Soundarya S
IPhD-20

THEORY-1

* Topics Covered:

1. Supervised Vs Unsupervised learning
2. Linear regression : Model & cost function
3. Gradient descent Algorithm
4. Implementing Squared Error Gradient Descent for Linear regression.
5. Classification using logistic regression
6. Cost function for log. regression
7. Implementation with Grad descent

Notes posted in the classroom are the primary material. This is just a supplement to understand it better. Approaches in both may differ

SOURCE:

- Coursera: Supervised ML by Andrew NG.

I Supervised: → Algorithms that learn to map input x to o/p y.
→ give learning Algo. examples to learn from.
→ learns by seeing correct pairs of i/p x & desired o/p label y and gives a reasonably accurate prediction or guess of o/p.

Instructor: Andrew Ng

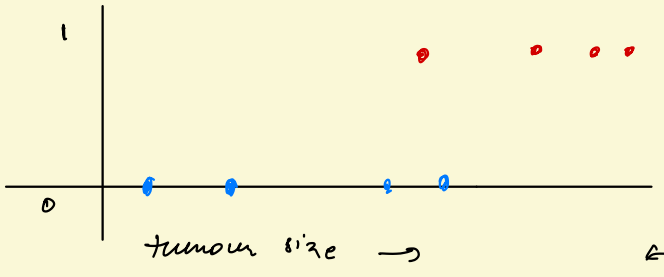
I/p: email, English, ad, user info

o/p: Spam (0/1)? Spanish. Click? (0/1)

Type of Sup. learning:

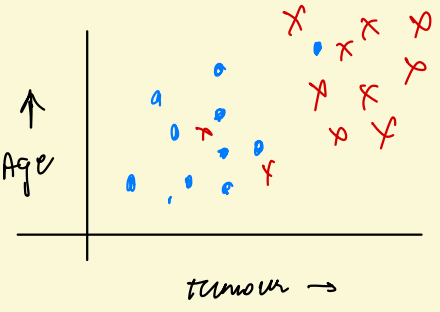
① Regression: given set of data, predict o/p for a new data
aim: how to get an algo. to systematically choose the most appropriate line / curve / other thing to fit this data
Eg Given the house area v/s the cost predict the cost for a given area.

② Classification:



a Type 1 cancer
a Type 2 cancer

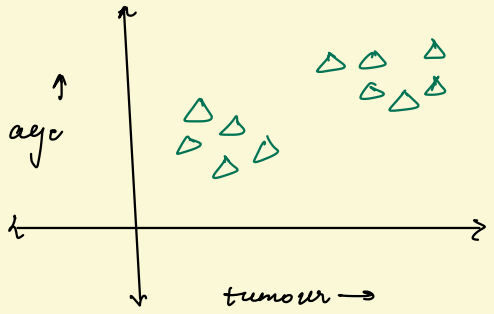
← One i/p one o/p



← The type of cancer here depends upon 2 parameters: tumour size and age.

II Unsupervised learning:
↓
We are given data that is not associated with any o/p label y.
It's unsupervised because we r not trying to supervise the algorithm to give some right answer for every i/p.

Supervised learn → data labeled with right answers



① Find something interesting in unlabeled data like clusters or something

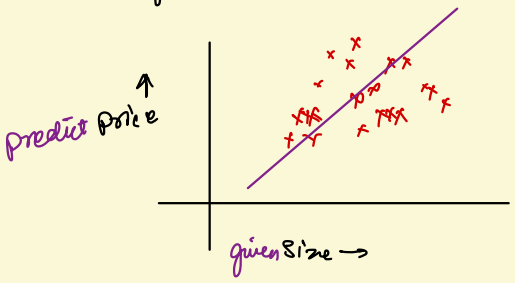
Eg: ⊙ Google news: groups similar stories together

② Grouping customers.

② Anomaly detection finds unusual data points

• Linear Regression:

Predict the price of house based on size.



Supervised learning

① reg. model predicts number.

② ∞ # of possible o/ps

Supervised learning: Data has "right" answers

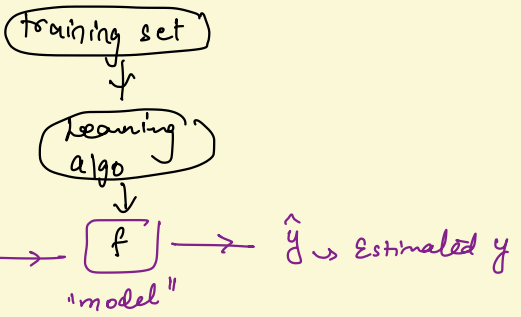
Learning

① Classification model. predicts categories.

② small no. of possible o/ps

m = # of training examples.

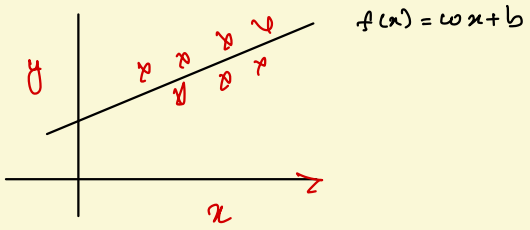
(x, y) = single training example.



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

choose w & b to predict b.

How to represent f?

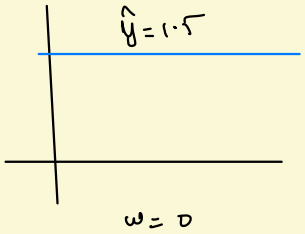


① Linear regression with one variable size / feature x

"Univariate" linear regression

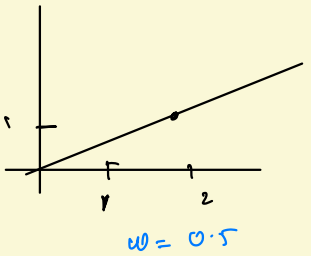
★ Cost function:

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



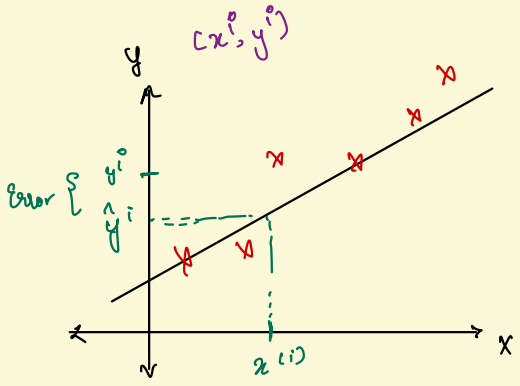
$w = 0$

$b = 1.5$



$w = 0.5$

$b = 0$



$\hat{y}^{(i)} = f(x^{(i)}) = wx^{(i)} + b$

Find w, b such that:

$\hat{y}^{(i)}$ is close to $y^{(i)}$ $\forall$ $(x^{(i)}, y^{(i)})$

Cost function: $(\hat{y} - y)^2 \rightarrow$ error $\quad m$ = # of training examples

$\sum_{i=1}^{m} (\hat{y}^{(i)} - y^{(i)})^2 \rightarrow$ gets larger & larger as m increases

(error)

we take the average hence.

★ Cost function: "Squared Error Cost function"

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

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

Goal: $\min_{w,b} J(w,b)$

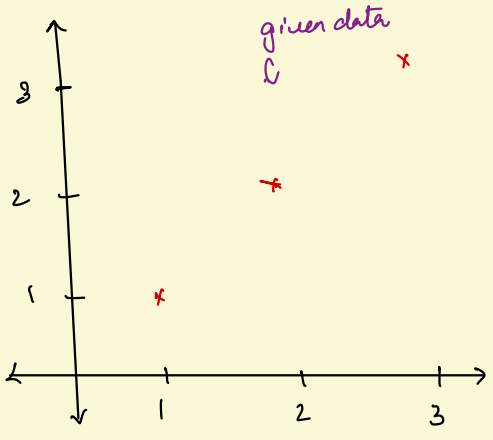
↳ How are the cost functions and the model related? let $b = 0$

$f_w(x) = wx$

> depends upon i/p x

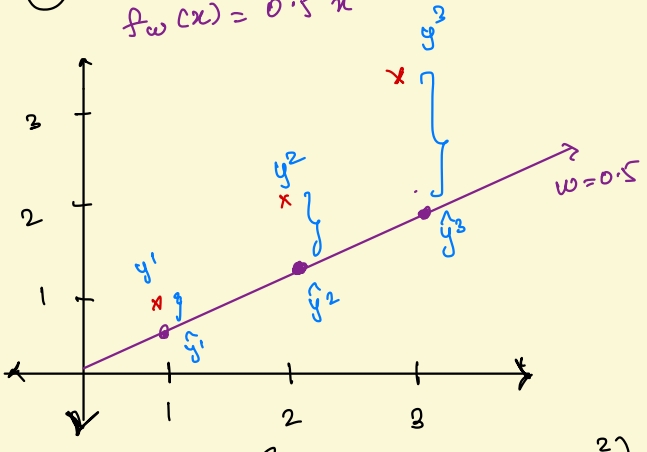
$$J(w) = \frac{1}{2m} \sum_{i=1}^{m} (f_w(x^{(i)}) - y^{(i)})^2$$

> Depends upon w parameter



(1) $w = 0.5$

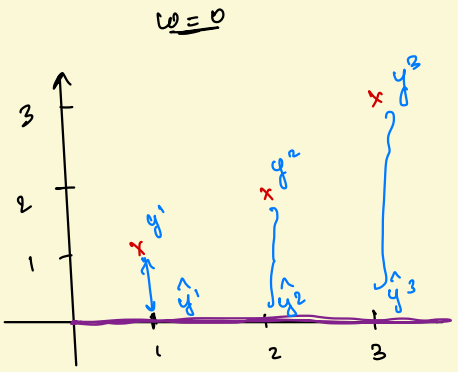
$f_w(x) = 0.5x$



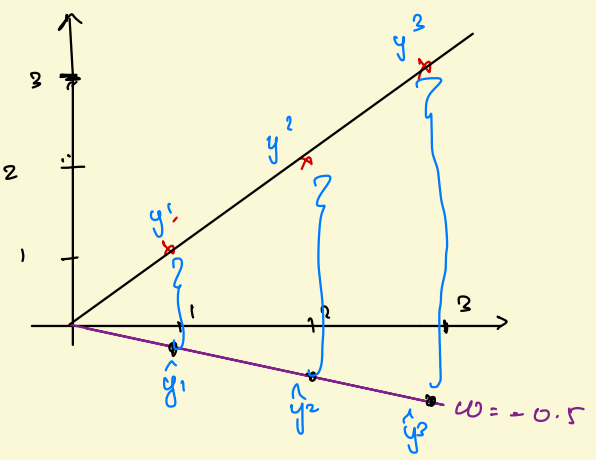
$m = 3$

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

≈ 0.58



$$J(w) = \frac{1}{2 \cdot 3}\left[1^2 + 2^2 + 3^2\right) = 2.3$$



Eg $w = 1 \Rightarrow \hat{y} = w\hat{x} = \hat{x}$

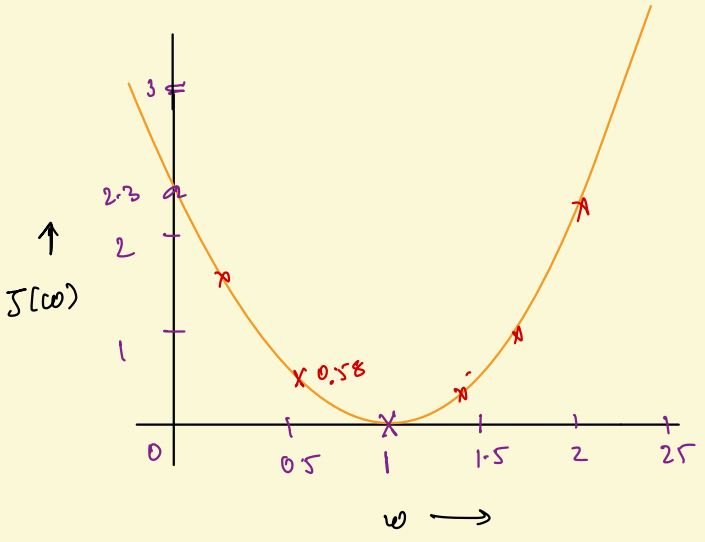
$$J(w) = \frac{1}{2m}\sum_{i=1}^{m}(wx^i - y^i)^2 = \frac{1}{2m}\sum_{i=1}^{m}(x^i - y^i)^2 = 0$$

$w = 1 \Rightarrow J(w) = 0$

Eg $w = 0.5 \quad \hat{y} = 0.5\hat{x}$

$$J(w) = \frac{1}{2m}\sum_{i=1}^{m}(0.5x^i - y^i)^2$$

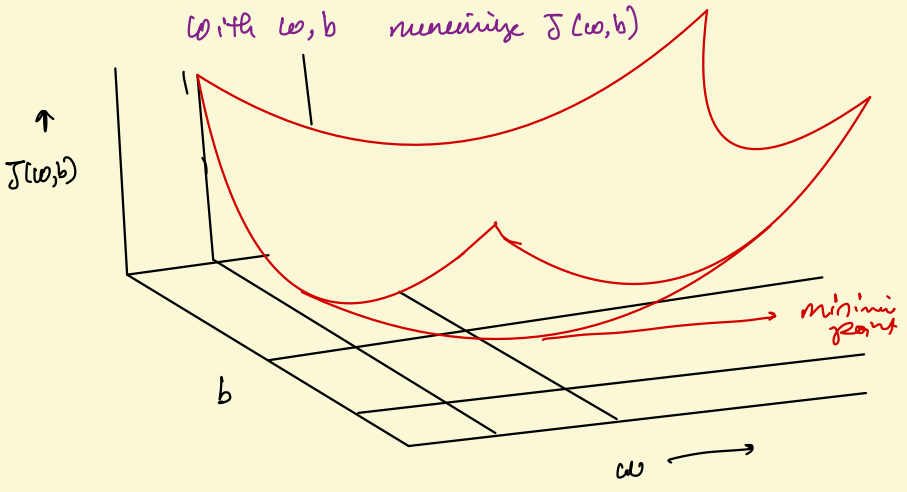
∴ We want to minimize $J(w)$ finding w



General Case: Model: $f_{w,b}(x) = wx + b$

$J(w,b)$

Need an algorithm for finding parameters w & b that give the best fit for $J(w,b)$



Gradient Descent.

Have some function $J(w,b)$

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

$\min_{w_1, w_2, \ldots, w_n, b} J(w_1, w_2, \ldots, w_n, b)$

Cost function for linear regression : squared error.

cost func for neural network : may be differ

Cost fun: error function.

GRADIENT DESCENT ALGORITHM:

Repeat until convergence: → until u reach a local minimum

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

$$b = b - \alpha \frac{\partial}{\partial b} J(w,b)$$ → old w only.

α = learning rate

Simultaneously update w & b

Correct: Simultaneous update

```
tmp_w = w - α ∂/∂w J(w,b)
tmp_b = b - α ∂/∂b J(w,b)
w = tmp_w
b = tmp_b
```

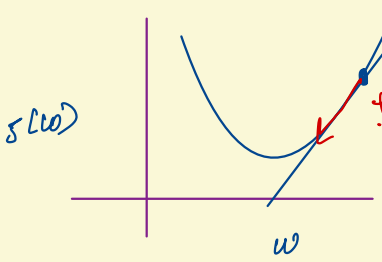
Incorrect: Simultaneous Update

```
tmp_w = w - α ∂/∂w J(w,b)
w = tmp_w
tmp_b = b - α ∂/∂b J(w,b)   ← new w
b = tmp_b
```

✗ wrong

α = learning rate how big a step u take in updating w & b

Intuition when b=0 :



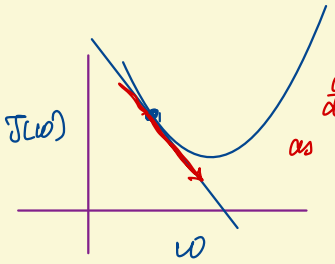
$\frac{d}{dw} J(w)$ +ive ∴ as w↓ J(w)↓

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

$\frac{d}{dw} J(w) \geq 0$ $\quad \alpha \geq 0$

$w = w - \alpha$ (positive number)

w↓



$\frac{d}{dw} J(w)$ as w↑ J(w)↓

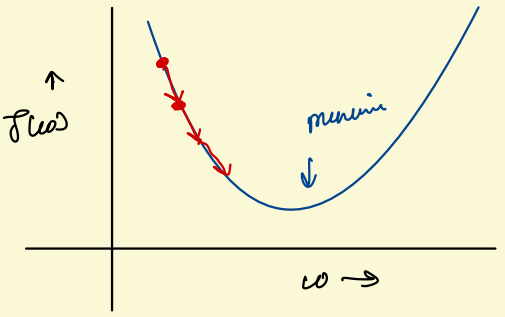
$w = w - \alpha \frac{d}{dw} J(w)$

$w = w - \alpha$ (-ive no.)

$w = w + \alpha$ (+ive no.) w↑

Learning rate :

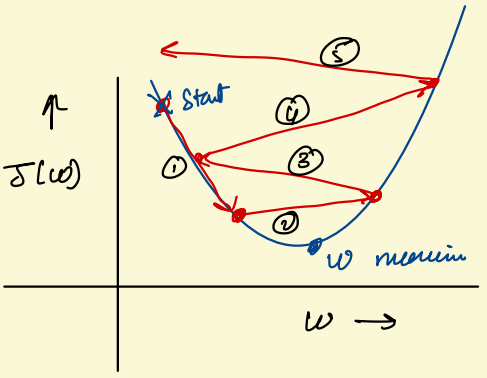
$$w = w - \alpha \frac{d}{dw} J(w)$$



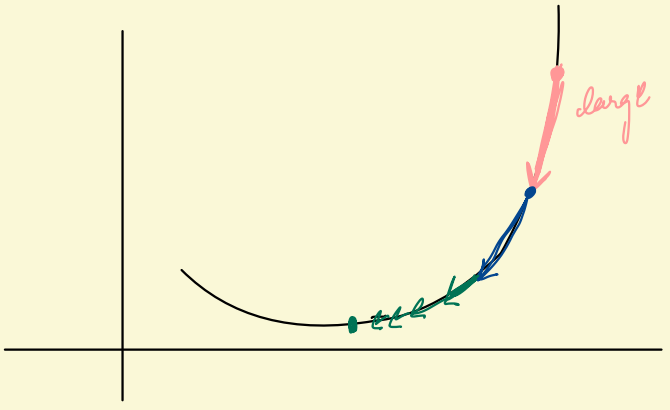
If α is too small ...
Gradient descent may be slow

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

If α is too large ...
Gradient descent may,
Overshoot, never reach min.
→ may diverge



At absolute minimum, the algo does nothing



Linear regression model with Gradient Descent algo:

Linear reg. model

$$f(x) = wx + b$$

Cost function

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

Gradient descent algorithm :

repeat until convergence {

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

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

}

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

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

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

∴ Gradient descent algorithm:

$$w = w - \alpha \frac{1}{m} \sum_{i=1}^{m} (f(x^i) - y^i)\, x^i$$

$$b = b - \frac{\alpha}{m} \sum_{i=1}^{m} (f(x^i) - y^i)$$

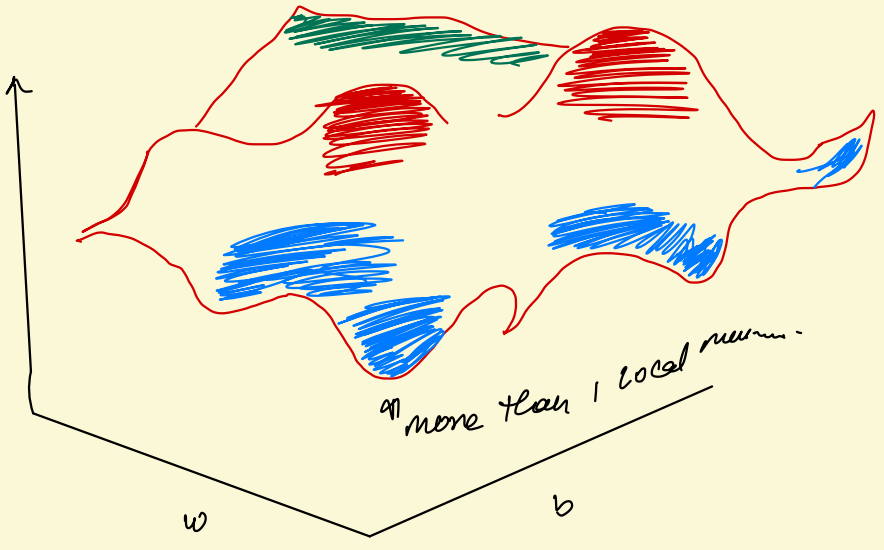
What if the cost func. has more than one local minima?

When squared error cost func. with linear reg. cost function $J(w,b)$ doesn't & will not have multiple local minima

It's a convex function

↓

has single global minimum



∴ As long as α is chosen appropriately grad. desc. always converges.

Batch Gradient descent

"Batch": Each step of gradient descent uses all training eg. (x^i, y^i)

"Stochastic G.D": Each step of grad. descent uses a sample.

→ **Classification**: o/p variable y can take up only a small handful values instead of ∞ range of numbers

Linear regression is not a good algorithm for classification.

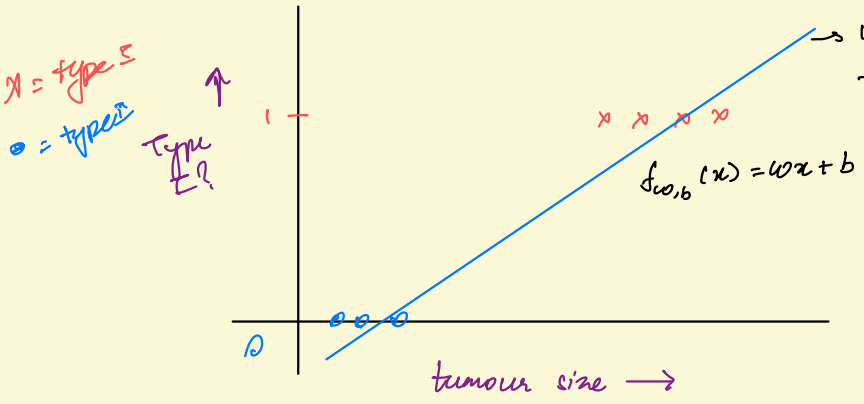
Is this email spam? no / yes

Is tumor Benign? no / yes

type I / type II cancer? no / yes

y can only be one of two values

"binary" classification



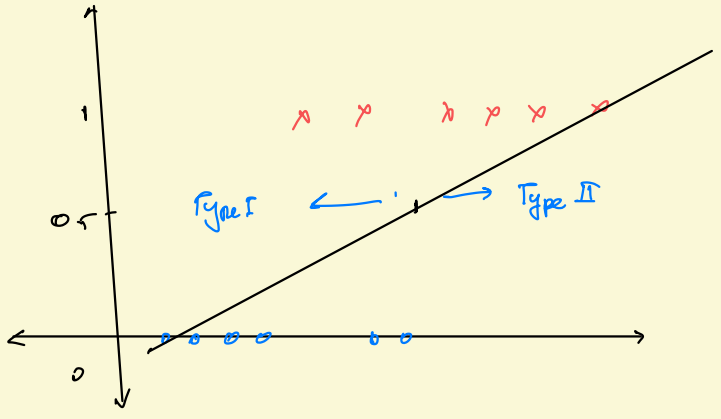
→ using Lenear reg. this line is the best fit for the given data.

Prob: It takes all values b/w 0 & 1

we can add a threshold say $y = 0.5$

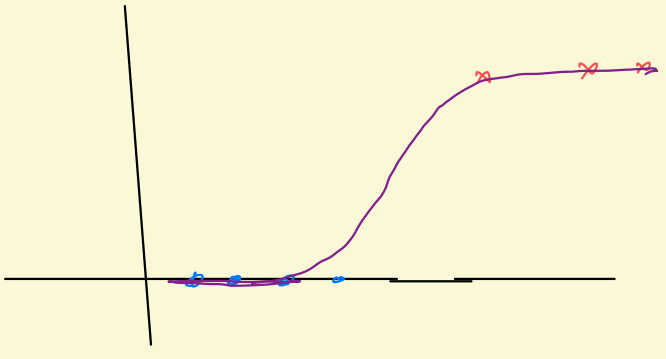
so that if $y > 0.5$ pt's 1 & $y < 0.5$ it is zero.

Prob.

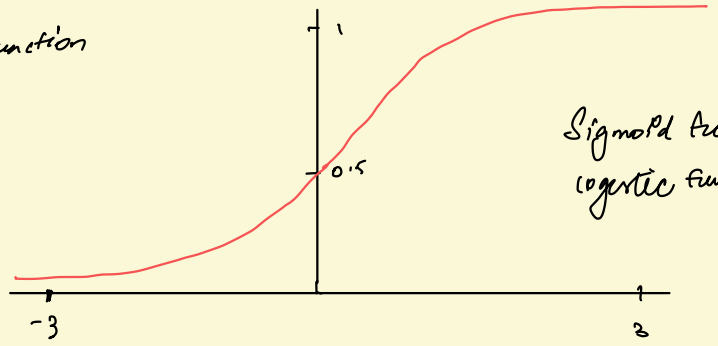


But to the left, the red x are not Type I

Logistic regression



Sigmoid function



Sigmoid func.
Logistic fun

$$g(z) = \frac{1}{1+e^{-z}}$$

$0 < g(z) < 1$

z = large $g(z) \approx 1$

when $z = -$ve, $g(z) = \frac{1}{\text{large no.}} \approx 0$

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

$z = w \cdot x + b$

$g(z) = \frac{1}{1+e^{-z}}$

∴ "Logistic regression" model:

$$f_{\vec{w},b}(\vec{x}) = g(\vec{w} \cdot \vec{x} + b) = \frac{1}{1+e^{-(\vec{w} \cdot \vec{x} + b)}}$$

$f_{\vec{w},b}(\vec{x})$ = "prob." that class is 1

Example: x = tumour size

y = type I or not

If $f_{\vec{w},b}(\vec{x}) = 0.7$ Model thinks that there is 70% chance that the true label $\hat{y} = 1$

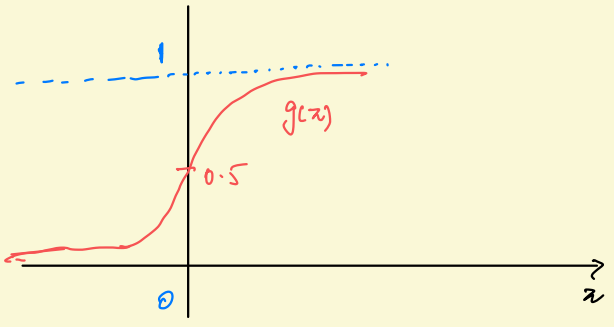
↳ Decision Boundary:

logistic regression is: Take linear reg. model & sigmoid func to fit.

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

$$g(z) = \frac{1}{1+e^{-z}}$$

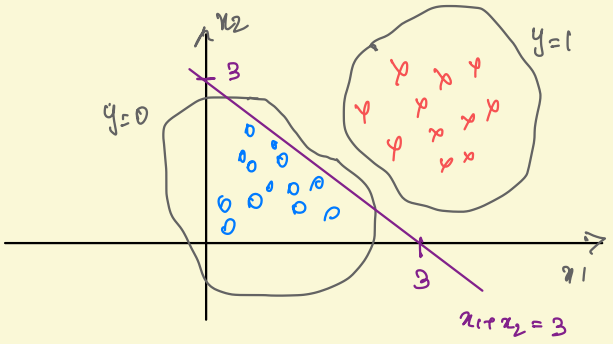
∴ Model is: $f_{\vec{w},b}(\vec{x}) = g(\vec{w} \cdot \vec{x} + b) = \frac{1}{1+e^{-(\vec{w} \cdot \vec{x} + b)}}$



The model predicts $\hat{y} = 1$ whenever:

- $g(z) \geq 0.5$ if $z \geq 0$
- $z \geq 0$ if $\vec{w} \cdot \vec{x} + b \geq 0$

∴ $g(z) \geq 0$ whenever $\vec{w} \cdot \vec{x} + b \geq 0$.



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

$$= g(w_1 x_1 + w_2 x_2 + b) \quad \text{let } b=3$$

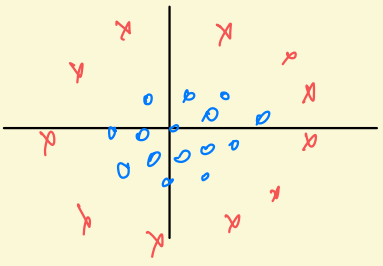
$w_1 = w_2 = 1$

$z = \vec{w}\cdot\vec{x}+b = 0 \rightarrow$ Decision Boundary

$z = x_1 + x_2 - 3 = 0$

$x_1 + x_2 = 3$

Here, Decision Boundary is a straight line.



Non-linear decision Boundary

$$f_{w,b}(x) = w_1 x_1^2 + w_2 x_2^2 + b$$

$$\therefore g(z) = \frac{1}{1+e^{-w_1 x_1^2 - w_2 x_2^2 - b}}$$

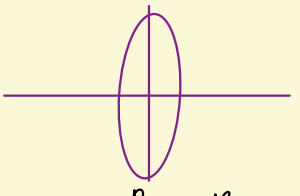
$w_1 = w_2 = 1$ $\quad z = w_1 x_1^2 + w_2 x_2^2 + b$, $w_1 = w_2 = 1$ $\quad b = -1$

Type I (red) when $g(z) \geq 0.5$ when $z \geq 0$ $\quad z = x_1^2 + x_2^2 - 1 \geq 0$ $\quad$ If $x_1^2 + x_2^2 \geq 1$ it's type I

Bound: $x_1^2 + x_2^2 = 1$

If $f_{w,b}(\vec{x}) = g(z) = g(w_1 x_1 + w_2 x_2 + w_3 x_1^2 + w_4 x_1 x_2 + w_5 x_2^2 + w_6 x_1^3 + \cdots + b)$

If $f_{w,b}(\vec{x}) = g(w_1 x_1 + w_2 x_2 + w_3 x_1^2 + w_4 x_1 x_2 + w_5 x_2^2)$



IMPT

(I) For logistic regression, choose a model (linear or poly) & feed it into the sigmoid function to observe the o/p.

(V) In order to choose a model, we need to choose parameters w, b by minimizing a cost function.

Cost function for logistic regression model.:

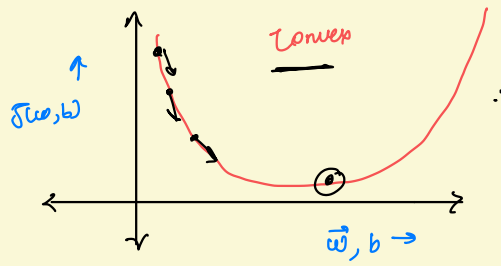
$$f_{w,b}(x) = \vec{w}\cdot\vec{x} + b$$

$$\therefore f_{w,b}(x) = g(z) = \frac{1}{1+e^{-(\vec{w}\cdot\vec{x}+b)}}$$

Choose w, b

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

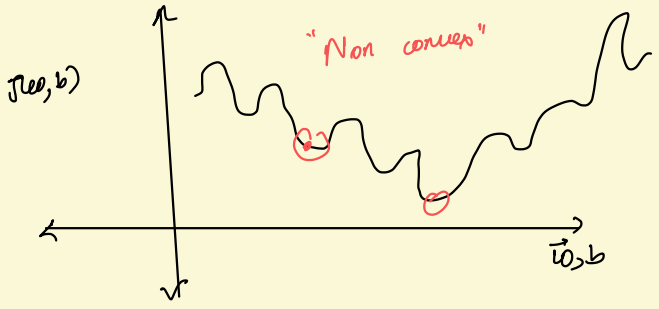
$$J(\vec{w},b) = \frac{1}{2m} \sum_{i=1}^{m} \left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)}\right)^2 \rightarrow \text{error or cost function}$$



$\therefore$ Single global minima

If we use the same cost fun. for $f_{\vec{w},b}(\vec{x}) = \frac{1}{1+e^{-(\vec{w}\cdot\vec{x}+b)}}$, logistic regression fun.

$$J(\vec{w},b) = \frac{1}{2m} \sum_{i=1}^{m} \left(\frac{1}{1+e^{-(\vec{w}\vec{x}+b)}} - y^{i}\right)^2 \rightarrow \text{this cost fun. is non linear}$$



$\therefore$ Squared error cost function is not good enough.

$\therefore$ We choose a different cost fun. that makes the cost func. convex again

$\therefore$ Gradient descent can be guaranteed to converge to the global minima

* Building a new cost function:

* loss function in a cost fun: $J(\vec{w},b) = \frac{1}{m} \sum_{i=1}^{m} \frac{1}{2} \left(f_{\vec{w},b}(x^{i}) - y^{i}\right)^2$

loss function: $L\left(f_{\vec{w},b}(x^{i}), y^{i}\right)$ → predictor o/p fun.

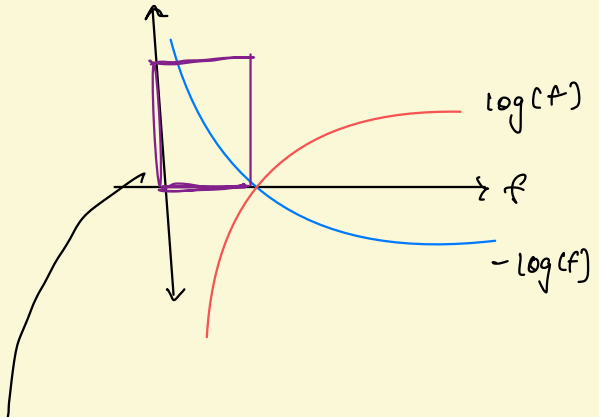
true value y.

Here we give $L\left(f_{\vec{w},b}(x^{i}), y^{i}\right) = \frac{1}{2}\left(f_{\vec{w},b}(x^{i}) - y^{i}\right)^2$

$$L\left(f_{\vec{w},b}(\vec{x}^{i}), y^{i}\right) = \begin{cases} -\log\left(f_{\vec{w},b}(x^{i})\right) & , \ y^{i}=1 \\ -\log\left(1 - f_{\vec{w},b}(\vec{x}^{(i)})\right) & , \ y^{i}=0 \end{cases}$$

Observe $f_{\vec{w},b}(x^{i})$ is always b/w 0 and 1

$\therefore$

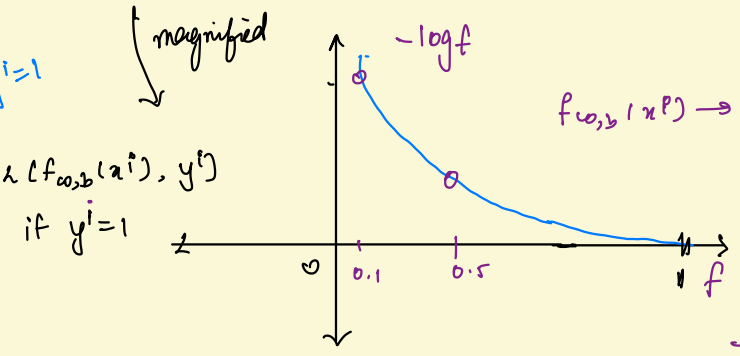


If algo. predicts $f_{\vec{w},b}(x^{i})$ close to 1 & true value is 1,

For logistic reg. o/p is b/w 0 & 1

Case I

Label $y^i = 1$



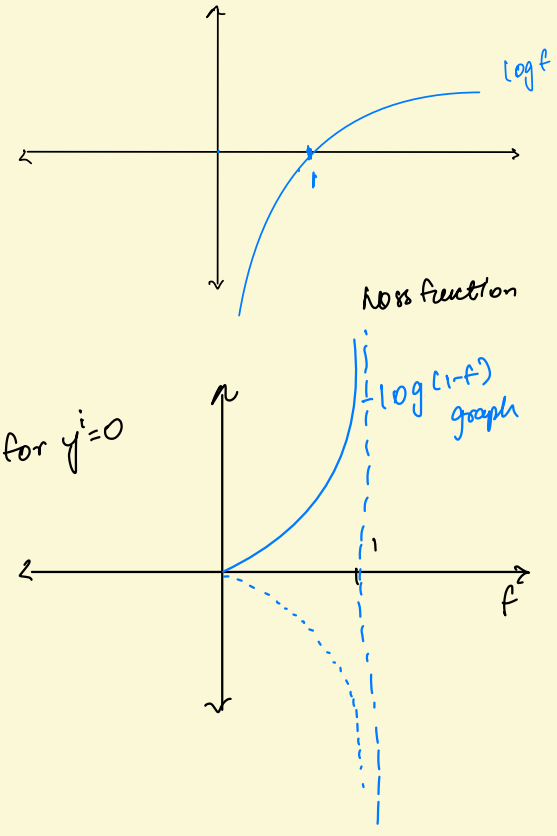
$f_{w,b}(x^i) \to 1 \quad \mathcal{L}[f_{w,b}(x^i), y^i) \to 0$

If $f_{w,b}(x^i) = 0.5$, loss is 0

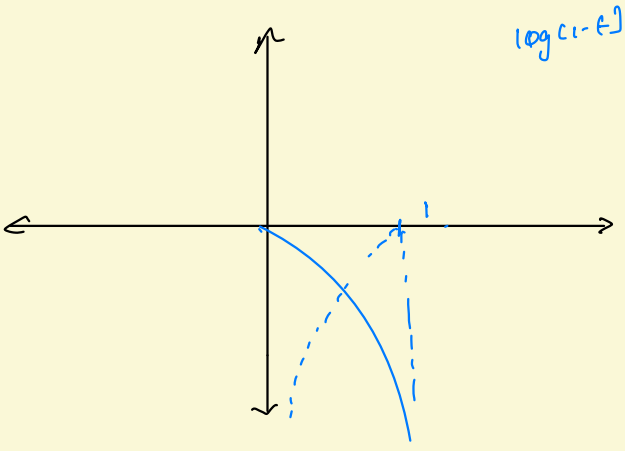
If $f_{w,b}(x^i) = 0.1$, this says that the chance of $y^i = 1$ is only 10 percent which is false.

Case II

Label $y^i = 0$



for $0 < f < 1$, $\log(1-f)$ graph looks like



∴ When $f = 0$, or very close to zero, loss func $\to 0$

i.e true label is zero & predictor label = 0.

When $f \to 1$, loss func $\to \infty$.

→ logistic squared error not a good choice.

- $\mathcal{L}[f_{\vec{w},b}(\vec{x}^i), y^i] = \begin{cases} -\log(f_{\vec{w},b}(x^i)) & , y^i = 1 \\ -\log(1 - f_{\vec{w},b}(\vec{x}^{(i)})) & , y^i = 0 \end{cases}$ → Also, this choice of loss func. is convex.

$$\therefore J(w,b) = \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}[f_{w,b}(\vec{x}^i), y^i]$$

$$\ln\left(\frac{1}{1 + e^{-wx+b}}\right)$$

Simplified cost func. for Logistic Regression:

$$\mathcal{L}[f_{w,b}(x^i), y^i] = \begin{cases} -\log(f_{w,b}(x^i)) & , y^i = 1 \\ -\log(1 - f_{w,b}(x^i)) & y^i = 0 \end{cases}$$

∴ loss function:

$$L[f_{\vec{w},b}(\vec{x}^{(i)}), y^{(i)}] = -y^{(i)} \log(f_{\vec{w},b}(\vec{x}^{(i)})) - (1-y^{(i)}) \log(1 - f_{\vec{w},b}(\vec{x}^{(i)}))$$

$$\therefore J(\vec{w},b) = \frac{1}{m} \sum_{i=1}^{m} [L(f_{\vec{w},b}(\vec{x}^{(i)}), y^{(i)})] = -\frac{1}{m} \sum_{i=1}^{m} \left(y^{(i)} \log(f_{\vec{w},b}(\vec{x}^{(i)})) + (1-y^{(i)}) \log(1 - f_{\vec{w},b}(\vec{x}^{(i)})) \right)$$

This particular cost func. is derived from statistics using max likelihood estimator.

★ GRADIENT DESCENT: to find the min of $J(\vec{w},b)$

cost:

$$J(\vec{w},b) = -\frac{1}{m} \sum_{i=1}^{m} \left(y^{(i)} \log(f_{\vec{w},b}(\vec{x}^{(i)})) + (1-y^{(i)}) \log(1 - f_{\vec{w},b}(\vec{x}^{(i)})) \right)$$

$$f_{\vec{w},b} = \frac{1}{1 + e^{-(\vec{w}\cdot\vec{x} + b)}}$$

repeat {

$$w_j = w_j - \alpha \frac{\partial}{\partial w_j} J(\vec{w},b) \qquad j = 1, \dots, n$$

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

} simultaneous updates

$$\frac{\partial}{\partial w_j} J(\vec{w},b) = -\frac{1}{m} \sum_{i=1}^{m} \frac{y^{(i)}}{f_{\vec{w},b}(\vec{x}^{(i)})} \left(x_j \left(\frac{1}{1 + e^{-(\vec{w}\cdot\vec{x} + b)}} \right)^2 e^{-(\vec{w}\cdot\vec{x} + b)} \right) - \frac{(1-y^{(i)})}{1 - f_{\vec{w},b}(\vec{x}^{(i)})} \left(x_j \left(\frac{1}{1 + e^{-(\vec{w}\cdot\vec{x} + b)}} \right)^2 e^{-(\vec{w}\cdot\vec{x} + b)} \right)$$

$$\left(\frac{y^{(i)}}{f} - \frac{1-y^{(i)}}{1-f} = \frac{y^{(i)} - y^{(i)} f - f + y^{(i)} f}{f(1-f)} = \frac{y^{(i)} - f}{f(1-f)} \right)$$

$$f(1-f) = \left(\frac{1}{1 + e^{-(wx+b)}} \right) \left(1 - \frac{1}{1 + e^{-(wx+b)}} \right) = \frac{1}{(1 + e^{-(wx+b)})^2} \cdot e^{-(wx+b)}$$

$$\frac{\partial}{\partial w_j} J(w,b) = -\frac{1}{m} \sum_{i=1}^{m} \left(\frac{y^{(i)}}{f} - \frac{(1-y^{(i)})}{1-f} \right) \left(x_j \, f(1-f) \right)$$

$$= -\frac{1}{m} \sum_{i=1}^{m} (y^{(i)} - f) x_j f = -\frac{1}{m} \sum_{i=1}^{m} (y^{(i)} - f) x_j = \frac{1}{m} \sum_{i=1}^{m} (f - y^{(i)}) x_j^{(i)}$$

$$J(\vec{w},b) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^i \log\left(f_{w,b}(x^i)\right) + (1-y^i)\log\left(1-f_{w,b}(x^i)\right)\right]$$

$$f_{w,b} = \frac{1}{1+e^{-(\vec{w}\cdot\vec{x}+b)}}$$

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

$$\left(\frac{y^i}{f} - \frac{(1-y^i)}{1-f}\right) = \frac{y^i(1-f) - f(1-y^i)}{f(1-f)} = \frac{y^i - f}{f(1-f)}$$

$$f(1-f) = \frac{1}{1+e^{-(w\cdot x+b)}}\left(1 - \frac{1}{1+e^{-(wx+b)}}\right) = \frac{e^{-wx+b}}{(1+e^{-wx+b})}$$

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

∴

Gradient descent algorithm:

repeat

{

$$w_j = w_j - \alpha\left[\frac{1}{m}\sum_{i=1}^{m}\left(f_{w,b}(\vec{x}^i) - y^i\right)x_j^i\right]$$

$$b = b - \alpha\left[\frac{1}{m}\sum_{i=1}^{m}\left[f_{w,b}(x^i) - y^i\right]\right]$$

} simultaneous updates

↓ with old w only

Linear Regression using closed form.

CLOSED FORM:

loss func. : $J(w,b) = \frac{1}{2m}\sum_{i=1}^{m}\left(y^{(i)} - f_{w,b}(x^{(i)})\right)^2$

$f_{w,b}(x^{(i)}) = w_1 x_1 + w_2 x_2 + b \cdot 1$

$\vec{w} = \begin{bmatrix} w_1 \\ w_2 \\ b \end{bmatrix}$ $\quad \vec{x}_i = \begin{bmatrix} x_{i1} \\ x_{i2} \\ 1 \end{bmatrix}$ $\quad \therefore f_{w,b}(x^{(i)}) = \begin{bmatrix} w_1 & w_2 & b \end{bmatrix} \begin{bmatrix} x_{1i} \\ x_{2i} \\ 1 \end{bmatrix}$

$f_{w,b}(x^{(i)}) = w^T \vec{x}_i$

$\vec{w} \cdot \vec{x}_i$

$J(w,b) = J(\vec{w})$

$J(\vec{w}) = \frac{1}{2m}\sum_{i=1}^{m}\left(y^i - \vec{w}\cdot\vec{x}_i\right)^2$

Minimize $J(\vec{w})$: $\frac{\partial J}{\partial w} = 0$

$\frac{\partial J}{\partial w} = \frac{1}{m}\sum_{i=1}^{m}\left(y^i - \vec{w}\cdot\vec{x}_i\right)(-\vec{x}_i) = 0$

$\frac{\partial J}{\partial w} = -\frac{1}{m}\sum_{i=1}^{m} y^i \vec{x}_i + \frac{1}{m}\sum_{i=1}^{m}(\vec{w}\cdot\vec{x}_i)\vec{x}_i = 0$ (scalar, vector, scalar, vector)

$(\vec{w}\cdot\vec{x}_i)\vec{x}_i = (\vec{x}_i \vec{x}_i^T)(\vec{w})$ (matrix)

$\sum_{i=1}^{m}(\vec{w}\cdot\vec{x}_i)\vec{x}_i = \sum_{i=1}^{m}(\vec{x}_i \vec{x}_i^T)(\vec{w})$

$\frac{\partial J}{\partial w} = -\sum_{i=1}^{m} y^i \vec{x}_i + \left(\sum_{i=1}^{m}(\vec{x}_i \vec{x}_i^T)\right)(\vec{w}) = 0$

vector B $\quad$ Matrix A

For $m=4$:

$\sum_{i=1}^{4} y^i \vec{x}_i = y^1\begin{bmatrix} x_{11} \\ x_{12} \\ 1 \end{bmatrix} + y^2\begin{bmatrix} x_{21} \\ x_{22} \\ 1 \end{bmatrix} + y^3\begin{bmatrix} x_{31} \\ x_{32} \\ 1 \end{bmatrix} + y^4\begin{bmatrix} x_{41} \\ y_{42} \\ 1 \end{bmatrix} = b$

$$\sum_{i=1}^{4} x_i \cdot x_i^T = \begin{bmatrix} x_{11} \\ x_{12} \\ 1 \end{bmatrix} \begin{bmatrix} x_{11} & x_{12} & 1 \end{bmatrix} + \begin{bmatrix} x_{21} \\ x_{22} \\ 1 \end{bmatrix} \begin{bmatrix} x_{21} & x_{22} & 1 \end{bmatrix} + \begin{bmatrix} x_{31} \\ x_{32} \\ 1 \end{bmatrix} \begin{bmatrix} x_{31} & x_{32} & 1 \end{bmatrix} + \begin{bmatrix} x_{41} \\ x_{42} \\ 1 \end{bmatrix} \begin{bmatrix} x_{41} & x_{42} & 1 \end{bmatrix}$$

$$\therefore \quad -\vec{b} + A\vec{w} = 0$$

$$\vec{w} = (A)^{-1} b$$

Disadvantage: ① A^{-1} not always invertible

② For problems having more no. of features it's difficult to invert matrix A

Obtain

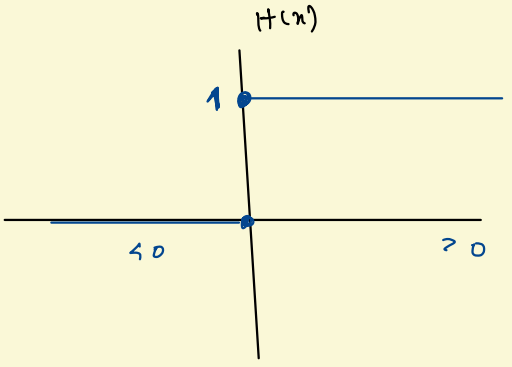
$$y = \text{sgn}(w^{*T} x) = \begin{cases} +1 & \text{if } w^{*T} x \geq 0 \\ -1 & \text{otherwise.} \end{cases}$$

Minimum Classification Error:

(Aim: to find optimum $\vec{w}$)

Idea: is to minimize misclassification

(1) $-y_i w^T x_i = \begin{cases} > 0, & \text{misclassification} \\ < 0, & \text{correct classification.} \end{cases}$ → (Do u agree?)



"Hard errors"

$$E_0(w) = \sum_{i=1}^{N} H(-y_i w^T x_i)$$

Claim: $E_0(w)$ counts the misclassification.

if misclassification: $-y_i w^T x_i > 0$ & $\therefore H(>0) = 1$

It assigns count 1 for each misclassification

if correct classification: $y_i w^T x_i < 0$ & $H(<0) = 0$

It assigns count 0 for each correct classification.

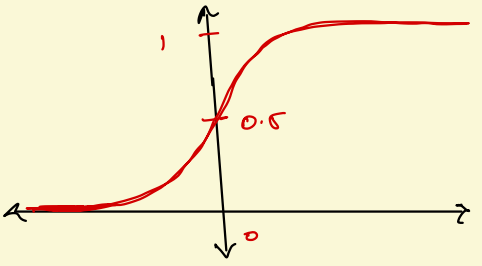
So if we minimize $E_0(w)$ & find optimum w we r done.

problem: the obj. func. E_0 is difficult to optimize.

why? $[\nabla H = 0$ a.e except @ origin)

$\therefore$ Minimizing $E_0(w)$ not a good idea.

(2) Using the Sigmoid function.



$$l(x) = \frac{1}{1 + e^{-x}}$$

as $x \to \infty$ $l(x) \to 1$ and $x \to -\infty$ $l(x) \to 0$

loss func is probability of misclassification.

Threshold = 0.5

$-y_i w^T x_i = \begin{cases} > \text{threshold}, & \text{misclassification} \\ < \text{threshold}, & \text{correct classification.} \end{cases}$ "Soft Error"

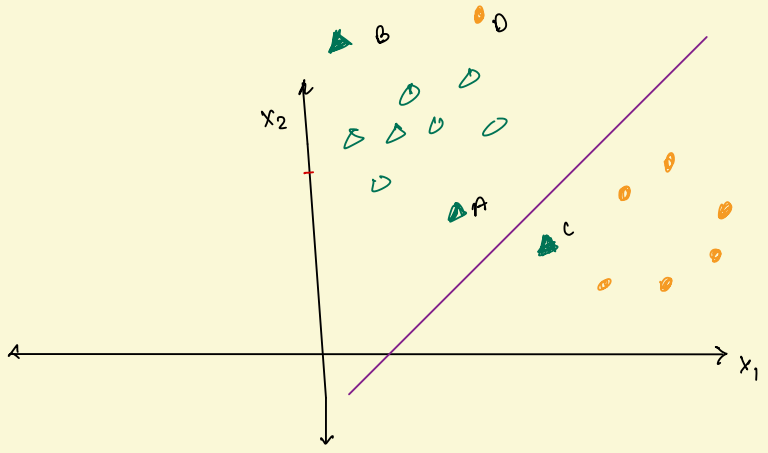
$$E_1(\vec{w}) = \sum_{i=1}^{N} l(-y_i w^T x_i)$$

loss function = 1 when misclassified
loss func. = 0 when correctly classified

Observations: ① $l(-y_i w^T x_i)$ is a qnty b/w 0 and 1

$-y_i w^T x_i > 0$ $\quad E_1(w) = \sum_{i=1}^{N} l(-y_i w^T x_i)$

Just think about this

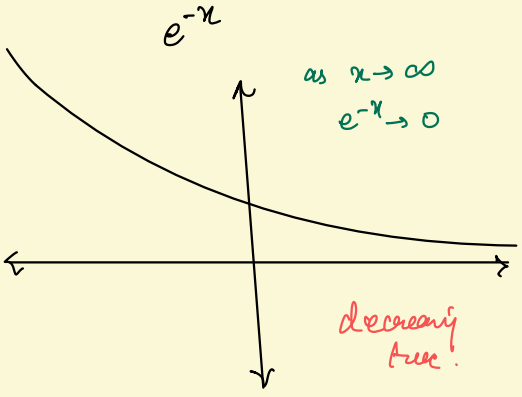


what can you say about data points value of l at points :

(a) A and B

@ A : $l(-y_i w^T x_i)$ threshold

@ B : $l(-y_i w^T x_i)$ threshold



Some maths :

$x_1 < x_2$

$e^{-x_1} > e^{-x_2} \Rightarrow 1 + e^{-x_1} > 1 + e^{-x_2}$

$\frac{1}{1+e^{-x_1}} < \frac{1}{1+e^{-x_2}} \Rightarrow l(x_1) < l(x_2)$

Both $-y_i w^T x_i$ and $-y_i w^T x_i$ are -ve at A & B. [∵ they are correctly classified.

$-y_i w^T x_i$ @ B < $-y_i w^T x_i$ @ A $\quad l(-y_i w^T x_i)$ @ B < $l(-y_i w^T x_i)$ @ A

loss @ B < loss @ A ∴ B is more correctly classified than A

(b) at C & D

$l(-y_i w^T x_i) <$ threshold @ C

$l(-y_i w^T x_i) <$ threshold @ D

both $-y_i w^T x_i$ @ C & $-y_i w^T x_i$ @ D are +ive. (∵ misclassified)

$-y_i w^T x_i$ @ C < $-y_i w^T x_i$ @ D (∵ it's a misclassified point & it is a negative value remember)

∴ $l(-y_i w^T x_i)$ @ C < $l(-y_i w^T x_i)$ @ D

↳ loss @ C < loss @ D

i.e. C is comparitively less misclassified than D.

Minimizing E_1:

$$E_1(\vec{w}) = \sum_{i=1}^{N} l(-y_i w^T x_i) = \text{Probability of misclassification}$$

Minimizing $E_1(w)$

$$E_1(w) = \sum_{i=1}^{N} \frac{1}{1 + e^{y_i w^T x_i}}$$

Minimize w: $\frac{\partial E_1}{\partial w} = 0$

$$\frac{\partial E_1}{\partial w} = \sum_{i=1}^{N} \frac{\partial l(-y_i w^T x_i)}{\partial w}$$

$$\frac{\partial l}{\partial w}(-y_i w^T x_i) = \frac{\partial}{\partial w}\left(\frac{1}{1 + e^{y_i w^T x_i}}\right) = \frac{-e^{y_i w^T x_i}}{(1 + e^{y_i w^T x_i})^2}(y_i x_i)$$

$$l(-y_i w^T x_i) = \frac{1}{1 + e^{y_i w^T x_i}} \qquad 1 - l(-y_i w^T x_i) = \frac{-e^{y_i w^T x_i}}{1 + e^{y_i w^T x_i}}$$

$$1 - l(-x) = 1 - \frac{1}{1 + e^{x}} = \frac{e^x}{1 + e^x} = \frac{1}{1 + e^{-x}} = l(x)$$

$$1 - l(-x) = l(x)$$

$$\frac{\partial E_1}{\partial w} = \sum_{i=1}^{N} y_i \, \sigma(y_i w^T x_i) \, (1 - \sigma(y_i w^T x_i)) \, x_i$$

Thus by minimizing $E_1(w)$ we can get the optimal w.

we can view $\sigma(-y_i w^T x_i)$ as prob. of making an error on training sample. (x_i, y_i) using model w.

* Logistic Regression :=>

Joint probability distribution :

correct classification:

prob. of misclassification : $\sigma(-y_i w^T x_i)$

prob. of correct classification : $1 - \sigma(-y_i w^T x_i)$ (But $1 - \sigma(-x) = \sigma(x)$)

Prob. of a correct classification @ (x_i, y_i) is : $\sigma(y_i w^T x_i)$

∴ Prob of correct classification for all samples in training set assuming samples are independent & i.i.d.

$$E_{LR}(w) = \arg\max_{w} L(w) = \arg\max_{w} \prod_{i=1}^{N} \sigma(y_i w^T x_i)$$ → maximize the prob. of correct classification.

$$= \arg\max_{w} \ln L(w) = \arg\max_{w} \sum_{i=1}^{N} \ln(\sigma(y_i w^T x_i))$$

($1 - \sigma(-x) = \sigma(x)$)

$\frac{1}{\sigma(x)} = \frac{1}{1 - \sigma(-x)}$

Equivalent to

Minimize $$\sum_{i=1}^{N} -\ln(\sigma(y_i w^T x_i))$$

$$\sum_{i=1}^{N} -\ln(\sigma(y_i (\vec{w}\vec{x}_i + b))) =$$

$$F(w) = \sum_{i=1}^{N} -\ln\left(\sigma(y_i \cdot w^T x_i)\right) \qquad \sigma(-x) = 1 - \sigma(x)$$

No constraints.

$$\log x = \frac{1}{x}.$$

$$F(w) = \sum_{i=1}^{N} -\ln\left(\frac{1}{1+e^{-y_i w^T x_i}}\right)$$

$$\frac{\partial F}{\partial w} = \sum_{i=1}^{N} -\left(1+e^{-y_i w^T x_i}\right) \frac{1}{\left(1+e^{-y_i w^T x_i}\right)^2} e^{-y_i w^T x_i} (y_i x_i)$$

$$= -\sum_{i=1}^{N} \frac{y_i\, e^{-y_i w^T x_i} x_i}{\left(1+e^{-y_i w^T x_i}\right)}$$

$$1 - \sigma(y_i w^T x_i) = 1 - \frac{1}{1+e^{-y_i w^T x_i}} = \frac{e^{-y_i w^T x_i}}{1+e^{-y_i w^T x_i}}$$

✯ Multivariable linear regression

* The model is given as:

$$f(w,b) = w_1 x_1 + w_2 x_2 + b$$

$$x\_train = \begin{matrix} \\ 1 \\ \vdots \\ m \end{matrix} \begin{matrix} x_1 \; x_2 \\ \begin{bmatrix} \cdot & \cdot \\ & \\ & \end{bmatrix} \end{matrix}$$

* y-pred.

"m samples"

$$y\text{-}pred. = \begin{matrix} 1 \\ 2 \\ \vdots \\ m \end{matrix} \begin{bmatrix} w_1 \cdot x_{11} + w_2 x_{12} + b \\ \\ w_1 x_{m1} + w_2 x_{m2} + b \end{bmatrix} = \begin{matrix} 1 \\ 2 \\ \vdots \\ m \end{matrix} \begin{bmatrix} [x_{11} \;\; x_{12}] \cdot [w_1, w_2] + b \\ [x_{21} \;\; x_{22}] \cdot [w_1, w_2] + b \\ \vdots \\ [x_{m1}, x_{m2}] \cdot [w_1, w_2] + b \end{bmatrix}$$

$$\begin{bmatrix} [x_{11} \;\; x_{12}] \\ \vdots \\ [x_{m1} \;\; x_{m2}] \end{bmatrix} \underbrace{\begin{bmatrix} [w_1] \\ [w_2] \end{bmatrix}} + b = np.dot(x, weights) + b$$

matrix multiplication

np.dot(A, B) = AB - mat. multip

Model: $f_{w,b}(x^{(i)}) = w_1 x_{i1} + w_2 x_{i2} + b$

* The cost function is given as:

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

$$x\_train = \begin{matrix} \\ 1 \\ \vdots \\ m \end{matrix} \begin{matrix} x_1 \; x_2 \\ \begin{bmatrix} \cdot & \cdot \\ & \\ & \end{bmatrix} \end{matrix}$$

Gradient descent algorithm:

$$w = w - \alpha \frac{1}{m} \sum_{i=1}^{m} (f(x^{(i)}) - y^{(i)}) x^{(i)}$$

$$b = b - \frac{\alpha}{m} \sum_{i=1}^{m} (f(x^{(i)}) - y^{(i)})$$

$$y\_train = \begin{matrix} 1 \\ 2 \\ \vdots \\ m \end{matrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}$$

$$y\_pred = \begin{matrix} 1 \\ 2 \\ \vdots \\ m \end{matrix} \begin{bmatrix} w^T x_1 + b \\ w^T x_2 + b \\ \vdots \\ w^T x_m + b \end{bmatrix}$$

Let $dw = \frac{1}{m} \sum_{i=1}^{m} \underbrace{(f(x^{(i)}) - y^{(i)}) \, x^{(i)}}$ dot product.

$$dw = \begin{bmatrix} dw_1 \\ dw_2 \end{bmatrix}$$

3 features →

repeat until convergence: {

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_0^{(i)}$$ → sum over all first components

$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_1^{(i)}$$

$$\theta_2 := \theta_2 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_2^{(i)}$$

...

}

$\theta_0 = w_0$

$\theta_1 = w_1$

$\theta_2 = w_2$

y - y_pred =

$$\begin{bmatrix} w^T x_1 + b - y_1 \\ w^T x_2 + b - y_2 \\ \vdots \\ w^T x + b - y_m \end{bmatrix}$$

$(X\_train)^T = \begin{bmatrix} \begin{bmatrix} x_{11} \\ x_{12} \end{bmatrix} & \begin{bmatrix} x_{21} \\ x_{22} \end{bmatrix} & \cdots & \begin{bmatrix} x_{m1} \\ x_{m2} \end{bmatrix} \end{bmatrix}$ (x_1, x_2, x_m)

$\begin{bmatrix} w^T x_1 + b - y_1 \end{bmatrix} \begin{bmatrix} x_{11} & x_{12} \end{bmatrix}$

$(X\_train)^T \cdot (y - y\_pred) = \begin{bmatrix} \begin{bmatrix} x_{11} \\ x_{12} \end{bmatrix} & \begin{bmatrix} x_{21} \\ x_{22} \end{bmatrix} & \cdots & \begin{bmatrix} x_{m1} \\ x_{m2} \end{bmatrix} \end{bmatrix} \cdot \begin{bmatrix} f(x_1) - y_1 \\ f(x_2) - y_2 \\ \vdots \\ f(x_m) - y_m \end{bmatrix}$ matrix Multip

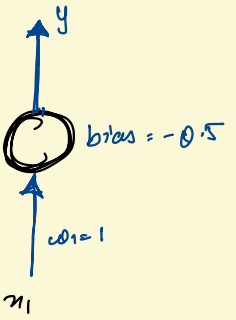
$= \begin{bmatrix} x_{11}(f(x_1) - y_1) + \cdots + x_{m1}(f(x_m) - y_m) \\ x_{12}(f(x_1) - y_1) + \cdots + x_{m2}(f(x_m - y_m)) \end{bmatrix} = dW$

$\therefore \quad dw = \left(\frac{1}{m}\right) np.dot\left((x\_train)^T, (y - y\_pred)\right)$

$db = \left(\frac{1}{m}\right) np.sum(y - y\_pred)$

Why do we need sigmoids?

↳ Thresholding logic used by perceptron is very harsh



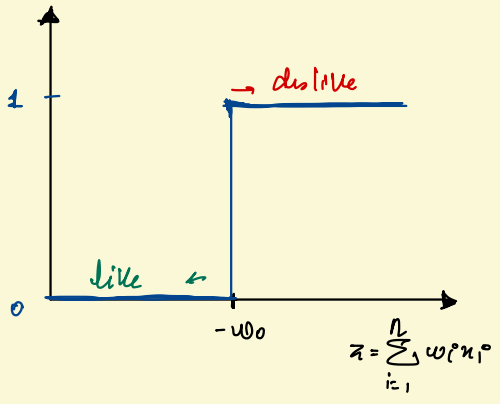
like / Dislike a movie, bias = -0.5 ⇒ thresh: 0.5

if critics = 0.51 ≥ 0.5 like ✔

if o/p = 0.49 dislike ✗

decision is harsh.

critics rating

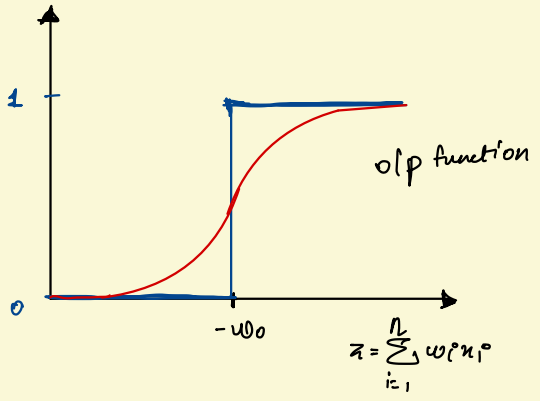


if $z \geq -w_0$ fire, else don't fire.

If else present

This is harsh

Not continuous, not diff.



Sigmoid: is a family of functions

One such func is Logistic function.

$$y = \frac{1}{1 + e^{-w^T x}}$$

No longer a sharp transition around the threshold $-w_0$ & value is b/w 0 & 1

Smooth, cts & diff.

Interpretation of sig. func. as a prob:

with 51% prob. will like the movie

with 49% prob. will like the movie