Supervised learning

$x \rightarrow y$

Input output

* Type

1) Regression — predict number
2) classification — predict categories

* Structure

training set

↓

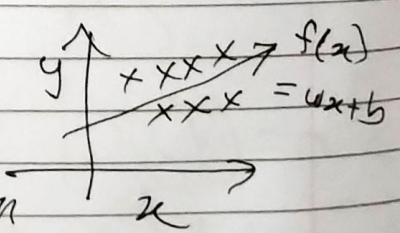
learning algorithm

$x \rightarrow \boxed{f} \rightarrow \hat{y}$

feature model prediction (estimated y)

for regression with straight line and one variable we can define with

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



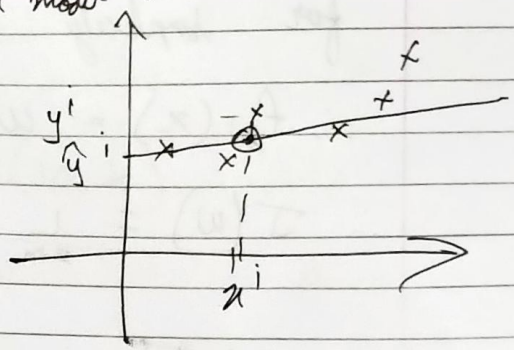
univariate linear regression

$y^{(i)}$ → actual value

$\hat{y}^{(i)}$ → Prediction our model made

$$\hat{y}^{(i)} = f_{w,b}(x^{(i)})$$

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



* cost function

$$\underbrace{(\hat{y}^{(i)} - y^{(i)})^2}_{\text{error}} \quad \frac{1}{2m}\sum_{i=1}^{m}\left(\hat{y}^{(i)} - y^{(i)}\right)^2 = J(w,b)$$

Mean squared error cost function

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

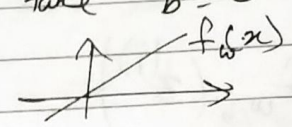
cost function is to check how well predicted line the ~~cost function~~ fit the data

So our goal is to minimize cost funct.

i.e minimize $J(w,b)$

for simplicity lets take $b = 0$

$\therefore f_w(x) = wx$

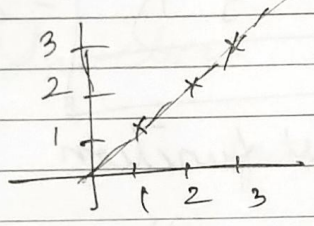


$$\therefore J(w) = \frac{1}{2m} \sum_{i=1}^{m} \left(f_w(x^{(i)}) - y^{(i)} \right)^2$$

$f_w(x)$

if w is fixed, function of x

i.e depended of x

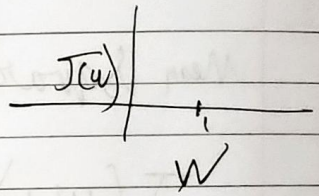


for $w=1$

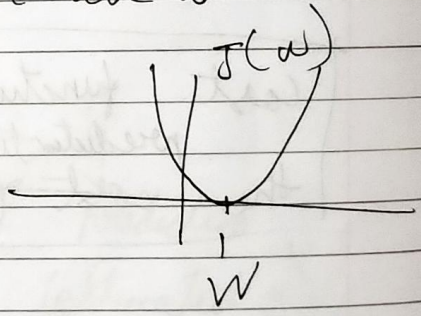
$J(w)$

depends on w i.e parameter

$J(w=1) = 0$



for all w



so as we can see our goal is to reduce w as much as possible to get the best fitted line

for both w on b you will get a 3d plot like a bowl there ~~you~~ you wount

* Gradient descent

It helps us find the local minima easily instead of doing all things manually

G.D algorithm –

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

α = learning rate

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

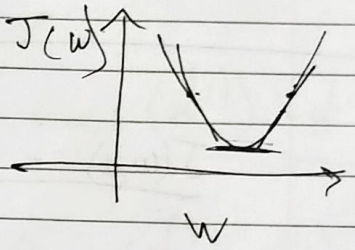
again consider $b=0$

$J(w) = wx$

Then for G.D

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

$$= w_i - \alpha w$$



$f_{w,b} = wx + b$

& $\hat{y}^{(i)} = f_{w,b}(x^{(i)})$ — prediction of our model

$= y^{(i)}$ — actual value

$\hat{y}^{(i)} - y^{(i)} \longrightarrow$ error

$$cost = \frac{1}{2m} \sum_{i=1}^{m} \left(\hat{y}^{(i)} - y^{(i)}\right)^2$$

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

Our goal is to find (w,b) so cost = 0

so to minimize cost we use gradient descent in which we update w, b simultaneously

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

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

$\alpha \Rightarrow$ learning rate

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

```
def compute_cost (x, y, w, b):
    m = x.shape[0]
    cost = 0

    for i in range(m):
        f_wb = w * x[i] + b
        y_i = y[i]

        cost += (f_wb - y_i)**2

    return cost/2m
```

compute_cost gives us J(w,b)

In Gradient descent we do

repeat until convergence : {

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

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

}

so $$\frac{\partial J(w,b)}{\partial w} = \frac{1}{m} \sum_{i=0}^{m-1} \left(f_{w,b}(x^{(i)}) - y^{(i)} \right) x^{(i)}$$

$$\frac{\partial J(w,b)}{\partial b} = \frac{1}{m} \sum_{i=0}^{m-1} \left(f_{(w,b)}(x^{(i)}) - y^{(i)} \right)$$

Move onto ~~Multi variable~~, Multiple linear regression when we have .

$$f_{w,b}(x) = w_1 x_1 + w_2 x_2 + w_3 x_3 + w_4 x_4 + b$$

$$f_{w,b}(x) = w_1 x_1 + w_2 x_2 \ldots w_n x_n + b$$

take $\vec{w}$ as $[w_1, w_2, w_3 \ldots w_n]$
and $\vec{x}$ as $[x_1, x_2, x_3 \ldots x_n]$

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

functions in vector

* Vectorization

Let $\vec{w} = [w_1 \; w_2 \; w_3]$

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

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

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

$J_{\vec{w},b} = \frac{1}{2m} \sum_{i=0}^{n-1}$

way faster than writing manually or using a for loops

* Feature scaling (for a dummy dataset)

* Mean Normalization

$300 \leq x_1 \leq 2000, \; 0 \leq x_2 \leq 5$

if average of $x_1 \Rightarrow 600 = u_1$

we do $x_1 = \frac{x_1 - u_1}{2000 - 300}$, $x_2 = \frac{x_2 - u_2}{5 - 0}$

$-0.18 \leq x_1 \leq 0.82$ $\quad -0.46 \leq x_2 \leq 0.54$

* Z-score normalization

find standard deviation σ

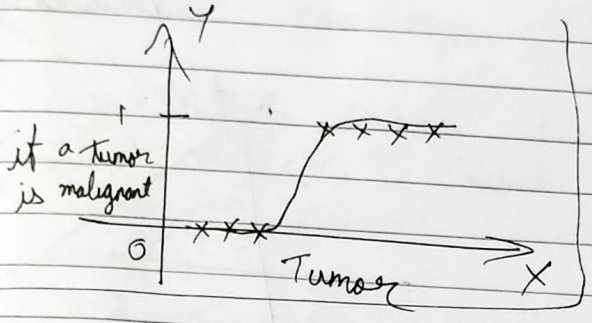
so $x_1 = \frac{x_1 - u_1}{\sigma_1}$ and $x_2 = \frac{x_2 - u_2}{\sigma_2}$

do feature scaling if the any feater is too large / small compared to other feature

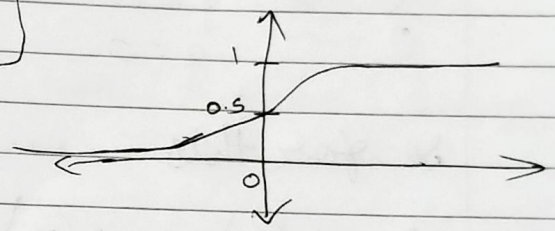
* Feature engineering

combining features to make new feature

Logistic regression → classifying algorith



to map this lines we use the sigmoid function



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

$$0 < g(z) < 1$$

we take $w \cdot x + b = z$

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

$$= P(y = 1 \mid x ; \vec{w}, b)$$

we take a threshold

if $f_{\vec{w},b}(\vec{x}) \geq 0.5$ then

yes → $\hat{y} = 1$

else

no → $\hat{y} = 0$

When is $f_{\vec{w},b}(\vec{x}) \geq 0.5$

$g(z) \geq 0.5$

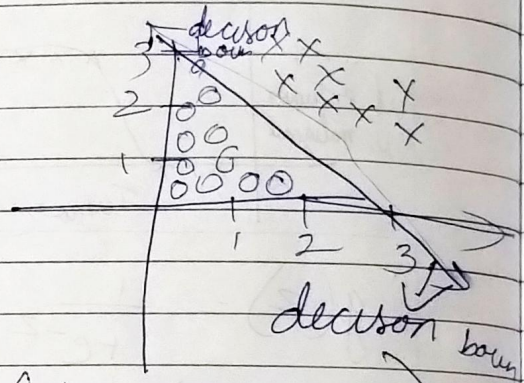
$z \geq 0 \Rightarrow \vec{w} \cdot \vec{x} + b \geq 0$

so when $\vec{w} \cdot \vec{x} + b \geq 0$, $\hat{y} = 1$

else $\vec{w} \cdot \vec{x} + b < 0$, $\hat{y} = 0$

When $z = \vec{w} \cdot \vec{x} + b = 0$ $\Rightarrow$ decison boundary line

example →



x → y = 1
0 → y = 0

so for this

$f_{w,b}(\vec{x}) = g(z) = g(w_1 x_1 + w_2 x_2 + b)$
as we have two features

Lets take $w_1 = 1$, $w_2 = 1$, $b = -3$

so we get $x_1 + x_2 - 3 = z$

decison boundary $\Rightarrow$ $x_1 + x_2 - 3 = 0$

$x_1 + x_2 = 3$

anything above the line will be 1 → true and below will be 0 → negative

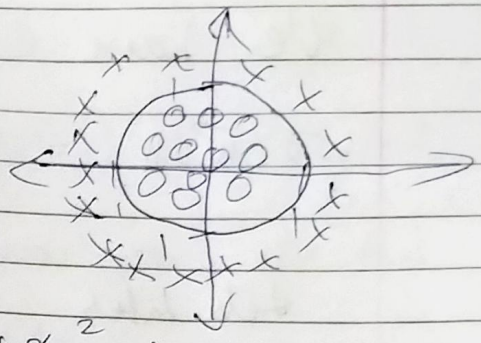
As with linear regression, we can do logistic regression with non linear functions to get non linear decison boundaries

example

here

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

$= g(w_1 x_1^2 + w_2 x^2 + b)$



Let $w_1 = 1, w_2 = 1, b = -1$

∴ decison boundary $\Rightarrow x_1^2 + x_2^2 - 1 = 0$

$x_1^2 + x_2 = 1$

anything inside the close boundary will be ~~false~~ 0 → negative and any point outside circle will be 1 → positive

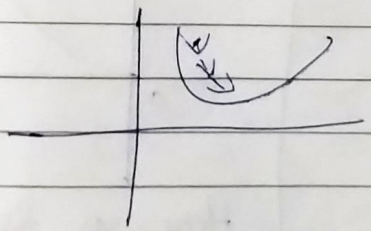
we can have even more complex decison boundaries by having more complex z

Mean squared cost function is not a good choice for regression

as $J(\vec{w}, b) = \frac{1}{m} \sum_{i=1}^{m} \frac{1}{2} \left(f_{\vec{w},b}(x^{[i]}) - \hat{y}^{(i)} \right)^2$

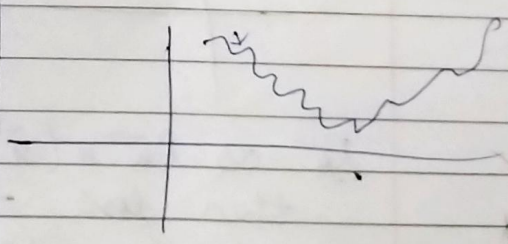
for linear regression

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



for logistic regn.

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



Too many local minimum

We have

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

Lets take $\frac{1}{2}\left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)}\right)^2 \Rightarrow$

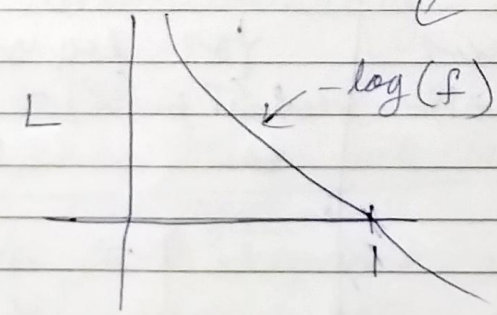
$$L\left(f_{\vec{w},b}(\vec{x}^{(i)}), y^{(i)}\right)$$

So for logistic regression the cost function we will use is

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

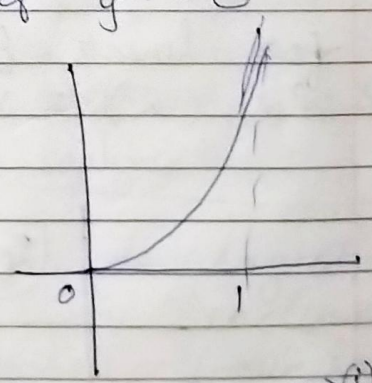
$$= \begin{cases} -\log\left(f_{\vec{w},b}(\vec{x}^{(i)})\right) & \text{if } y^{(i)} = 1 \\ -\log\left(1 - f_{\vec{w},b}(\vec{x}^{(i)})\right) & \text{if } y^{(i)} = 0 \end{cases}$$

if $y^{(i)} = 1$, then graph looks like



So as $f_{\vec{w},b}(\vec{x}^{(i)}) \to 1$ then loss $\to 0$ as its the correct answer else loss would be very more

if $y^{(i)} = 0$



Here as $f_{\vec{w},b}(\vec{x}^{(i)}) \to 0$ Loss approaches $\to 0$ as its the correct label

Or we can simply write loss as

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

if $y^{(i)} = 0$ this becomes zero, else this

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

Gradient descent

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

We do the same ~~for~~ as linear regression for gradient descent

repeat {

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

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

} simultaneously updates

different from Linear regression that

In L.i.R $f_{\vec{w},b}(\vec{x}) = \vec{w}\cdot\vec{x} + b$

In L.O.R $f_{\vec{w},b}(\vec{x}) = \dfrac{1}{1+e^{-(\vec{w}\cdot\vec{x}+b)}}$