COURSE 1

What is Machine Learning?

Definition : Field of study that gives computer the ability to learn without being explicitly programmed. [Checkers example]

Major 2+1 types of learning:

1. Supervised
2. Unsupervised
3. Reinforcement

Supervised Learning part 1

The key characteristic of supervised learning is that you give your learning algorithms example to learn from.

It learns x to y or input → output mapping
The correct pairs of x and y are given and learning algorithm eventually learns to take just the I/P without O/P label and gives reasonably accurate prediction / guess of O/P

$$X \longrightarrow Y$$

I/P O/P label

learns from being given "right answers"

Escamples:

Email	→ spam (0/1)	- spam filter
audio	→ text trans	- speech rec
English	→ spanish	- m/c transl
ad, user info	→ click? (0/1)	- OIL adv
image, radar info	→ posit° of car	- self driving
image of phone	→ defect (0/1)?	- visual inspec

Supervised Learning part 2

Regression
Predict a number from infinitely many possible O/P

Classification
Predict category from small number of possible O/P.

Unsupervised learning part 1

Find something interesting in unlabeled data.
(given I/P x and O/P y, find structure in data)

Part 2

Unsupervised Learning

Clustering
Group similar points together

Dimentionality Red
Compress data using fewer numbers

Anamoly det
Find unusual data points

Linear Regression Model

Linear Regression with one Variable

Terminology:

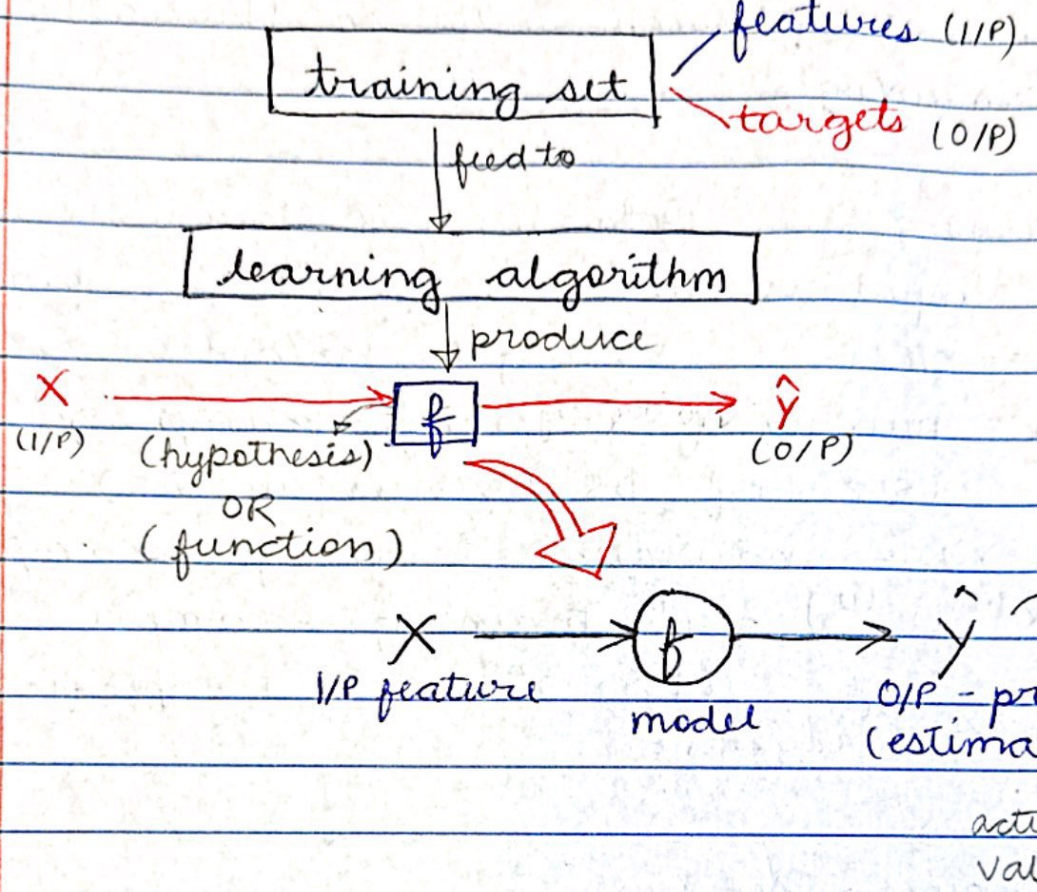
→ Training set : Data used to train the model
→ X = 'input' variable, I/P features, features
→ Y = 'O/P' " , target variables
→ m = number of training example
(size of DS / rows in DS)
→ (X, Y) = individual training example
→ $(X^{(i)}, Y^{(i)})$ = i^{th} training example
↳ ≠ X^i (exponent) (1^{st}, 2^{nd} ..... i - specific row)

Training set:

	x size in feet²	y price ($1000's)
1)	2104	400
2)	1416	232
	1534	315
	⋮	⋮
47)	3210	870

m = 47
$x^{(1)}$ = 2104
$y^{(1)}$ = 400
$x^{(1)}, y^{(1)}$ = 2104, 400

Linear Reg (P2)



How to represent f?

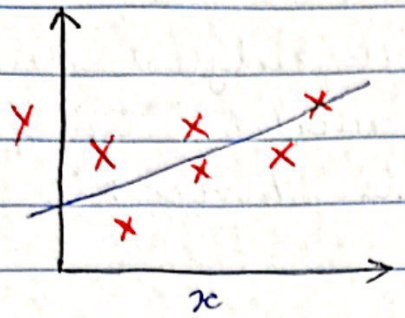
If we assume (for now) that f is a straight line:

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

$= f(x)$

w and b are numbers and predicted $\hat{y}$ will depend on X (I/P feature)

$f_{w,b}(x)$ = f is a f(n), which takes X and depending of w & b it gives $\hat{y}$.



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

$f(x) = wx + b$

This is a linear line : linear regression with one variable / Univariate linear reg.
↳ (Single feature x)

---- LAB ----

Cost Function

Cost fcn tells us how well a model is doing so that we can try to make it better

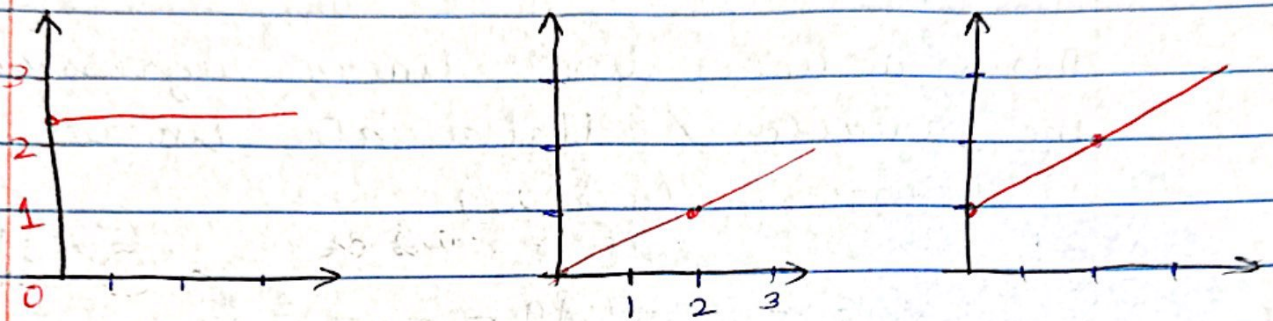
Training set

size in feet² (x) features	price $1000's targets
2104	460
1416	232
⋮	⋮

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

w, b = parameters

Parameters: Variables you adjust during training to improve the model. They are also known as co-efficients / weights.



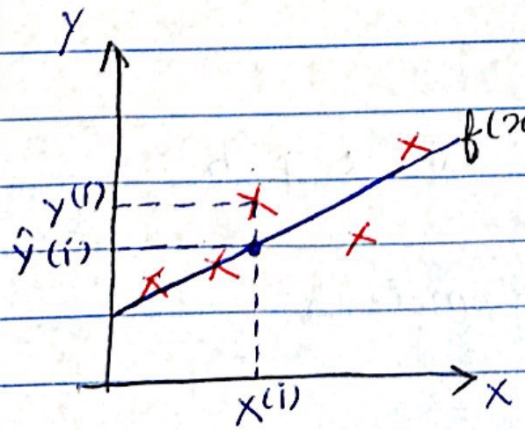
$f(x) = 0.x + 2.5$ $\quad$ $f(x) = 0.5 \cdot x + 0$ $\quad$ $f(x) = 0.5x + 1$

$\hat{y} = 2.5$

$w = 0$ $\quad$ $w = 0.5$ $\quad$ $w = 0.5$

$b = 2.5$ $\quad$ $b = 0$ $\quad$ $b = 1$

Linear Regression:- Choose values of params w and b such that straight line you get for $f(x)$ somehow fits the data (well) passing roughly / somewhere close to almost all train^g ex



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

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

Que: How do you chose values of w and b such that value of $\hat{y}^{(i)}$ (predicted) is close to $y^{(i)}$ (target)

For this, we will see how well a line fits training data. This is measured by cost fn

Cost Function

Squared Error Cost Function

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

(m: # of train ex; 2: sq of error terms)

(m: average calc because we will have large DS)

Since $\hat{y}$ is predicted value, we can rewrite as

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

Cost function intuition

Recap:

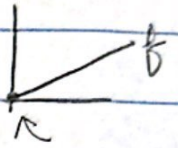
Model	$f_{w,b}(x) = wx + b$
Parameters	w, b
cost f(n)	$J(w,b) = 1/2m \sum_{i=1}^{m} (\hat{y}^{(i)} - y^{(i)})^2$ (predicted, target)
Goal	$\min_{w,b} J(w,b)$

Simplified Version:

Lets assume that $b = 0$ in our eqn

$$f_w(x) = wx \qquad b = \emptyset$$

($b = 0$ ∴ line pass through origin)



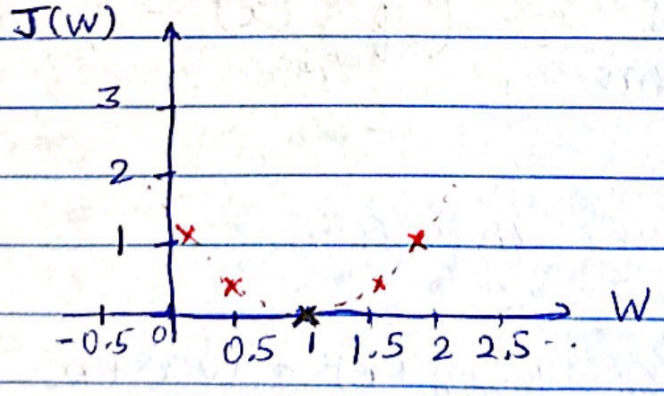
$$J(w) = \frac{1}{2m} \sum_{i=1}^{m} \left(f_w(x^{(i)}) - y^{(i)} \right)^2$$

$f_w(x^{(i)}) \rightarrow wx^{(i)}$ (since $b = 0$)

Visualization:

lets suppose we have points as (1,1), (2,2), (3,3) ....

we start by keeping $w = 0, 0.5, -1, 1$ .. and plot graph b/w w and $J(w)$ for cost:



Goal of linear regression:- Minimize $J(w,b)$ $_{w,b}$

Gradient Descent

$$w = w - \alpha \left[\frac{\partial}{\partial w} J(w, b) \right] \quad \text{(★)}$$

Where

α = learning rate

It controls how big step do we take. If α is large, this means GD is aggressive and we take huge steps & α is too small means very small steps.

★ = Derivative term of cost f(n) J. It tells us about direct^n of learning step.

α and ★ together tells about size of steps

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

Correct: Simultaneous update of parameters

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

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

$w = tempw$

$b = tempb$

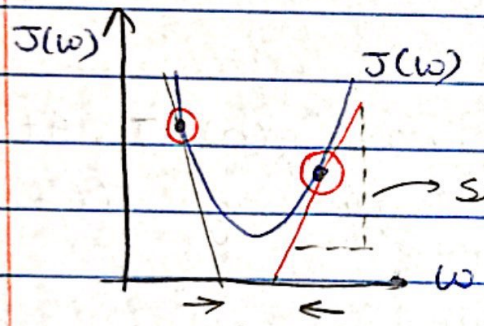
Intuition :-

Repeat Until Convergence {

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

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

}



$$\text{slope} = \frac{\partial}{\partial w} J(w)$$

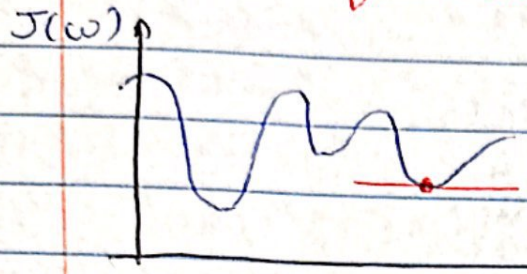
Learning rate is always +ve no.
So, if slope is +, we decrease w and move left. If slope is -ve negative, we move rt

If α is :

① too small
→ GD will be very slow

② too large
→ may fail to converge
→ Overshoot & never reach min

In case of 2 minmas (more than 1)



local minimum (slope = 0)

$\rightarrow W = W - \alpha(0)$

$\therefore W = W$

As we approach minima, slope ↓ i.e derivative becomes ↓

Near local minima

- Derivative gets smaller
- Update steps become smaller

⇒ At local minima GD leave value of W unchanged

Then further GD steps do nothing

Gradient Descent for Linear Algo:

Linear Regression Model

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

Cost function

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

Gradient descent algorithm

repeat until convergence

{

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

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

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

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

}

Optional [Math Derivation]

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

$$= \frac{\partial}{\partial w} \frac{1}{2m} \sum_{i=1}^{m} \left(wx^{(i)} + b - y^{(i)} \right)^2$$

$\frac{d}{dw}(wx+a)^2$ = derivative of $2(wx+a)\cdot x$

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

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

Similarly, we get for $\frac{\partial}{\partial b} J(w,b)$ and put them in GD algo.

Repeat until convergence {

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

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

}

with **simultaneous update** of w and b

This type of curve is known as convex f(n), which have only one global min & we always converge to it.

(optional)

--- LAB WORK VDO ---

WEEK 2 Multiple Variables

Linear Regr with multiple variables

Price (1000's $)	Size in feet^2	#bedrooms	#floors	Age of home
	X_1	X_2	X_3	X_4
Y	①	②	③	④

X_j = j^{th} feature

n = total no. of features (here ④)

row vector ← $\vec{x}^{(i)}$ = features of i^{th} training example.

$x_j^{(i)}$ = value of feature j in i^{th} example

↳ j^{th} feature of i^{th} sample

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

now, since we have multiple features:

$f_{w,b}(X) = w_1x_1 + w_2x_2 + \cdots w_nx_n + b$

eg: $0.1x_1 + 4x_2 + 10x_3 + -2x_4 + 80$

(x_1: size, x_2: #bedrooms, x_3: #floors, x_4: years)

if house is 1 yr old, its price = price − 2k

base

$$f_{\vec{w},b}(\vec{x}) = w_1x_1 + w_2x_2 + \cdots + w_nx_n + b$$

$$\vec{w} = [w_1 \quad w_2 \quad w_3 \quad w_4 \cdots]$$

b is a number

vector → $\vec{x} = [x_1 \quad x_2 \quad x_3 \cdots x_n]$

$$f_{\vec{w},b}(\vec{x}) = \vec{w} \cdot \vec{x} + b \quad (\cdot \text{ → dot product})$$

$$= w_1x_1 + w_2x_2 + w_3x_3 \cdots + w_nx_n + b$$

multiple linear regression
OR
multivariate " "

Vectorization Part 1

Parameters and features: linear algebra = 1-n, python = 0-n-1

$$\vec{w} = [w_1 \quad w_2 \quad w_3 \cdots]$$

b is a number

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

★ dot product in np: $f = np.dot(w,x) + b$ (imp.)

Vectorization makes code shorter

Vectorization part 2

W10 Vect.

```
for j in range(0,16):
    f = f + w[i] * x[i]
```

With Vect:

```
np.dot(w,x)
```

t_0	$f = f + w[0] * x[0]$
t_1	$f = f + w[1] * x[1]$
t_2	⋮
t_{15}	

take t_0 to t_{15}.

t_0	w[0]	w[1]	⋯	w[15]
	x[0]	x[1]	⋯	x[15]

t_1: w[0] * x[0] + [ ] ⋯

takes t_0 & t_1 only

Vectorization enables parallel processing

C1, W2

L01

Gradient Descent for multiple LR:

n features ($n \geq 2$)

$(\vec{f} = \vec{w}\vec{x} + b)$

repeat {

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

$$j=n \quad w_n = w_n - \alpha \frac{1}{m} \sum_{i=1}^{m} (f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)}) x_n^{(i)}$$

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

}

Simult. update $(w_1, w_2, \cdots, w_n)$ and b

	Previous notation	Vector notation
Parameters	$w_1, \cdots, w_n$ b	$\vec{w} = [w_1 \quad \cdots \quad w_n]$ ← vector of length n b still a number
Model	$f_{\vec{w},b}(\vec{x}) = w_1 x_1 + \cdots + w_n x_n + b$	$f_{\vec{w},b}(\vec{x}) = \vec{w} \cdot \vec{x} + b$ ← dot product
Cost function	$J(w_1, \cdots, w_n, b)$	$J(\vec{w}, b)$

Gradient descent

repeat {

$w_j = w_j - \alpha \frac{\partial}{\partial w_j} J(w_1, \cdots, w_n, b)$

$b = b - \alpha \frac{\partial}{\partial b} J(w_1, \cdots, w_n, b)$

}

repeat {

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

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

}

Gradient descent

One feature

repeat {

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

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

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

simultaneously update w, b

}

n features ($n \geq 2$)

repeat {

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

$\frac{\partial}{\partial w_1} J(\vec{w}, b)$

$\vdots$

$j = n$ $w_n = w_n - \alpha \frac{1}{m} \sum_{i=1}^{m} (f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)}) x_n^{(i)}$

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

simultaneously update

w_j (for $j = 1, \cdots, n$) and b

}

← P(1)

Alternative to gradient descent

Normal Equation

→ only for linear regression

→ Solve for w and b w/o iteration

Disad:

→ doesn't solve other (can't generalize)

→ Slow if # features is large (>10,000)

Normal eqn method may be used in ML lib that implement linear regression (in backend)

— LABS — C1W2L2

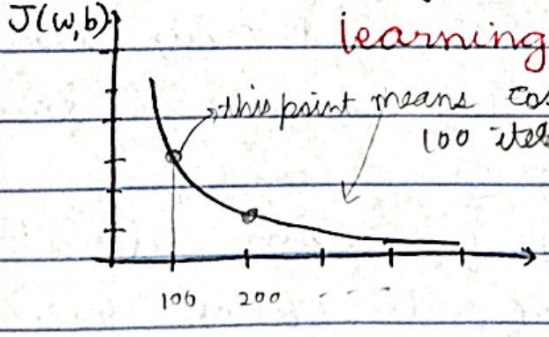
Feature Scaling Part 1

Practical tips for Linear Regression

small change in w_1 ⇒ large change in price (very important)

Print out ← P(2)

Part 2: Choosing α ↓①



better. {gives adv warning & visualizations}

→ ②

Auto convergence test

let ε be 10^{-3} (0.001)

If $J(\vec{w}, b)$ decreased by $\leq \varepsilon$ in one iteration, declare convergence (found w & b close to minima)

Feature and parameter values

$\widehat{price} = w_1 x_1 + w_2 x_2 + b$ (size, #bedrooms)

x_1: size (feet^2) range: $300 - 2{,}000$ (large)

x_2: # bedrooms range: $0 - 5$ (small)

House: $x_1 = 2000$, $x_2 = 5$, $price = \$500$k — one training example

size of the parameters w_1, w_2?

$w_1 = 50$, $w_2 = 0.1$, $b = 50$

$\widehat{price} = 50 * 2000 + 0.1 * 5 + 50$ (100,000K, 0.5K, 50K)

$\widehat{price} = \$100{,}050.5$k $= \$100{,}050{,}500$

$w_1 = 0.1$ (small), $w_2 = 50$ (large), $b = 50$

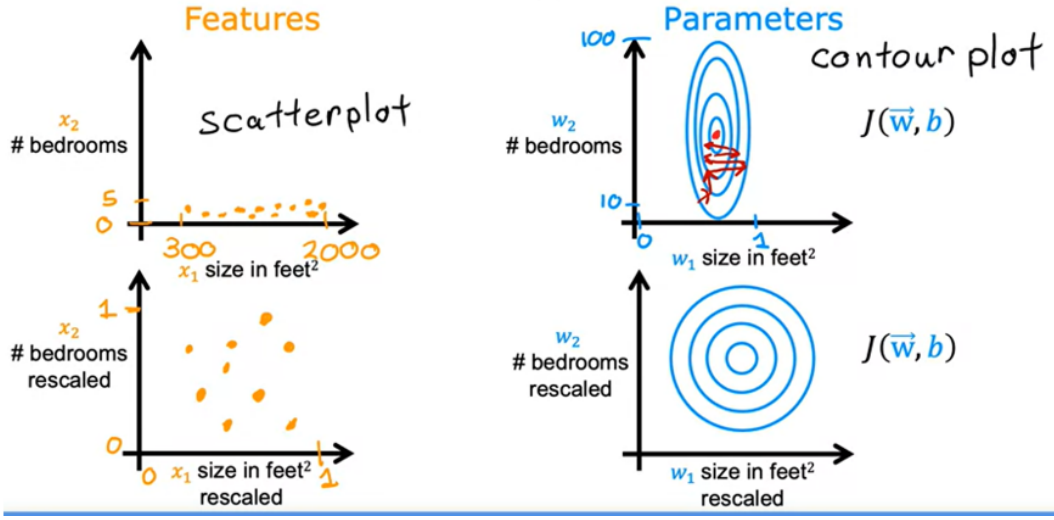
$\widehat{price} = 0.1 * 2000k + 50 * 5 + 50$ (200K, 250K, 50K)

$\widehat{price} = \$500$k more reasonable

Feature size and parameter size

	size of feature x_j	size of parameter w_j
size in feet^2	⟷	↔
#bedrooms	↔	⟷

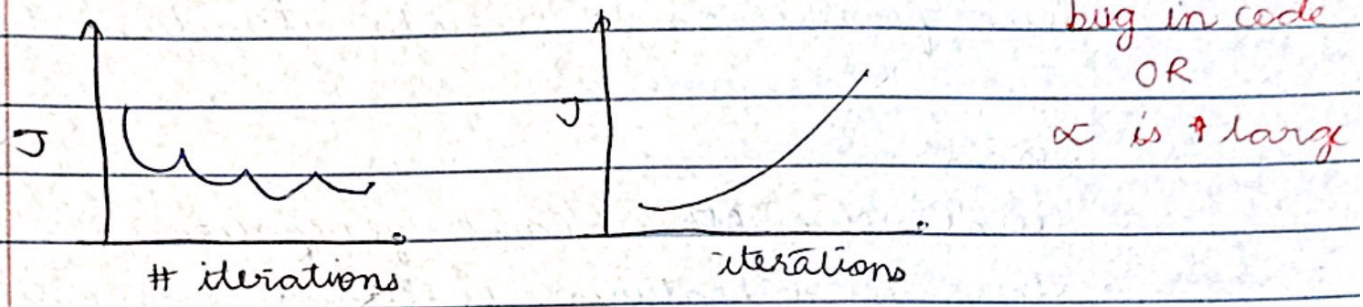
Feature size and gradient descent

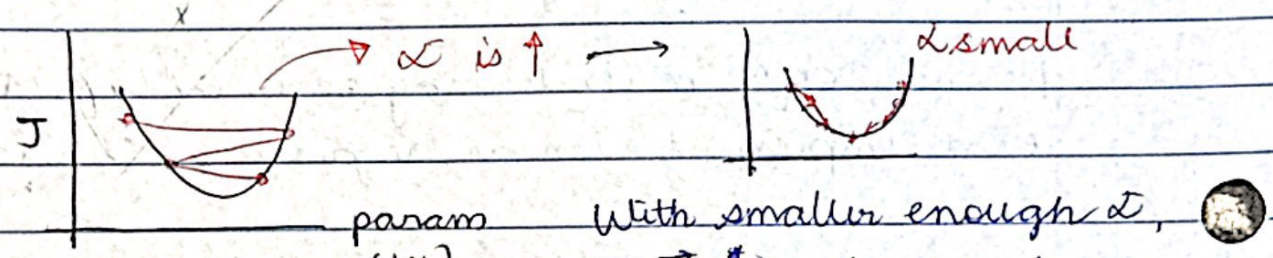


J(w,b) i.e cost should always decrease in every iteration

Choosing good learning rate

Problems with GD:





With smaller enough α, $J(\vec{w}, b)$ should ↓ on every iteration

Always do:
Keep α very small & check if $J(\vec{w},b)$ [cost] is decreasing in every iteration. If not, there's a bug in code

try: (0.001 0.01 0.1 1)

although these aren't best & may take lots of iteration, but can tell us about buggy code

∴ For these α, run for some iterats and plot learning curve to check for improvement

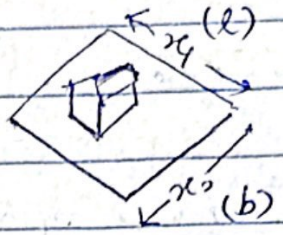
— — IMP — LAB — IMP — —

C1W2L3: Feature scaling and learning Rate

Feature Engineering

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

(x_1: length, w_1: length; x_2: breadth, w_2: breadth)



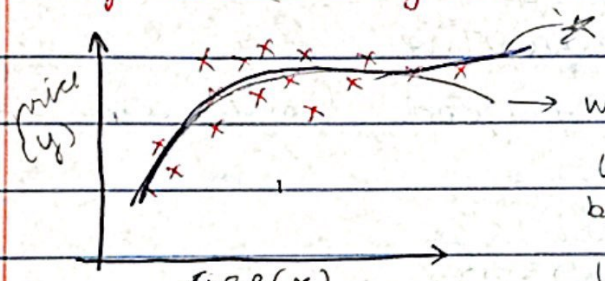
area = $l \times b$

$\therefore x_3 = x_1 x_2$ (new feature)

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

Defi: Using intuition to design new features by transforming or combining original features

Polynomial Regression



→ we might think quadratic

$w_1 x + w_2 x^2 + b$

but, quadratic goes down (1)

which isn't possible (size of house ↑ but price ↓)

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

— × — LAB — ^ —

—× — WEEK 1 DONE — × —

WEEK 3 — Classification

Que	Ans (y)	
Is email spam	no/0	yes/1
Is transact° fradulant	no/0	yes/1
Is tumor malignant	no/0	yes/1

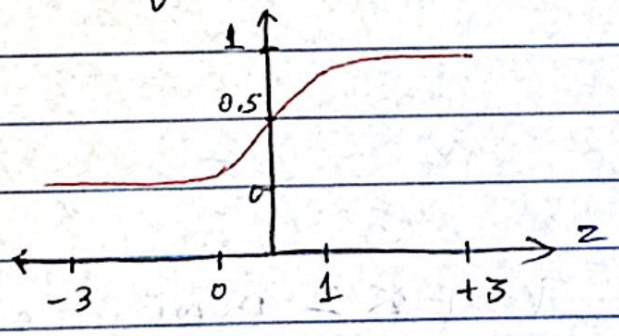
Here, 'y' can have only 2 values { yes, no } — BINARY CLASSIFICATION
2 possible classes = categories

False	True
0	1
negative class ≠ bad	positive class ≠ good (but ans to question)

Logistic Regression

It is a classification algorithm

Sigmoid function



sigmoid f(n)
logistic function
O/P b/w 0 and 1

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

where
$0 < g(z) < 1$
$e = 2.7$

If we take $f_{\vec{w},b}(\vec{x})$

$$z = \vec{w} \cdot \vec{x} + b$$

$$\downarrow$$

$$z$$

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

∴ for logistic regression

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

Interpretation of logistic regression O/P

= Probability that class is one '1'

Example

x is 'tumor size'

y is 0 (not malignant)
or 1 (malignant)

$f_{wb}(\vec{x}) = 0.7$ → this means 70% chance that y is 1 (malig. tumor)

then $f_{wb}[(\vec{x}) \& y=0] = 0.3$

↳ 0.7+0.3=1

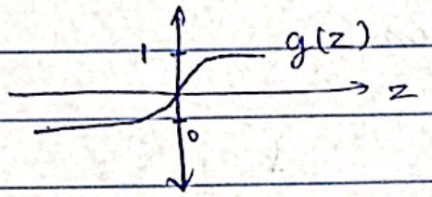
Bayes's theorem

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

→ Probability that $y=1$ given I/P $\vec{x}$ and parameters $\vec{w}$ and b

Decision Boundary

LoR Recap



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

$$z = \vec{w} \cdot \vec{x} + b$$

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

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

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

0.7 / $1 - 0.7 = 0.3$

0 or 1

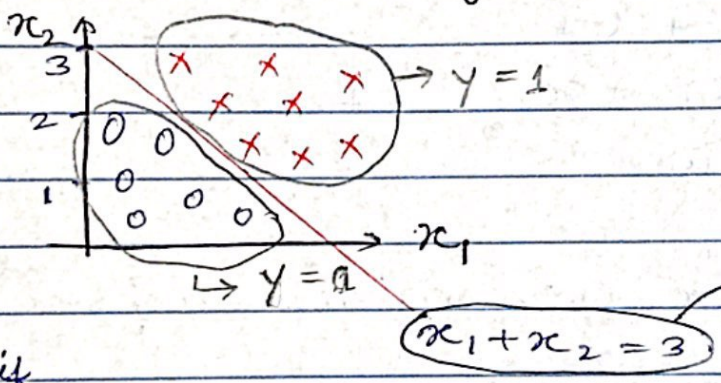
if $f_{\vec{w}b}(\vec{x}) \geq 0.5$
$\hat{y} = 1$
else
$\hat{y} = 0$

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

this means when is $g(z) \geq 0.5$
when (by graph) $z \geq 0$

ie $\vec{w}x + b \geq 0$ then $\hat{y} = 1$
if $\vec{w}x + b < 0$ then $\hat{y} = 0$

Decision boundary



if
$f_{\vec{w},b}(\vec{x}) = g(z) = g(w_1x_1 + w_2x_2 + b)$
↳ let $w_1 = 1$ & $w_2 = 1$ & $b = -3$

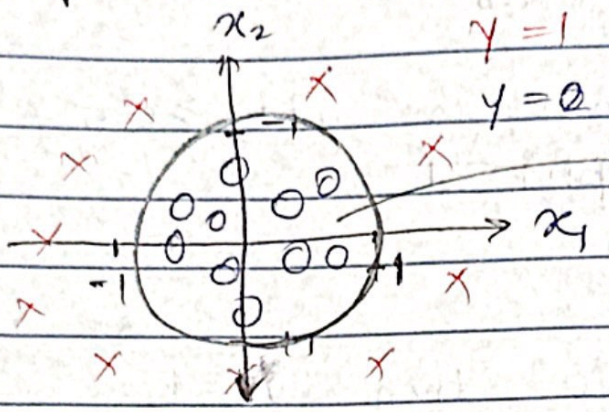
$\therefore z = x_1 + x_2 + b$

Decision boundary: $z = 0$

$\therefore x_1 + x_2 - 3 = 0$

$x_1 + x_2 = 3$

Complex example



z = polynomial

$$f_{\vec{w},b}(\vec{x}) = g(z) = \underbrace{w_1 x_1^2 + w_2 x_2^2 + b}_{z}$$

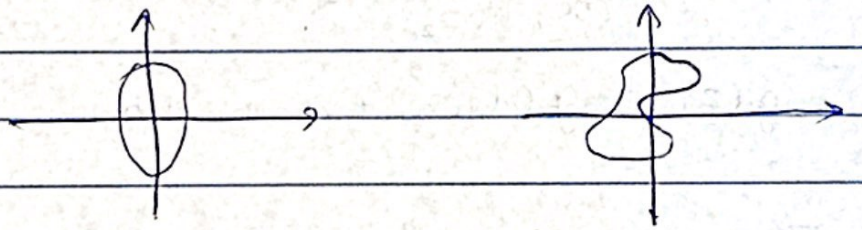
$$1 \qquad 1 \qquad -1$$

$$z = x_1^2 + x_2^2 + (-1)$$

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

Equate of circle

We can have even more complex DB.



Cost Function (Log. Regression)

Cost function gives us a way to measure how well the parameters fits the training data, giving a way to choose better parameters

Training set

	tumor size	.....	patients age	malignant
	x_1	.....	x_n	y
$i=1$	10		52	1
$i=2$	2		73	0
	⋮		⋮	0
$i=m$	⋮		⋮	⋮

$i = 1, 2 \ldots m$ (training examples)

$j = 1, 2 \ldots n$ (features).

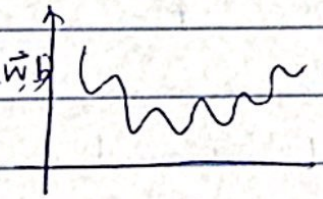
target y is 0 or 1

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

How to chose w & b?

J ∪ $\vec{w}, b$

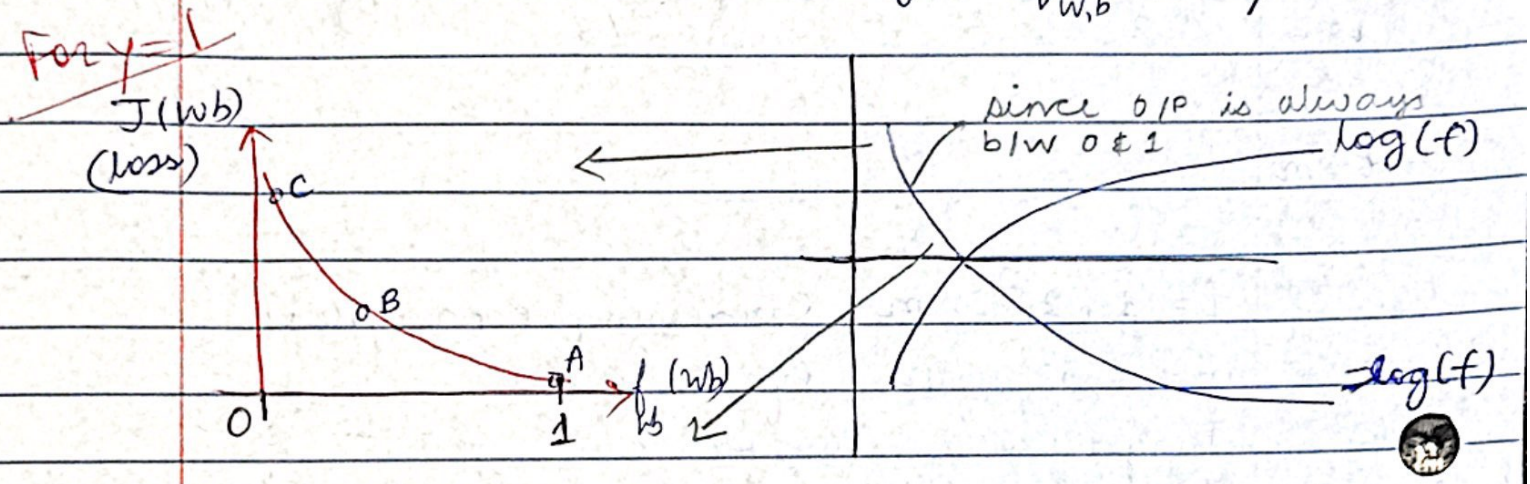
If we plot cost, as in linear reg then we get non-convex curve with multiple local minimas



∴ Squared Error Cost fn isn't a good choice

Logistic loss function

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



when $y^{(i)} = 1$

at point A: predicted value = $f_{\vec{w},b}(\vec{x}^{(i)}) \to 1$

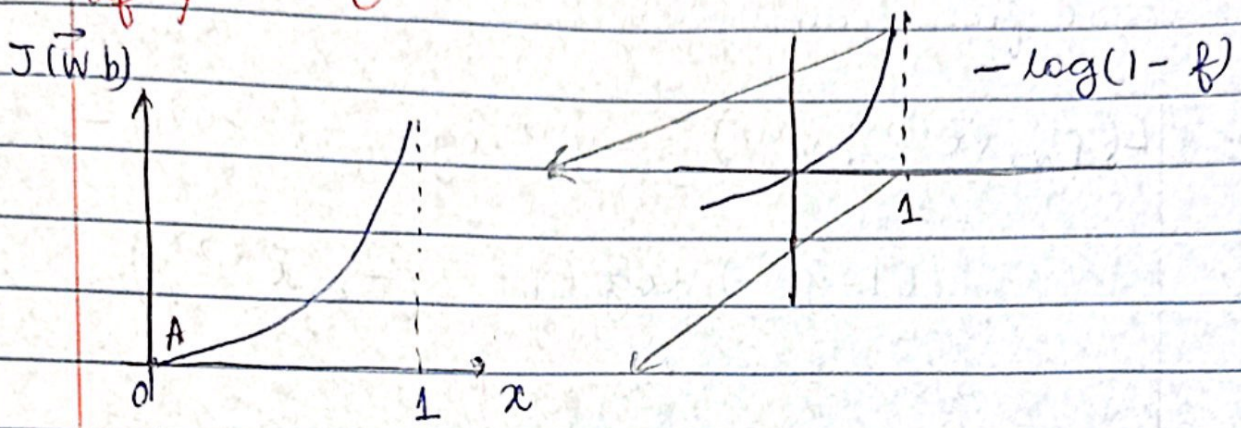
$\therefore$ loss $\to 0$ ☺

at point B: if $f_{\vec{w},b}(\vec{x}^{(i)}) = 0.5$

ie its says 50% chances, but it is really malig then we see at B, loss is ↑

at C, if it says 0.1, ie 10% chance but actually it is malignant, then we see at A, loss is really ↑.

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



point A Similar to $y=1$, if $f_{wb}(\vec{x}) \to 0$, loss $\to 0$

Cost =

$$J(\vec{w},b) = \frac{1}{m}\sum_{i=1}^{m} L\underbrace{\left(f_{wb}(x^{(i)}) - y^{(i)}\right)^2}_{\text{loss (previous page)}}$$

∴ Find w, b to ↓↓ cost

Simplified Cost Function:

$$L(f_{\vec{w}b}(x^{(i)}), y^{(i)})^2 =$$

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

→ Just a combined form of loss fun of on previous (-2) page

Cost function: → convex only 1 minima

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

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

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

Although there are so many cost fcn, we choose this because of maximum likelihood

Implementing GD:

Training LR:

① Find $\vec{w}$ and b

② Then give model $\vec{x}$ = age, size of tum & other param

③ Model predict

$f_{\vec{w},b}(\vec{x}) = \frac{1}{1+e^{-(\vec{w}\vec{x}+b)}}$ $P(y=1)$ & $P(y=0)$

For implementation, we repeat until loss is $\approx$ with simultaneous update of w and b.

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

repeat {

$$w_j = w_j - \alpha \frac{\partial}{\partial w_j} J(\vec{w}, b) \quad \text{for } j = 1 \text{ --- } n \rightarrow \text{for all features}$$

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

}

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

($x_j^{(i)}$: j^{th} feature of i^{th} train example)

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

$\rightarrow$

Equat^n on previous page might look exactly similar to ~~too~~ linear R. But, funct^n f is different in both

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

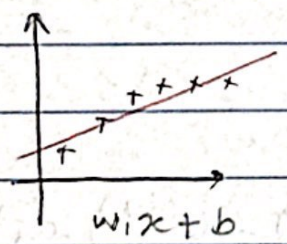
Logistic " $f_{\vec{w}b}(\vec{x}) = \frac{1}{1 + e^{(-\vec{w}\vec{x} + b)}}$

Same concepts

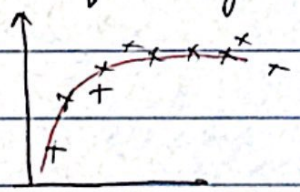
- Learning curve monitoring
- Vectorized implem
- Feature scaling (to speed up GD)

The Problem of Overfitting

REGRESSION

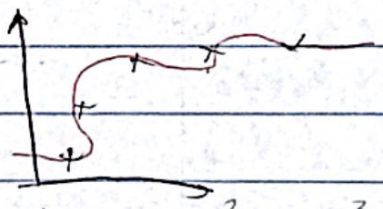


doesn't fit well
Underfitting
↑ bias



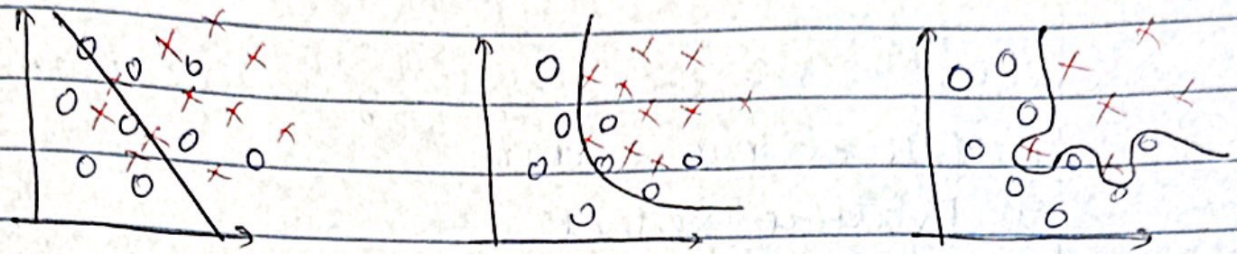
fits pritty well
Generalization
'just right'

GOAL



fits extremely well
Overfitting
↑ variance
error = 0

CLASSIFICATION



Underfit | just right | Over fit

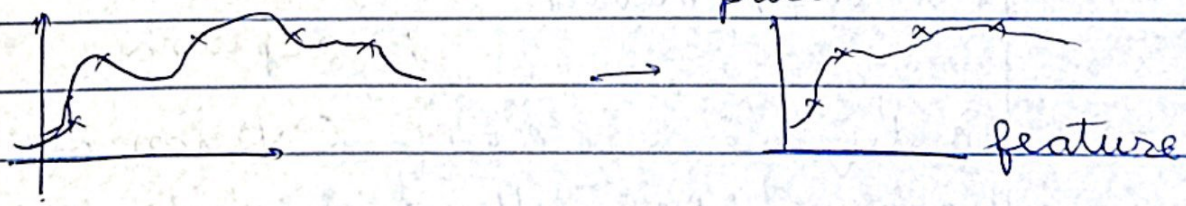
Adressing Overfitting

① Collect mor training data

② Check if we can use fewer features. A greater # features will ↑ training examples causes overfitting. To avoid, use feature select to select subset of most relevant features

③ Regularization : When we have (suppose 100) features, and all are IMP (cant eliminate any). In this case, instead of just removing (& being harsh) we ↓ the value of paramenter. (reduce impact of that feature gently).

param =0



$f(x) = 28x + 385x^2 + 39x^3 + 174x^4 + 100$

elimination means setting $w = 0$ instead of 174 (↑ value)

reg →

$f(x) = 12x + 0.23x^2 + 0.00014x^3 + 0.00001x^4 + 10$

→ smaller values of w_j

Recap - Addressing regularization.

1. Collect more data
2. Feature Select°
3. Regularization (↓ size of parameters)

Cost Function with Regularization

If we have $w_1, w_2 \cdots w_{100}$ parameters, we dont know which param is imp & which isn't. So we penalize every param by:

↳ gently mild them

reg. term

$$J(\vec{w}, b) = \frac{1}{2m} \sum_{i=1}^{m} \left(f_{wb}(x^{(i)}) - y^{(i)} \right)^2 + \frac{\lambda}{2m} \sum_{j=1}^{n} w_j^2$$

only to make sure both these terms are scaled by $1/2m$

λ = regularizat° param

some might have $\uparrow + \frac{\lambda}{2m} b^2$ (it doesn't matter)

↳ optional

λ defines Tradeoff (option of how one balance b/w fitting of data curve to keeping w_j smaller).

if $\lambda = 0$ (↓ it will overfit)
if λ is ↑↑↑ (it will underfit)
if λ = justright (it gives generalizat°)

Regularized Linear Regression

$$\min_{\vec{w},b} J(\vec{w},b) = \min_{\vec{w},b} \left[\underbrace{\frac{1}{2m} \sum_{i=1}^{m} \left(f_{wb}(\vec{x}^{(i)}) - y^{(i)} \right)^2}_{\text{orig cost fcn}} + \frac{\lambda}{2m} \sum_{j=1}^{n} w_j^2 \right]$$

↳ Reg. term

∴ Now GD algo remains same but $\frac{\partial}{\partial w_j} J()$ changes and $\frac{\partial}{\partial b} J()$ remains same (bcz we dont reg. b)

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

Print out

Regularized Logistic Reg

similar to above for logistic as well

$$J(\vec{w},b) = \min_{\vec{w}\, b} \sum_{i=1}^{m} \underbrace{(\cdots)}_{\text{cost fcn}} + \frac{\lambda}{2m} \sum_{j=1}^{n} w_j^2$$

& the change in $\frac{\partial}{\partial w_j} J(\vec{w},b)$ is

$$\Rightarrow \frac{1}{m} \sum_{i=1}^{m} \left[\left(f_{wb}(\vec{x}^{(i)}) - y^{(i)} \right) x_j^{(i)} \right] + \frac{\lambda}{m} w_j$$

↑ logistic fcn

Urvi Jain