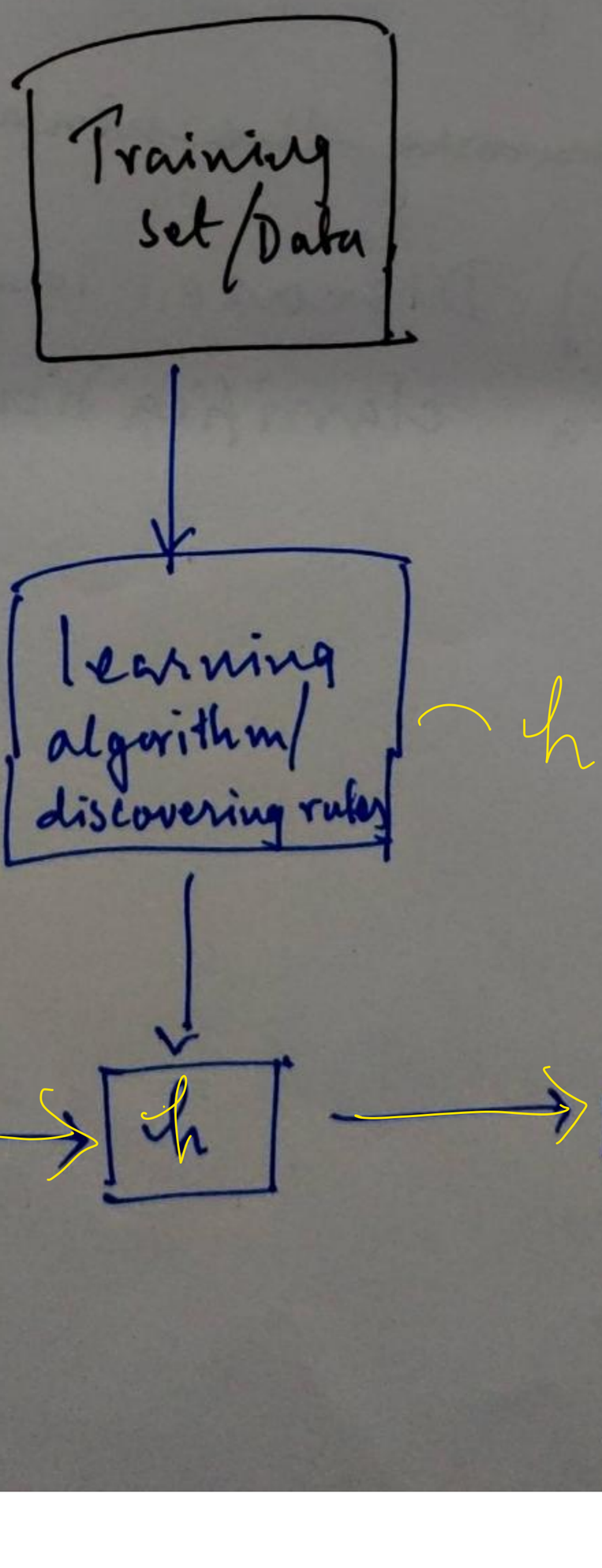
- given a training set, to learn a function $h : X \longrightarrow Y$ so that $h(x)$ is a good predictor for the corresponding value of y

h : hypothesis function.

$h_w(\text{year}) \longrightarrow t$ (in s)

- h_w is called hypothesis. function.

hence,



Classification: Logistic regression.

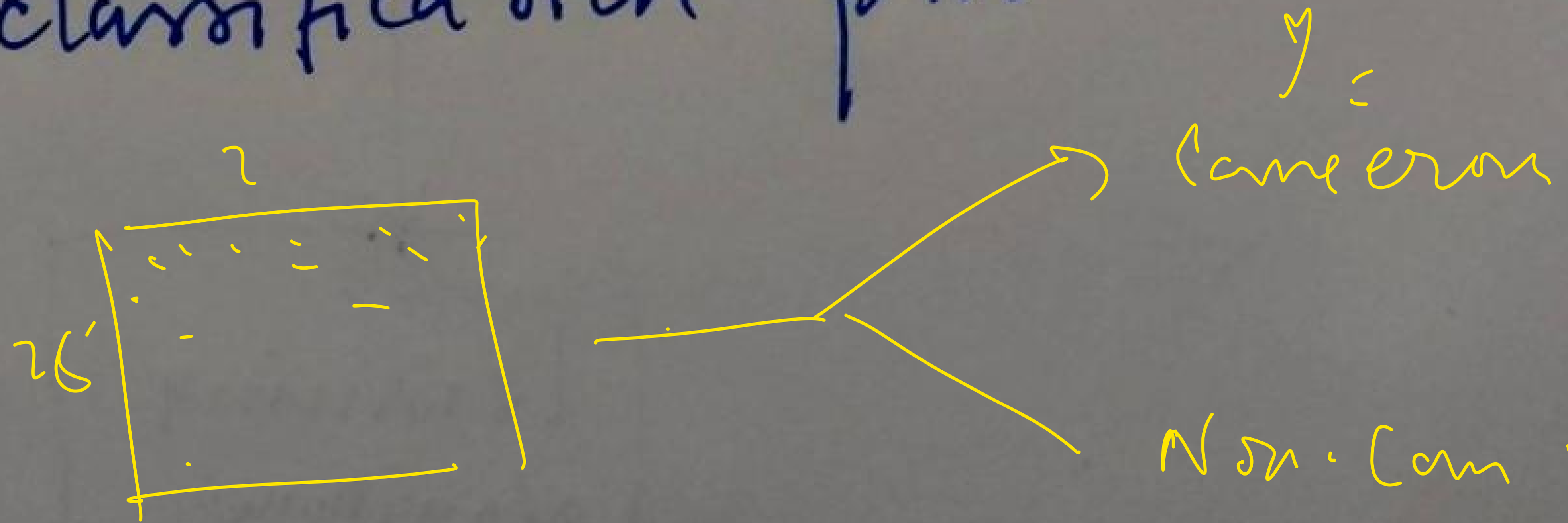
when the ~~ta~~ target variable. y can take on only small number of discrete. values.

example: i) If given the data, we want to know wheather the winner is bellow. 20 years of age or, ~~above~~ above. 20 ~~to 25, or 25~~ >

$D = \{ (x^{(1)}, 1), (x^{(1)}, 0), (x^{(3)}, 1), (x^{(4)}, 1) \dots \}$

ii) ~~Those among the winners.~~

~~Both this.~~ ii) Disease identification from. images is a classification prob.



Linear Regression

Andrew NG (IRef).

Lets consider a set of data. of property price in. Delhi.

Living area (ft^2)	# bedrooms	Price (Lakhs INR)
2104	3	400
1600	3	330
2400	3	369
1416	2	232
3000	4	540
⋮	⋮	⋮

$\underline{x} = \begin{pmatrix} x_1 : \#\text{ of b rooms.} \\ x_2 : \#\text{ area of the apt} \end{pmatrix} \in \mathbb{R}^2.$

let $\underline{x} \in \mathbb{R}^2$ (a two dimentional vector)

and $\underline{x}^{(i)}$ (i 1 to m denotes the data set)

w/ i is the. house identifier.

$\underline{x}^{(i)} = \begin{pmatrix} x_1^{(i)} \to \text{feature 1} : \text{Living area size} \\ x_2^{(i)} \to \text{feature 2} : \#\text{ of bedrooms.} \end{pmatrix}$

Now before we do supervised learning on the given data; following steps need to be clarified

i) Defining the model:

identifying the input $\underline{x}^i$ (with two attributes : living area, # of bed rooms.

Output y^i (the price in Lakhs)

ii) Modelling assumptions. $y \simeq h_w(\underline{x}^i)$

This is an important step of the process. Whether the relationship bet output and input is linear OR higher order has to be decided. Based on this, parameters are identified and hypothesis is made.

if linear, $\Rightarrow$ say $h_w(x) = w_0 + w_1 x_1 + w_2 x_2$.

$\therefore w_0 + w_1 x_1 + w_2 x_1^2 + w_3 x_2 + w_4 x_2^2 + w_5 x_1 x_2$

w_i, w_i are the parameters. (or weights)

bias. area of apt # bed

x_1 x_2 x_3

$h_w(x) : X \to Y$

~ This is a linear mapping from X to Y space.

~ To give it a structure. Lets say $x_0 = 1$.

the

$$h_w(\underline{x}) = w_0 x_0 + w_1 x_1 + w_2 x_2$$

$$= \underline{w}^T \underline{x} \quad \leftarrow \text{vectorization}$$

Both, $\underline{w} = \begin{pmatrix} w_0 \\ w_1 \\ w_2 \end{pmatrix}$, $\underline{x} = \begin{pmatrix} x_0 \\ x_1 \\ x_2 \end{pmatrix}$ are vector..

- x_0 is the intercept term.

---

iii) Defining Measure of a "goodness" model.

Given a data set, the problem now boils down to identifying the model parameters. $\underline{w}$: How do we define this "Goodness"

One assumption is to make the predicted output from $h(\underline{x})$ close to y (actual output) for the known data.

$h_w(x_1^{(i)}, x_2^{(i)}) - y^{(i)} = \Delta h$

— And here we define a COST function or ~~loss~~ loss function. between predicted and actual data y^i

$$\sum_{i=1}^{m} |h_w - y^i|^2 \quad \ldots$$

$$J(\theta_w) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_w(x^i) - y^i \right)^2 \quad \text{(A)}$$

Cost function | loss of error func. | normalize. | Least square minimization.

of m: # of data points.

$\underline{x}$: # is the input feature vector. $\in \mathbb{R}^2$.

$\underline{y}$: output (price of the house) $\in \mathbb{R}$

h_w: Hypothesis function. (we have assumed it to be linear)

θ_1, w_0, w_1 — we want to model.

The cost function $J(w)$ is the least square error. in this case. But, there may be some other defination of cost function for defining the measure of goodness of the model.

— $\underline{w} = \begin{pmatrix} w_0 \\ w_1 \\ w_2 \end{pmatrix}$.

Optimization (minimization) Algorithm for the cost function

We need to find out the weights (w) in defining the model.

Generally The weights are found so that they minimize the cost function. An iterative algorithm is adopted for the "optimization" prob. as defined bellow.

$$\underline{w}^* = \underset{\underline{w}}{\operatorname{argmin}}\ J(\underline{w}) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_w(x^{(i)}) - y^{(i)}\right)^2$$

for $J : \mathbb{R}^{n+1} \to \mathbb{R}$, $\underline{w} \in \mathbb{R}^{n+1}$

$\begin{pmatrix} w_0 \\ w_1 \\ w_2 \end{pmatrix} \in \mathbb{R}^3$ — $J(w_0, w_1, w_2)$

Unconstrained optimization

Different algorithm exist:

i) Gradient descent

ii) Conjugate gradient descent

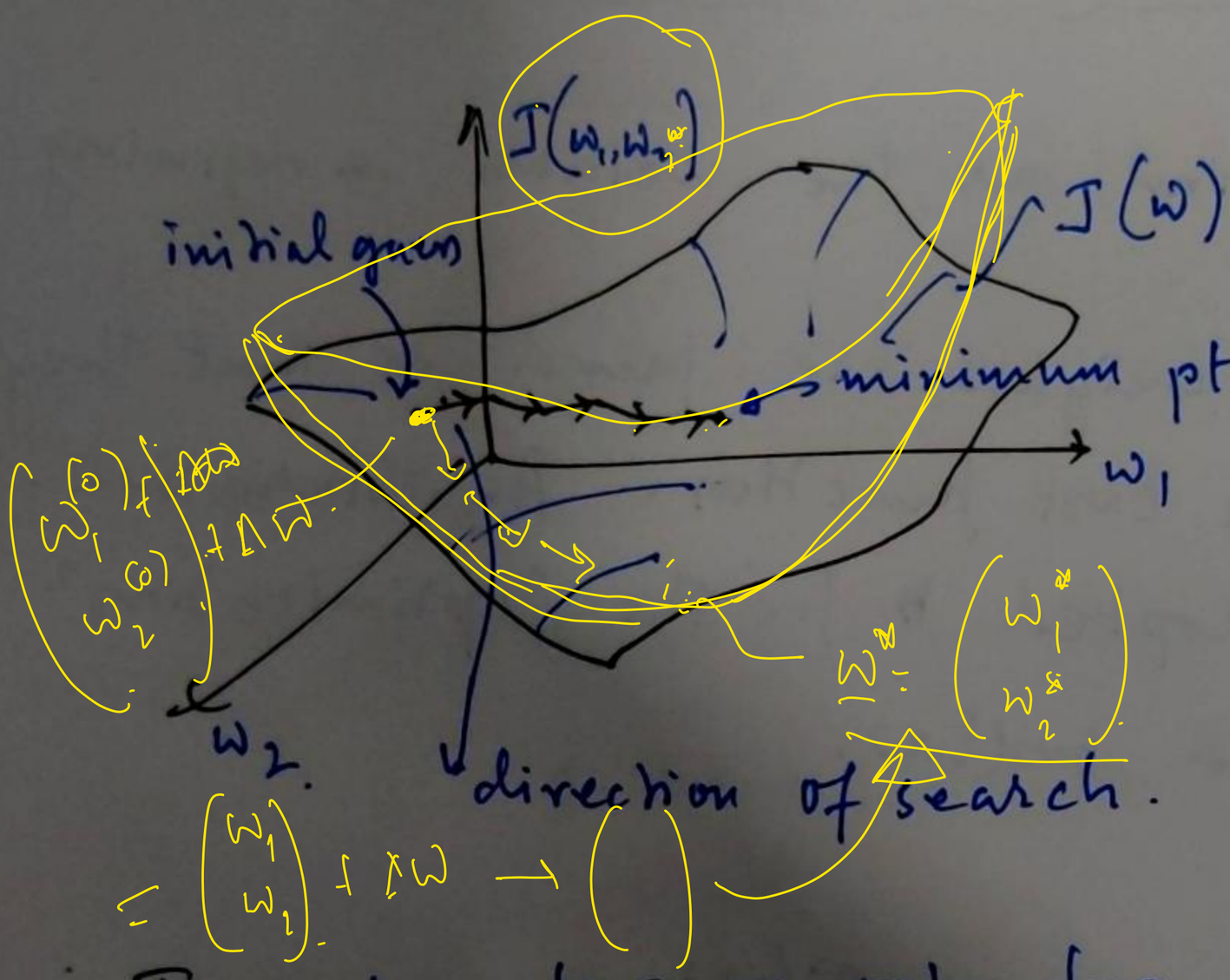
iii) Newton's method.

iv) Quasi-Newton method.

v) Gauss-Jordon method

Least square minimization (LMS) is a convex optimization.

Gradient descent



Inorder to search for.

Gradient descent is an iterative procedure.

In order to search for the optimum (minimum) we need a search direction.

"The "gradient" at a point of a real valued differentiable function gives the direction of max^{m} increase.

Hence, inorder to find the max^{m} decrease we need to search along -ive gradient dir.

$$\underline{d} = -\nabla f = -\begin{pmatrix} \frac{\partial f}{\partial w_1} \\ \frac{\partial f}{\partial w_2} \end{pmatrix}$$

$$\underline{w}^k = \begin{pmatrix} w_1^k \\ w_i^k \end{pmatrix} - \alpha \begin{pmatrix} \frac{\partial f}{\partial w_1} \\ \frac{\partial f}{\partial w_i} \end{pmatrix}_{w_1^k, w_i^k} = \underline{w}^{k+1}$$

Any iterative algorithm needs to have the following features.

$\underline{w}^* = \arg\min J(\underline{w})$

i) An initial guess ($\underline{w}^0$)

ii) An update rule. This depend on the search direction.

$\underline{w}^{k+1} = \underline{w}^k + \alpha \underline{d}^k$ → search direction

In. • gradient descent $\underline{d}^k = -\nabla f(\underline{x}^k)$

Widrow-Hoff L.R. iii) A step size or learning rate. α. There are techinques to find this learning rate. using line search algorithm.

iv) A stopping criteria.

The iterative algorithm needs to stop.

The magnitude of the update is proportional to the error. If the error is within a certain tolerance we stop.

e.g. $\text{tol} = \frac{J(\underline{w}^{k+1}) - J(\underline{w}^k)}{\max(1, J(\underline{w}^k))} \leq \epsilon$ ~ 10^{-5} $J(\underline{w}^{k+1}) \simeq J(\underline{w}^k)$

$$J(w) = \frac{\partial}{\partial w_j} \frac{1}{2m} \sum_{i=1}^{m} \left(\sum_{j=1}^{n} w_j x_j^{(i)} - y^i \right)^2$$

$\underline{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$ $n = 2$

[division by m normalizes it.]

$$\Rightarrow \frac{\partial J}{\partial w_j} = \frac{1}{2} \frac{1}{m} 2 \sum_{i=1}^{m} \left(\sum_{j=0}^{n} w_j x_j^{(i)} - y^i \right) x_j^{(i)}$$

$x^{(i)}$ ← data #

i : # of data pts.

j : # of features.

$\underline{w} = \begin{pmatrix} w_0 \to j=0 \\ w_1 \to j=1 \\ w_2 \to j=2 \end{pmatrix}$

Now the update rule.

$$w_j^{k+1} = w_j^k - \alpha \frac{\partial J}{\partial w_j} \qquad j = 0, 1, 2.$$

$$= w_j^k - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_w(\underline{x}^i) - y^i \right) x_j^i$$

(A)

repeat for k until convergence.

For $\forall$ $j = 0$ to n

$(\underline{x}^i, y^i)$ is a data pt.

$\longrightarrow j$

$\downarrow i$

$x_0^{(1)}$	$x_1^{(1)}$	$x_2^{(1)}$	....	1st data
$x_0^{(2)}$	$x_2^{(2)}$	$x_3^{(2)}$	....	2nd data
$x_0^{(3)}$	$x_2^{(3)}$	$x_3^{(3)}$	...	
		⋮		
$x_0^{(m)}$	$x_1^{(m)}$	$x_3^{(m)}$	... $x_n^{(m)}$	m th data

$m, i = 1$

$w^{k+1} = w^k - \alpha \nabla J$ $\nabla J = \begin{pmatrix} \frac{\partial J}{\partial w_0} \\ \frac{\partial J}{\partial w_1} \\ \frac{\partial J}{\partial w_n} \end{pmatrix}$

Batch Gradient descent Δ.

Stochastic Gradient or (incremental) gradient descent

Batch Gradient descent:

i) The update rule in eqn (A) is defined over the total "training set" i.e, $\{(x^i, y^i)\}_{i=1}^{m}$

ii) This method looks at every example or data pt in the training set at every step k

Note: The gradient descent can be susceptible to local minimum in general, the optimization problem we have posed here for linear regression has only one global minima, and no other local optima; thus gradient descent always converges. (assuming the learning rate α is not too large) to the global minimum.

Least square "error" cost function J is CONVEX function. The property of a convex function is all its local minimum is global minimum. (unique)

Stochastic gradient descent

The above results we obtained with 'batch gradient descent' which involve procesing the entire training set in one go, can be computationally costly for large data set.

→ If the dataset is sufficiently large, it may be worthwhile to use. "sequential learning" also known as "on-line learning" algorithms.

In this data pts are considered one-at-a time. model parameters are updated after each such presentation.

→ Sequential learning is also appropriate for real time applications in which the data observations are arriving in a continuous stream and predictions must be made before all the data points are seen.

→. Stochastic gradient descent also known as sequential gradient descent.

For LMS algo

$$J = \frac{1}{2}\left(y^{(1)} - h_w(x^{(1)})\right)^2 + \frac{1}{2}\left(y^{(2)} - h_w(x^{(2)})\right) + \cdots$$

$$= J_1 + J_2 + \cdots J_m$$

$$\Rightarrow \nabla J = \nabla J_1 + \nabla J_2 + \cdots \nabla J_m$$

The algorithm is.

Loops.

loops $i = 1$ to m

$$\underline{w}^{k+1} = \underline{w}^k - \alpha \nabla J_{j=i} \quad \text{w/} \quad \nabla J_{jm} = (y^m - h_w(\underline{x}^m))x_j^m$$

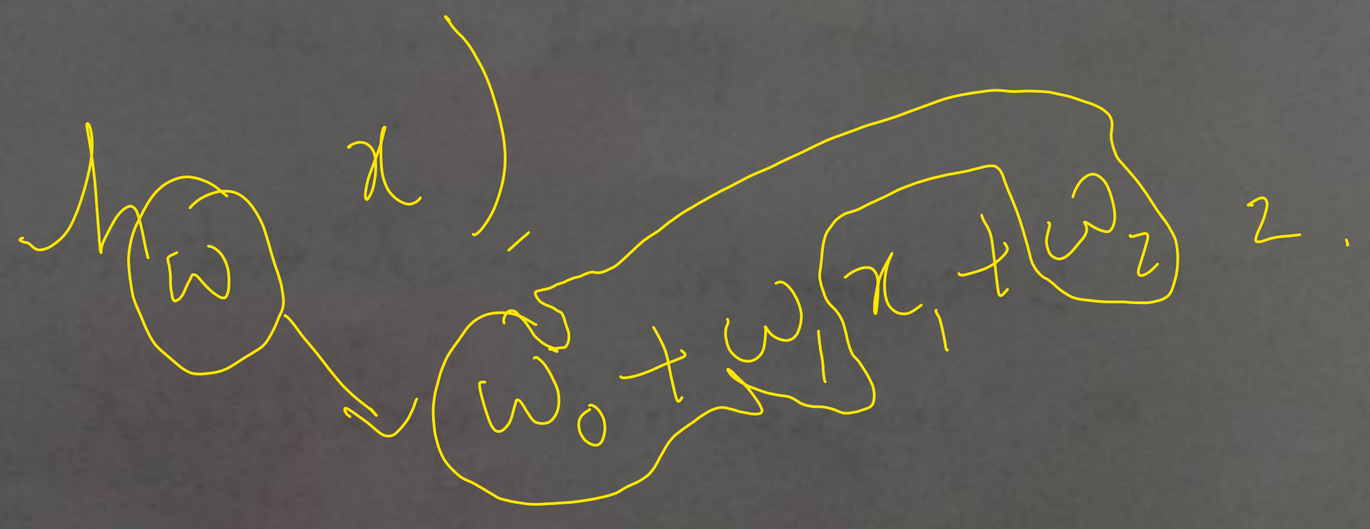
vector.

$$= \underline{w}^k - \alpha\left(x^{(i)} - \underbrace{\underline{w}^{kT}\underline{x}_{j=i}}_{h_w(\underline{x}^i)}\right)\underline{x}_j^{(i)}$$

end

end.

The algorithm is written in vector form.



Normal Equations.

The property of a multivariate function like $J(w_0, w_1, w_2, \cdots) = J(\underline{w})$. to be minimum is. $J: \mathbb{R}^n \longrightarrow \mathbb{R}$.

$f(x): \mathbb{R} \to \mathbb{R}$.

$\frac{df}{dx} = 0$, $\frac{d^2f}{dx^2} > 0$ i.e. x_{min}

$f'(x^*) > 0$.

$\nabla f = 0$; $H(x^*) > 0$.

$D_w J = 0$ & $H_J(\underline{w}) \gg 0$.

gradient

$$D_w J = \begin{pmatrix} \frac{\partial J}{\partial w_1} \\ \frac{\partial J}{\partial w_2} \\ \vdots \\ \frac{\partial J}{\partial w_n} \end{pmatrix} = 0.$$

$\underline{a}^T H_J \underline{a} > 0$.

↳ Hessian.

$$\underline{H} = \begin{bmatrix} \frac{\partial^2 J}{\partial w_1^2} & \frac{\partial^2 J}{\partial w_1 \partial w_2} & \frac{\partial^2 J}{\partial w_1 \partial w_3} \\ \frac{\partial^2 J}{\partial w_2 \partial w_1} & \frac{\partial^2 J}{\partial w_2^2} & \cdots \\ \vdots & \cdots & \end{bmatrix} = \left(\frac{\partial^2 J}{\partial w_i \partial w_j}\right)$$

$D_w J$: always a vector. $= (D_w J)^T$.

↑ derivative of J.

The Hessian of the function $J(\underline{w})$ is.

$$H_{ij} = \left[\frac{\partial^2 f}{\partial w_i \partial w_j}\right].$$

$$= \begin{bmatrix} \frac{\partial^2 J}{\partial w_1^2} & \frac{\partial^2 J}{\partial w_1 \partial w_2} & \cdots & \frac{\partial^2 J}{\partial w_1 \partial w_n} \\ \frac{\partial^2 J}{\partial w_2 \partial w_1} & \frac{\partial^2 J}{\partial w_2^2} & \cdots & \frac{\partial^2 J}{\partial w_n \partial w_2} \\ & & & \vdots \end{bmatrix}$$

For the current prob in hand the training set data can be arranged in matrix form as

$n = 2$

$$\underset{\sim}{y} = \begin{pmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{pmatrix} \quad \underset{=}{X} = \begin{bmatrix} x_0^{(1)} & x_1^{(1)} & \cdots & x_n^{(1)} \\ x_0^{(2)} & x_1^{(2)} & \cdots & x_n^{(2)} \\ & & & \\ x_0^{(m)} & x_1^{(m)} & \cdots & x_n^{(m)} \end{bmatrix} \quad \underset{\sim}{w} = \begin{pmatrix} w_0 \\ w_1 \\ w_2 \\ \vdots \\ w_n \end{pmatrix}$$

hence. $J(w)$ can be written as.

$$\begin{pmatrix} \Delta J_1 \\ \Delta J_2 \\ \vdots \\ \Delta J_m \end{pmatrix} = \begin{bmatrix} x_0^{(1)} & x_1^{(1)} & \cdots & x_n^{(1)} \\ & & & \\ x_0^{(m)} & x_1^{(m)} & \cdots & x_n^{(m)} \end{bmatrix} \begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_n \end{pmatrix} - \begin{pmatrix} y^{(1)} \\ \\ \\ y^{(m)} \end{pmatrix} =$$

now $\Delta J(w) = \frac{1}{2m} \sum_{i=1}^{m} \Delta J_i^2$.

$$= \frac{1}{2m} (\underset{=}{X}\underset{\sim}{w} - \underset{\sim}{y})^T (\underset{=}{X}\underset{\sim}{w} - \underset{\sim}{y})$$

$$= \frac{1}{2m} (\underset{\sim}{w}^T \underset{=}{X}^T \underset{=}{X} w - \underset{\sim}{w}^T \underset{=}{X}^T \underset{\sim}{y} - \underset{\sim}{y}^T \underset{=}{X} \underset{\sim}{w} + \underset{\sim}{y}^T \underset{\sim}{y})$$

$$\nabla_w J = \underset{=}{X}^T \underset{=}{X} \underset{\sim}{w} - \underset{=}{X}^T \underset{\sim}{y} + 0$$

It can be shown that

$$\nabla_{\underline{w}} J = \underline{\underline{X}}^T \underline{\underline{X}}\, \underline{w} - \underline{\underline{X}}^T \underline{y} = 0 \quad \text{---(1)}$$

Now, the condition for optimum $\underline{w}^*$ is $\nabla_{\underline{w}} J = 0$.

$$\underline{w}^* = \arg\min J(\underline{w}).$$

From equⁿ (1) we get.

$$\underline{\underline{X}}^T \underline{\underline{X}}\, \underline{w} - \underline{\underline{X}}^T \underline{y} = 0.$$

$$\Rightarrow \boxed{\underline{w}^* = \left(\underline{\underline{X}}^T \underline{\underline{X}}\right)^{-1} \underline{\underline{X}}^T \underline{y}}$$

↳ Closed form equⁿ for $\underline{w}^*$

$\underline{\underline{X}}^T \underline{\underline{X}}$: 2×2

$\underline{w}^* \leftarrow \left(w^{k+1} = w^k - \alpha \nabla f(w^k)\right)$

Probabilistic viewpoint

$$P(D|w) = \prod_{i=1}^{m} p\left(y^{(i)} \mid x^{(i)}; w\right)$$

Note: In machine learning literature, the -ive log of the likelihood function is called an error function. Because, the -ive logarithm is a monotonically decreasing function, maximizing the likelihood is equivalent to decreasing the error.

Why least square error function is a good. 'estimator' of a model's "goodness"

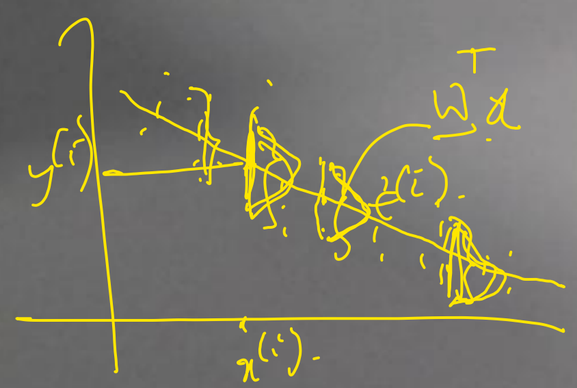
Lets assume target variables and inputs are related through an equ^{n}.

$$y^{(i)} = \underset{\sim}{w}^T \underline{x}^{(i)} + \epsilon^{(i)}$$

$y^{(i)}$: scalar

$\underline{x}^{(i)}$: vector (multiple features)

$\epsilon^{(i)}$: error in estimate.



what does $\epsilon^{(i)}$ measure?

i) $\epsilon^{(i)}$ is a random variable, which captures either unmodeled effects / or features / attributes in the data. or random noise.

ii) $\epsilon^{(i)}$ also follows a zero mean Gaussian distribution. i.e with.

$$\epsilon^{(i)} \sim \mathcal{N}(y^i \mid 0, \sigma^2)$$

mean, S.D.

$\sigma^2 \simeq \frac{1}{\beta} : \beta^{-1}$

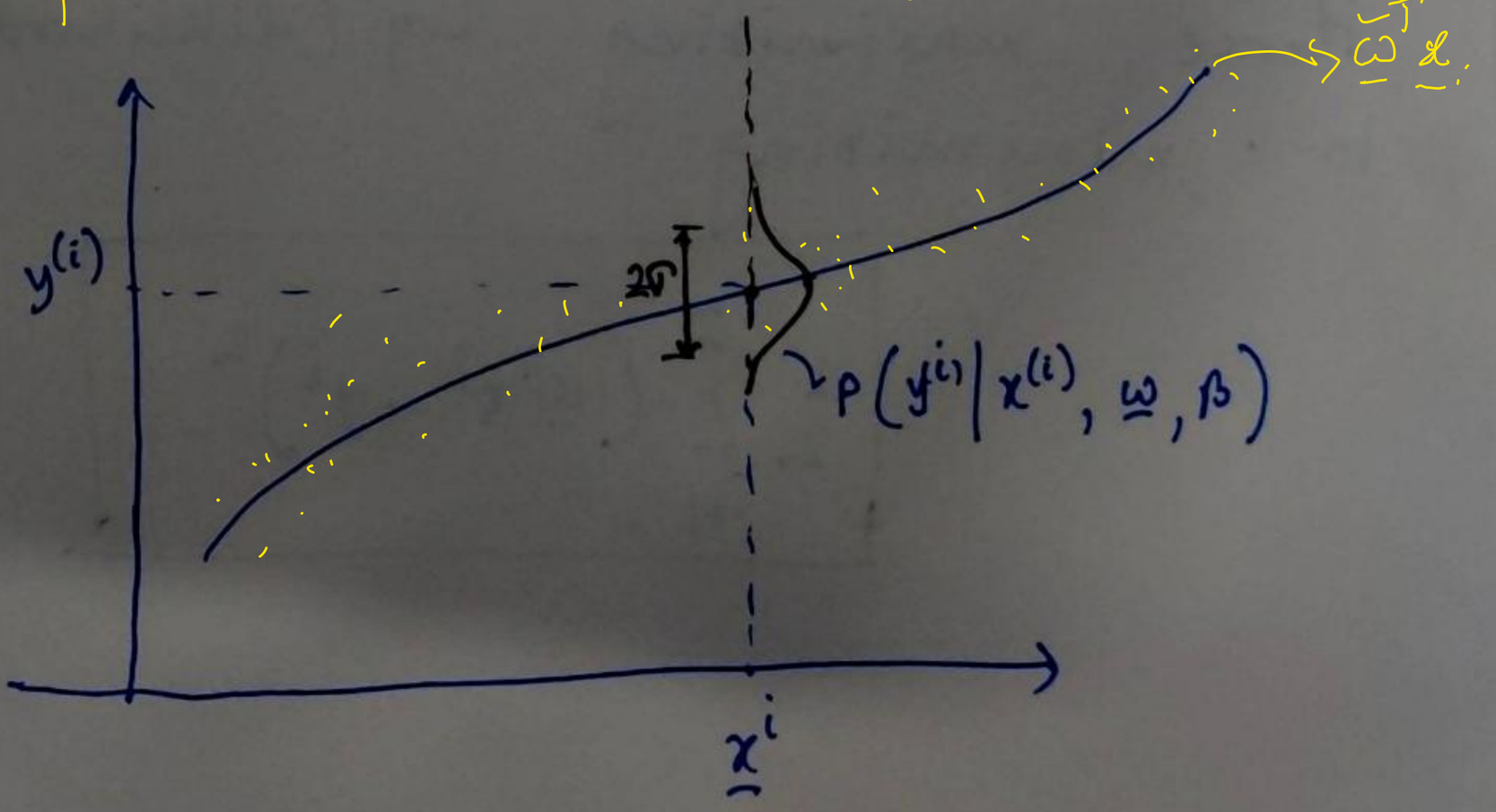
precision β (inverse of variance)

$$\epsilon^{(i)} \sim \mathcal{N}(0, \beta^{-1}) \quad \text{or} \quad p(\epsilon^{(i)}) = \frac{1}{\sqrt{2\pi}\,\sigma} \exp\left(-\frac{(\epsilon^{(i)})^2}{2\sigma^2}\right)$$

Now, the probability distribution of the target variable $y^{(i)}$ is given by.

$$p\left(y^{(i)} \middle| x^{(i)}, \underline{w}, \beta\right) = \mathcal{N}\left(y^{(i)} \middle| \underline{w}^T \underline{x}^{(i)}, \beta^{-1}\right)$$

μ , σ^2 .



We now use the training dataset $\{(\underline{x}^i, y^i)\}_{i=1}^m$ to determine the values of the unknown parameters $\underline{w}$ and β by "maximum likelihood"

$P(D | \underline{X}, \underline{w}, \beta)$.

As we assume every data pt is independent of each other, i.e ϵ^i, y^i & $\underline{x}^i$ are independent random variables. So, likelihood function is defined as

$$\underline{y} = \begin{pmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{pmatrix} = D \qquad p(\underline{y} | \underline{X}, \underline{w}, \beta) = \prod_{i=1}^{m} \mathcal{N}\left(y^{(i)} \middle| \underline{w}^T \underline{x}^i, \beta^{-1}\right)$$

i.i.d. $e^{-\frac{z^2}{2\sigma^2}}$

Likelihood $= \prod_{i=1}^{n} p\left(y^i \middle| \underline{x}^i; \underline{w}\right)$, is the given.

probabilistic model relating y^i with $\underline{x}^i$'s.

Now what is a reasonable way of choosing our best guess of the parameters $\underline{w}$?

$$-\log\{p\left(\underline{y} \middle| \underline{X}, \underline{w}, \beta\right) = -\frac{\beta}{2}\sum_{i=1}^{m}\left(\underline{w}^T\underline{x}^i - y^i\right)^2 + \frac{m}{2}\ln\beta - \frac{m}{2}\ln(2\pi)$$

Hence, maximizing log (liklihood) is equivalent to minimizing

$$\frac{1}{2m}\sum_{i=1}^{m}\left(\underline{w}^T\underline{x}^i - y^i\right)^2.$$

normalized by a constant.

least sq error term

$-f(x)$

$= f(x)$

Note: consts β, m & 2π does not change the optimum pt. of the objective function.