Supervised Machine learning? → ARJUN ADHIKARI

Memo Date:

Machine learning.

How machine recognize?

→ Start.

What is:

Supervised learning.?

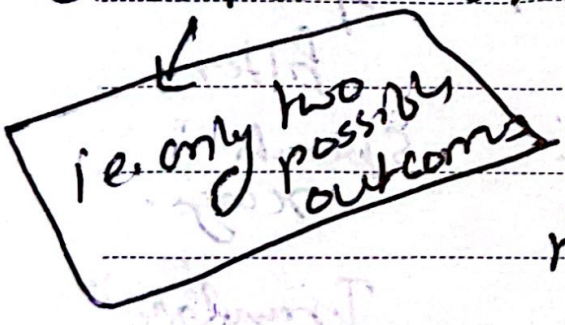
→ Algorithm to learn x to y.

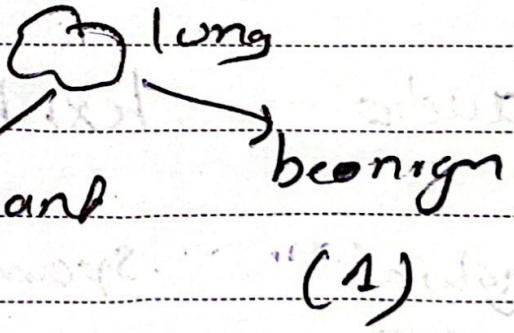
Input (x)	Output (y)	Applicam
Email	Spam? (0/1)	Spam filters
audio	text transcrip	Speech recog.
English	Spanish	Translation
ad, user info	Click (0/1)	online advertising
image, radar info	position of other cars.	self-driving car
image of phone	defect? (0/1).	visual inspection

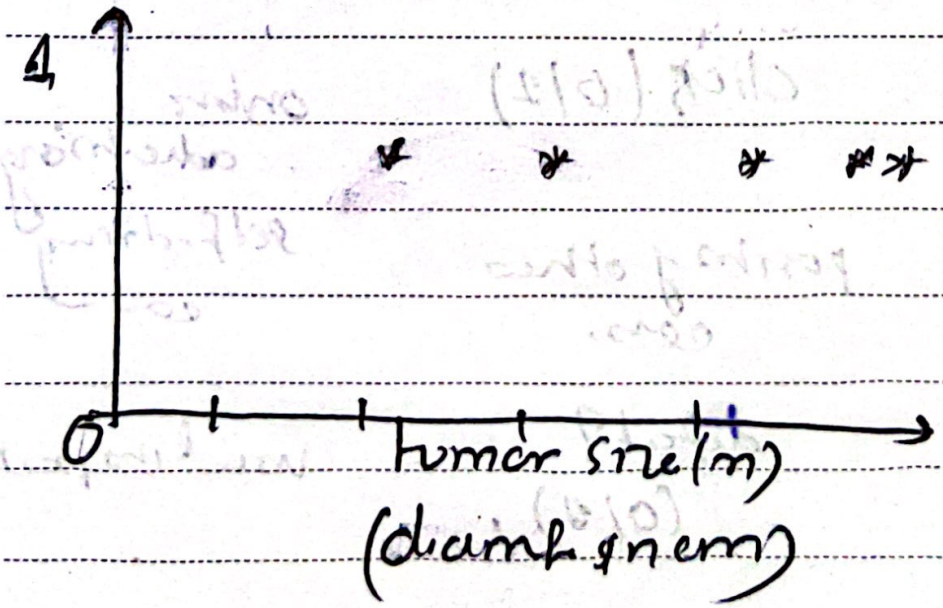
Memo Date: . .

Regression: House price prediction.

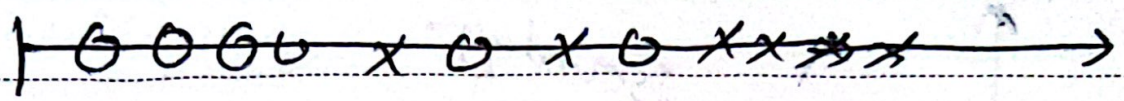
Classification: Breast cancer detection.







Memo Date: . .



O- benign
X Malign

classifier :

predict word.

↳ small number of possible outcomes
(0, 1, 2 . . only)

predict category/class

↳ ~~ref~~ can be word than number

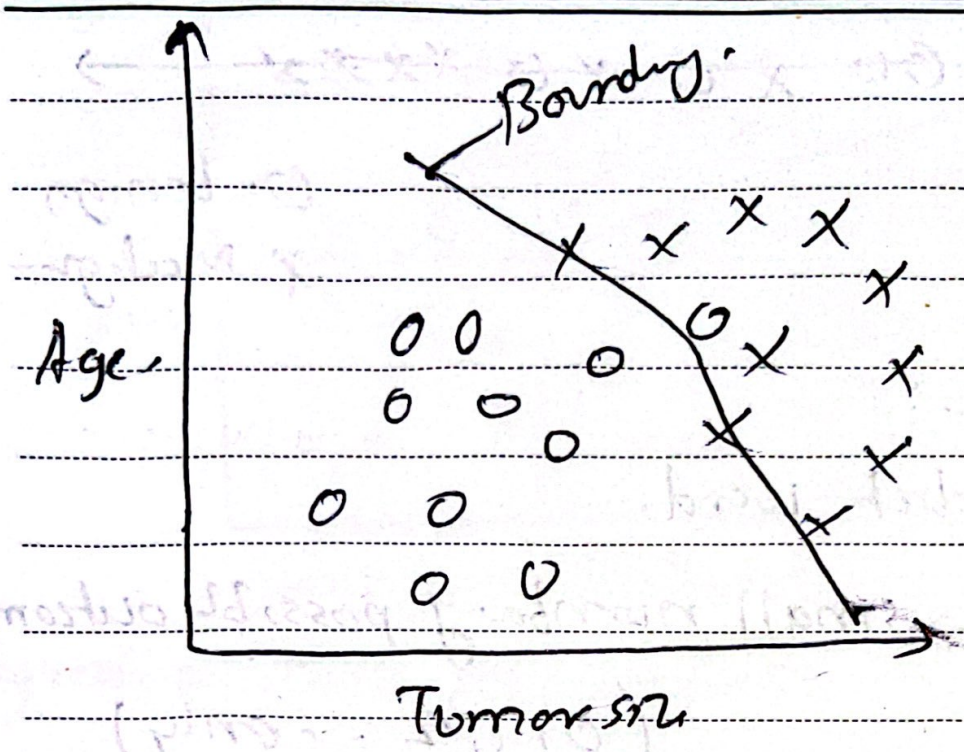
Regression: cost, . .

Regression

whether pictures Cat or dog ?

Malignant or benign ?

Memo Date: . .



Tu

Supervised learning

learn from data given right answers

Regression	Classification
predict a number	predict categories.
Infinitely many possible outcomes.	Small number

Memo Date: . .

Unsupervised learning

- clustering
- Dimensionality reduction
- Anamaly detection.

→ Data only comes with inputs x, but not output labels y.

* Algorithm has to find structure in data.

clustering

Groups similar data points together

Anomaly detectn.

Find unusual data points.

Dimensionality reductn.

Compress data using fewer numbers.

Memo Date: . .

Linear regression Model

predict number.

Train and Model. (by using data)

Terminology:

(1) → Training set

$(x^{(i)}, y^{(i)})$ = i^{th} training example

(2) learning algorithm.

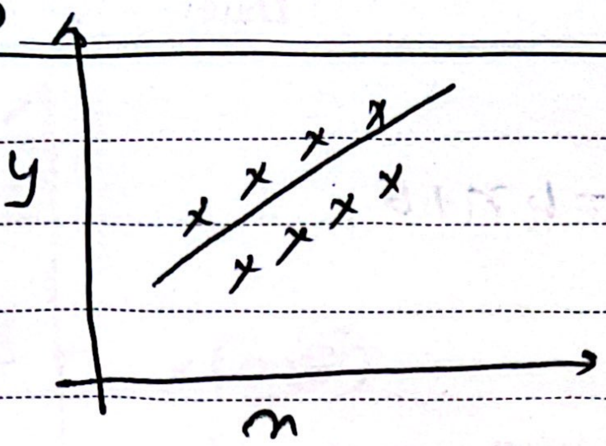
produce

f → take new input x and output (prediction) $\hat{y}$

Estimate / prediction for model

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

Memo Date: . .



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

linear regression with one variable.

Univariate linear regression

Cost function:

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

m = number of training examples

Squared error cost function

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

For multiple,

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

Scanned with CamScanner

Memo Date: . .

model:

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

parameters:

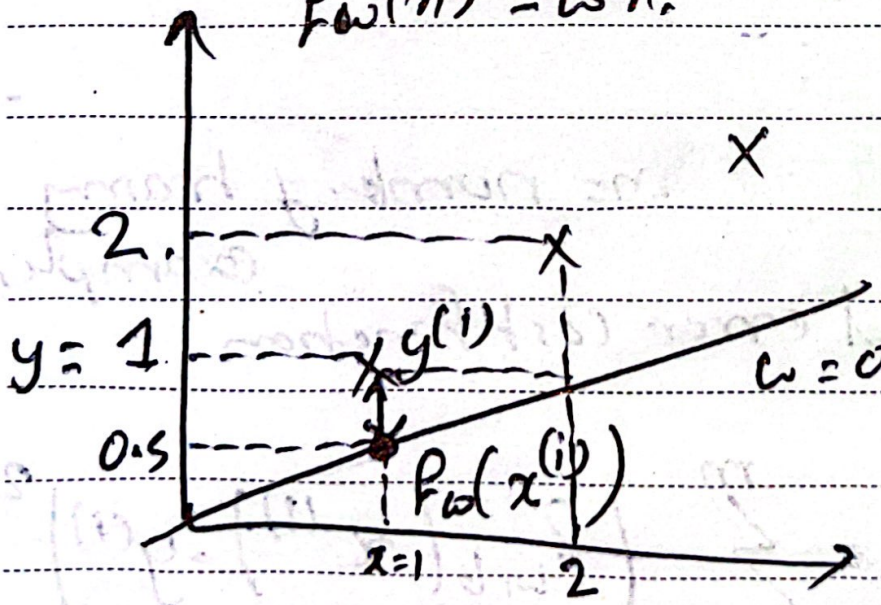
w, b.

Cost function:

Model pred - Actual value

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

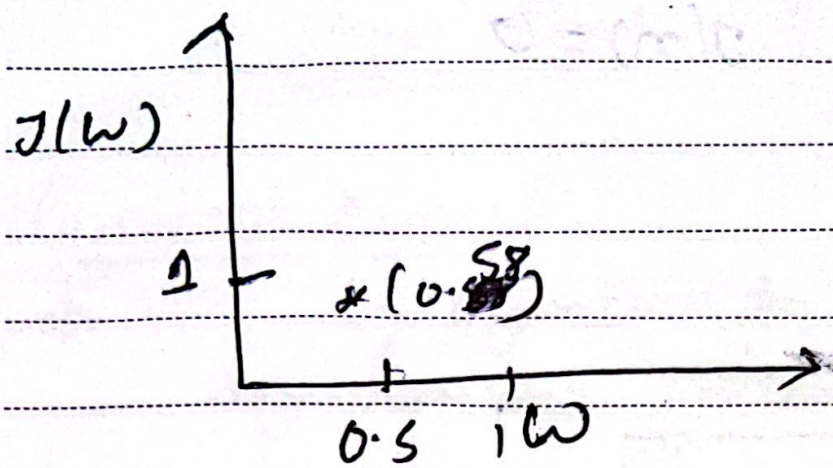
$f_w(x) = wx$.



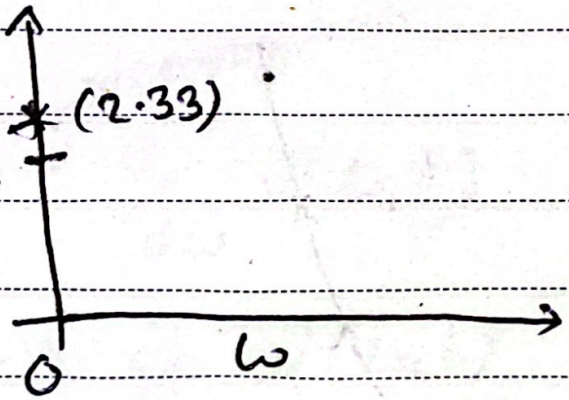
$$J(0.5) = \frac{1}{2m}\left[(0.5-1)^2 + (1-2)^2 + \dots\right]$$

$$= 0.58$$

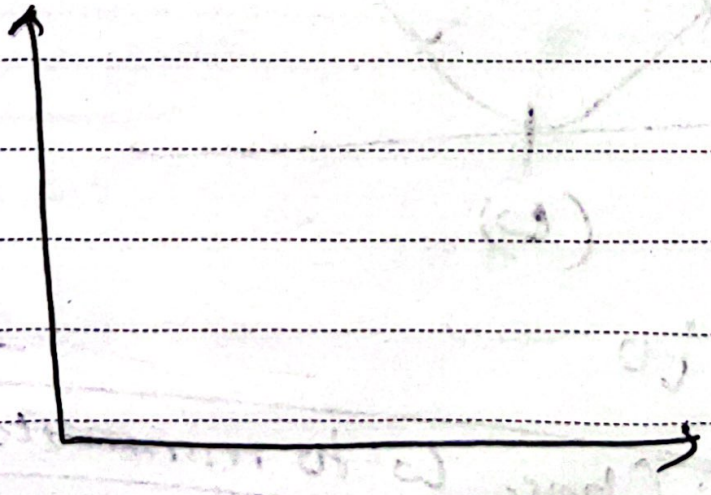
Memo Date:



When w = 0

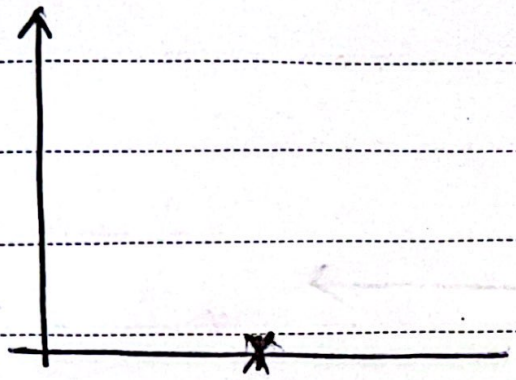


When w = -0.5π

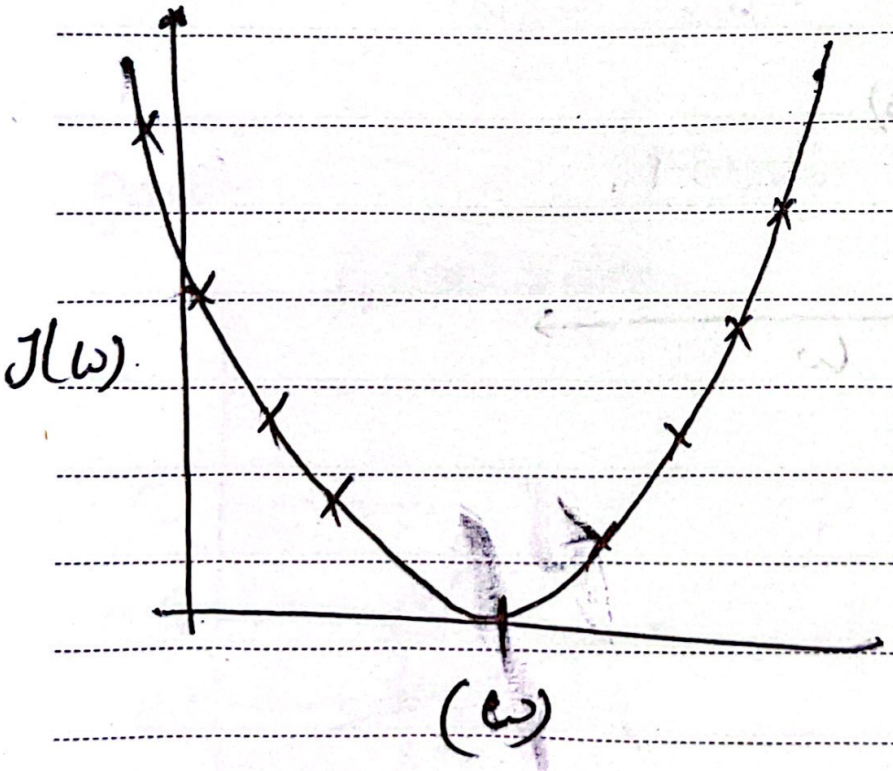


Memo Date: . .

When $w=1$ $J(w)=0$



Thus



loss function.

Thus goal is to

Choose w to minimize J(w)

Memo Date: . .

\# Shape of bowl (For two parameters)

Gradient Descent

\# ↳ Algorithms to minimize any function including cost function.

~~w = w - α~~

↑ Direction, कहाँ छोटा जाना है Idea.

learning rate

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

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

Simultaneous update

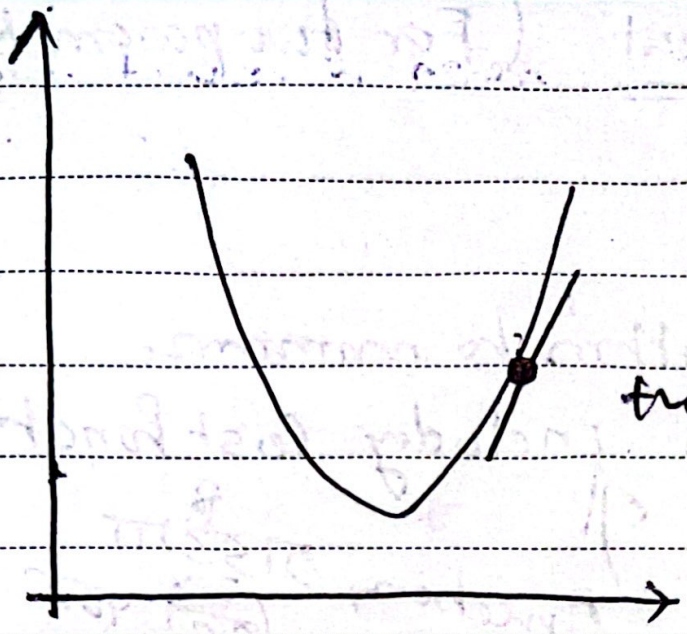
tmp_w =

temp_b

w = temp_w

b = tmp_b

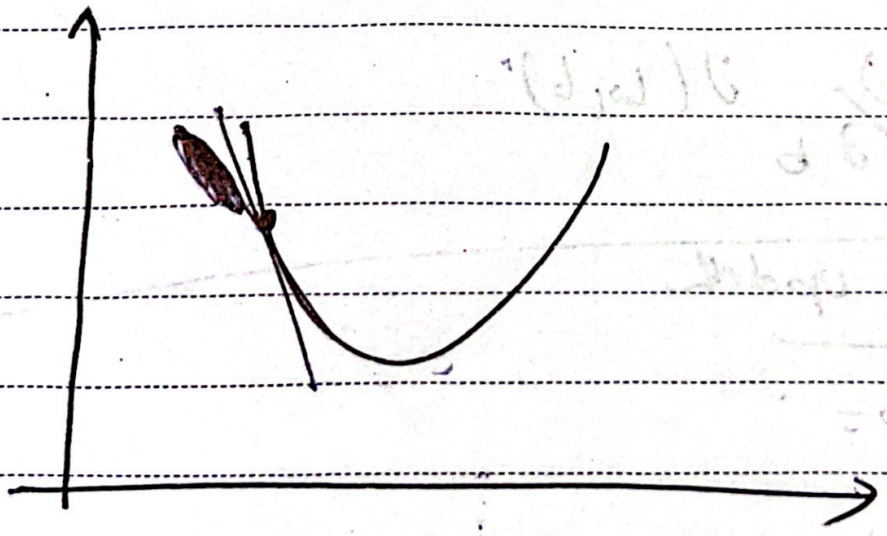
Memo Date: . .



$\Rightarrow \omega = \omega - \alpha$ (-ve)

sine the slop

= decrease



$\omega = \omega - \alpha$

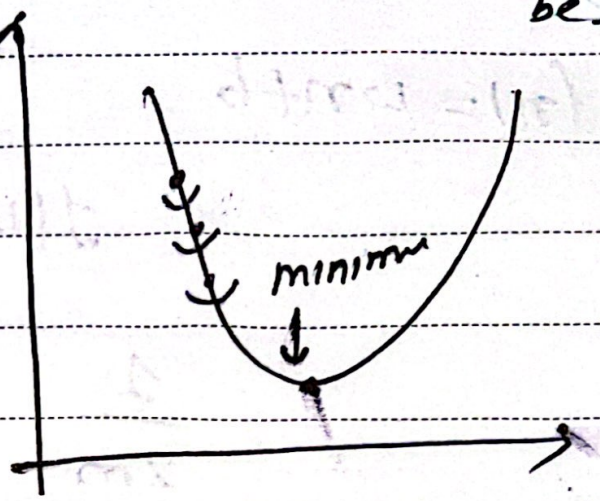
sine -ve slop

$= \omega + \alpha$

= increasy

Memo Date: . .

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



If large fail to converge.

We are

At If's at minimum,

$\alpha = 0$

$\therefore w = w$.

Training Linear Regression.

Gradient Descent for Linear Regression

Memo Date: . .

linear regression model.

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

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 {

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

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

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

Memo Date: . .

"Batch" gradient descent.

↳ Each step of gradient descent uses all training examples.

Week-2

previously, ~~fffn~~ single variable

But now,

~~Multiple variables~~ Multiple variable.

~~Size in feet²~~

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

↑ size ↑ rooms.

$\vec{w} = [w_1\ w_2\ w_3 \cdots w_n]$ → n numbers parameters of model

vector.

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

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

Memo Date: . .

This is
multiple linear regression

Vectorization Part-1.

$\vec{w} = [w_1, w_2, w_3]$ $n=3$

b is number

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

Numpy.

```
w = np.array([1.0, 2.5, -3.3])
b = 4
x = np.array([10, 20, 30])
```

Vectorization.

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

Unvectorized

↳ For j in range(0, n)

```
f = f + w[j] * x[j]
```

Memo Date: . .

Vector

notation: x (bold)

np.array([1,2,3])

```
a = np.arange(10)
c = a[2:7:1]
print("a[2:7:1] =", c)
```

start stop end

↓

2, 3, 4, 5, 6

a[3:] = 3, 4, 5 - -

a[:3] = 0 1 2

a[:] = all.

np.sum(a) => sum all.

Memo Date: . .

Fn wth.

```
def __( ):
    retur
```

```
a = n'p.array ([ 1)
b = np.array ([1])
c = np.dot (a,b).
```

$$a.b = \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \end{bmatrix} = a_0 b_0 + a_1 b_1 + \cdots$$

np.zeros(4) => 0. 0. 0. 0.

np.zeros((4,)) => "

a.shape => (4,)

np.zeros((1,5)) => 0. 0. 0. 0. 0.

a.shape = (1, 5)

But,

np.array([[5], [4], [3]])

↑ a = [[5]
[4]
[3]]

Memo Date: . .

Creating Matrix

#reshape

a = np.arange(6).reshape(3,2)

a.reshape: (3,2)

a = [[0 1]
[2 3]
[4 5]]

np.reshape(a, (2,3))

or

np.arange(6).reshape(-2,2)

P(-1) says, given column and no. of array, calculate yourself.

Memo Date:

Slicing.

a = np.arange(20).reshape(-1,10)

a[0, 2:7:2] ⇒ [2 3 4 5 6]

Array are seperated by commas

array = np.array([[1,2,3], [4,5,6], [7,8,9]])

array.shape ⇒ (3,3)

~~np.shape~~

a.shape → no of elements for 1D array

→ row/column for 2D array

Gradient descent for Multiple Regression

Memo Date: . .

For n, features

Cost Function: → $J(\vec{w}, b)$ $[w_1, w_2, w_3]$

Gradient descent

repeat {

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

⋮

$$w_n = w_n \ldots$$

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

Memo Date:

X_vec = X_train[0, :]

0 row 전체 element

X_vec value : [_ _ _ _]

Gradient w, b

new value 구하기

Feature and parameter value

#

' ' '

Memo Date: . .

Feature scaling is a method to scale numeric features in the same scale or range (like -1 to 2, 0 to 2)

Also called Data normalization.

Apply on independent variable

" " train data and transform train and test.

Why?

If height 140 cm and 8.2 feet

→ ML algorithm understand only numbers but not feature unit

ie Acc to ML, 140 > 8.2

#

Memo Date: . .

$price = w_1 x_1 + w_2 x_2 + b$

↓ ↓

size bedrooms.

x_1 : size (feet^2) x_2 = bedrooms.

range : 300 - 2,000 range 0-5.

(large)

Trans

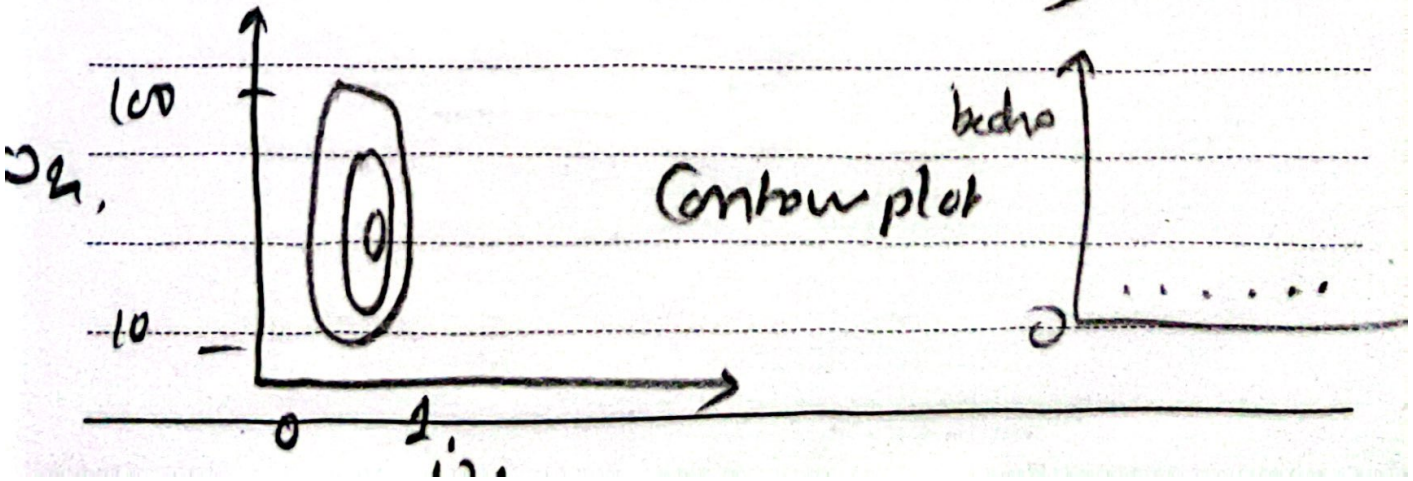
~~Archive~~: $x_1 = 2000$, $x_2 = 5$, price = \$500K.

Size of parameter w_1, w_2?

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

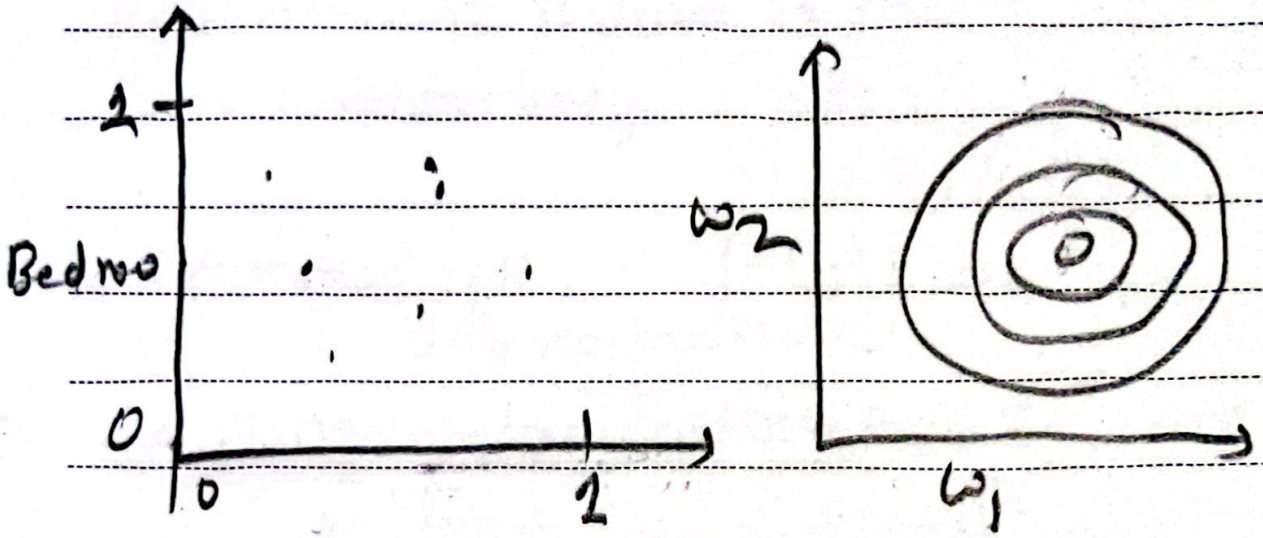
$price = 0.1 \times 2000 + 50 \times 5 + 50$

$= 500K$ (reasonable)



Memo Date: . .

Bad rescaling.

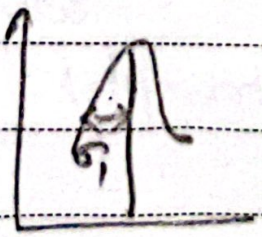


This rescaling helps in decreasing gradient descent. run fastly.

Rescaling.

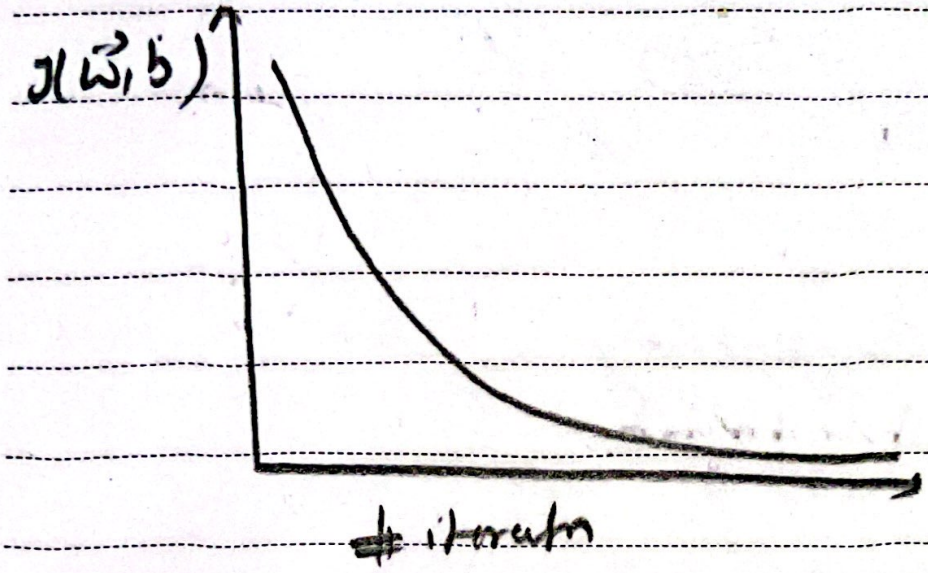
1. Dividing by max
2. Mean normalizn.
3. Z-score normalizatn.

$$x_1 = \frac{x_1 - \mu_1}{\sigma_1}$$



Memo Date: . .

learning curve.



Automatic convergence test.

If $\varepsilon = 0.001$

and

$J(\vec{w}, b)$ decreases by $\leq \varepsilon$ in

1 iteration ⇒ look into graph.

Memo

Value of α to try

0.001 0.003 0.01

x3 x

Feature Engineering:

$V = l \times b$

Polynomial regression.

Choice of features:

$f(\vec{w}, b | m) =$

Linear Regression using Scikit learn.

```
from sklearn.linear_model import
                    SGDRegressor
from sklearn.preprocessing import
                    StandardScaler
from lab_utils_multi import
                    load_house_data
```

Memo Date: . .

Classification

→ logistic Regression.

↳ 0 and 1

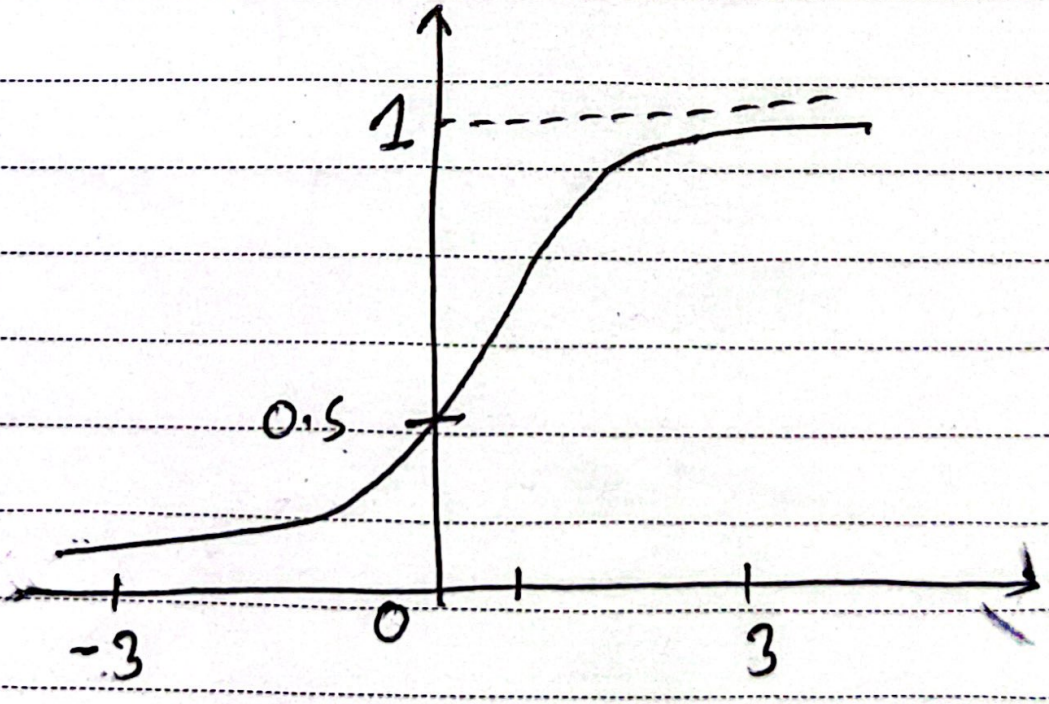
binary classification

Class ≡ category.

* Decision boundary.

Though regression, it is used for classification

Sigmoid function.



logistic function output between 0 and 1

Memo Date: . .

$$g(z) = \frac{1}{1+e^{-z}} \qquad 0 < g(z) < 1$$

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

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

↓

z

↓

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

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

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

" Logistic Regression. "

Memo Date: . .

$f_{w,b}(\vec{x}) = 0.7$

ie. Model is 70 % chance of

Numpy.

exp()

```
q = np.array([1, 2, 3])
np.exp(a)
```

g = 1/(1+np.exp(-z))

$$\frac{1}{1+e^{-z}}$$

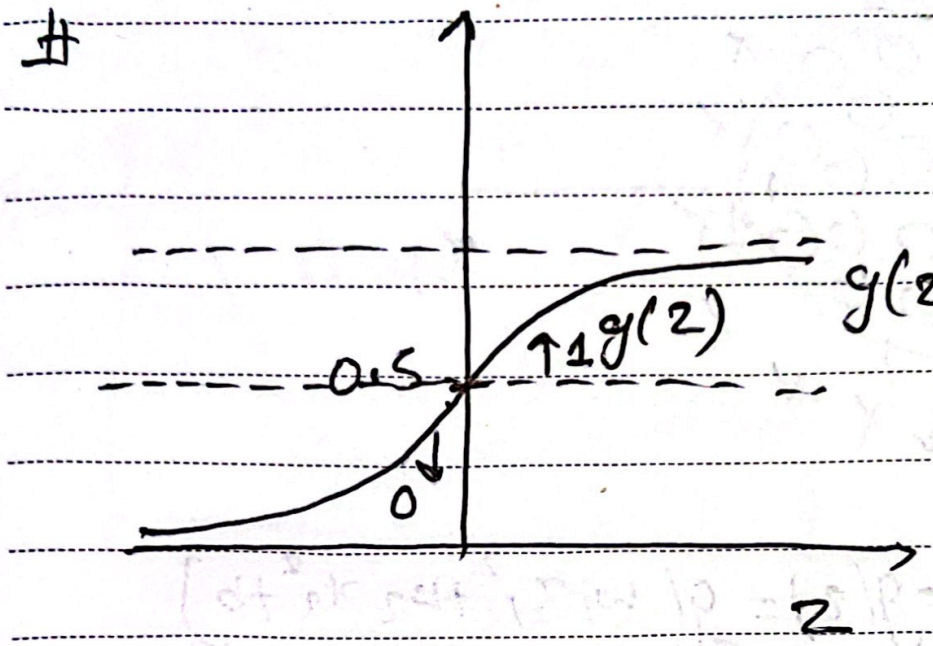
np.set_printoptions(precision=3)

```
np.c_[np.array([1,2,3]), np.array
                          ([4,5,6])]
```

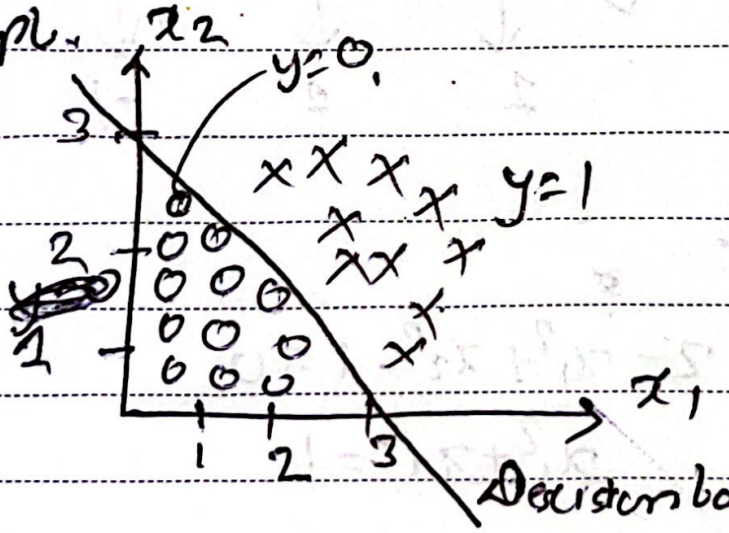
⇒ array([[1, 2],
[2, 5],
[3, 6]])

Memo Date: . .

draw_vthresh(ax, 0)



Example.



If, $f_{w,b}(\vec{x}) = g(z) = g(w_1x_1 + w_2x_2 + b)$

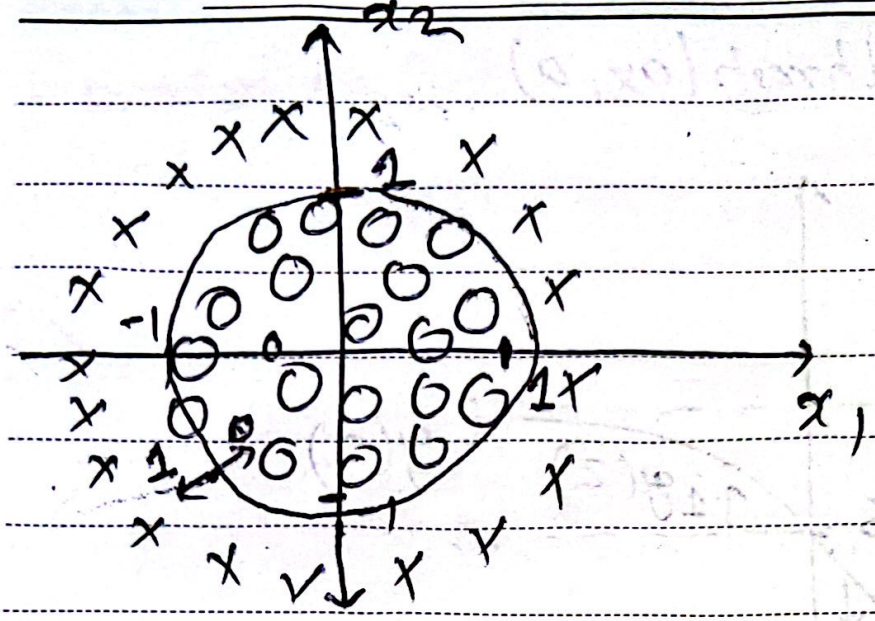
and if $w_1 = 1, w_2 = 1, -3$

$z = \vec{w}\vec{x} + b = 0$ } Decision boundary.

$z = x_1 + x_2 - 3 = 0$ (put in $w_1 = 1, w_2 = 1, b = -3$)

$x_1 + x_2 = 3$

Memo Date: . .



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

$$\downarrow \quad \downarrow \quad \downarrow$$

$$1 \quad 1 \quad -1$$

decision boundary:

$$z = x_1^2 + x_2^2 - 1 = 0$$

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

Squared Error Cost function

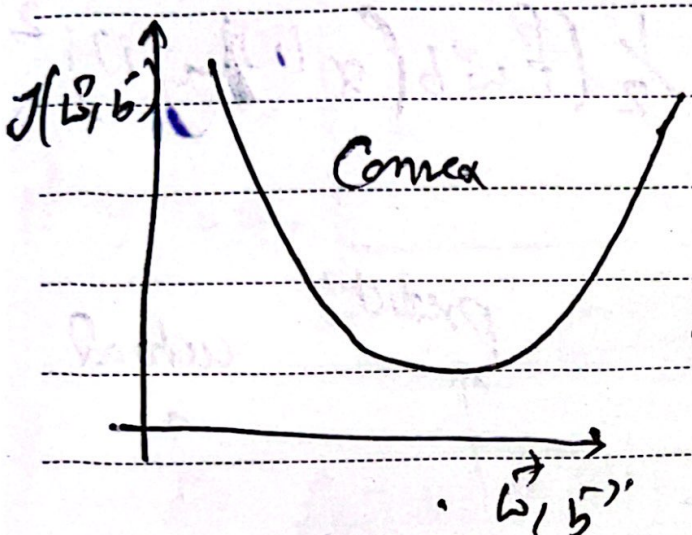
Will not be in logistic function.

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

Memo Date: . .

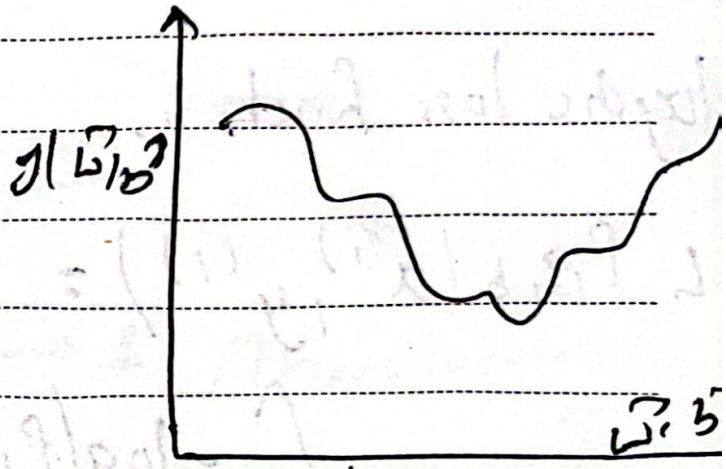
Linear regression

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



logistic regression

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



This is not easy curve so, we redefine cost function as.

#

Memo Date: . .

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

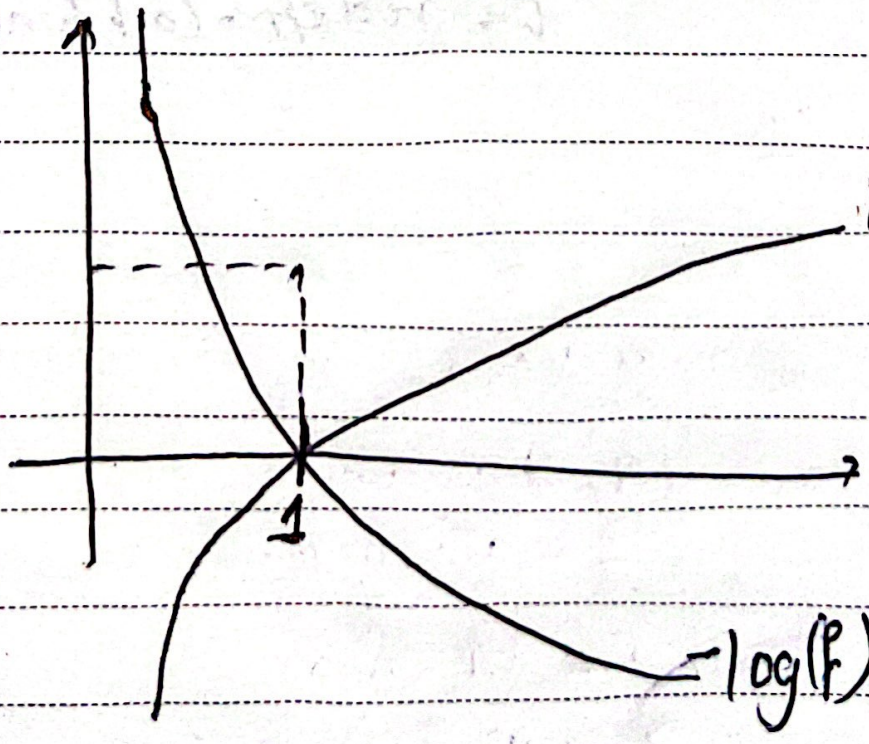
$\frac{1}{2}\left(f_{\vec{w}b}\left(x^{(i)}\right) - y^{(i)}\right)^2$

logistic loss function,

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

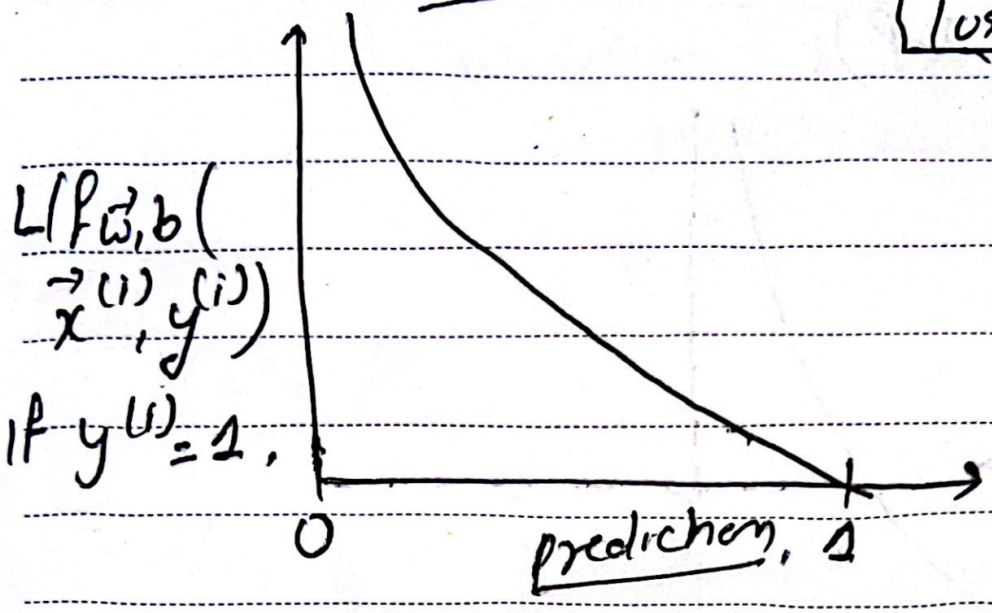
$$\begin{cases} -\log\left(f_{\vec{w},b}\left(\vec{x}^{(i)}\right)\right), & \text{if } y^{(i)}=1 \\ -\log\left(1 - f_{\vec{w},b}\left(\vec{x}^{(i)}\right)\right) & \text{if } y^{(i)}=0 \end{cases}$$

First part



Memo — First part — Date:

loss fn & cost fn



As $f_{\vec{w},b}(\vec{x}^{(i)}) \to 1$, Then loss $\to 0$.

↑ algorithm predicts true ↑ loss = 0

As $f_{\vec{w},b}(\vec{x}^{(i)}) \to 0$, loss $\to \infty$

↑ prediction

Th.

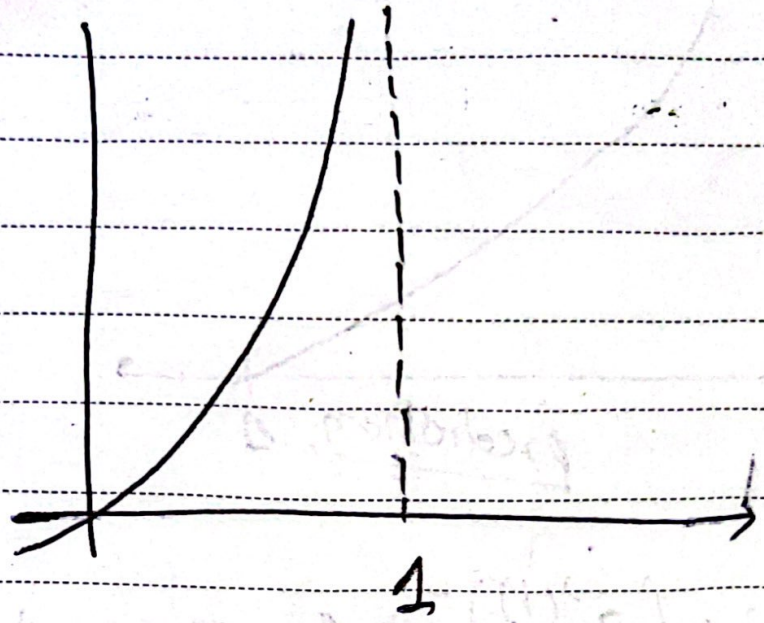
sing log 0 = ∞

If $y = 1$, and model predicts 0,
Then, loss function becomes infinite
i.e. cost fn is infinite.

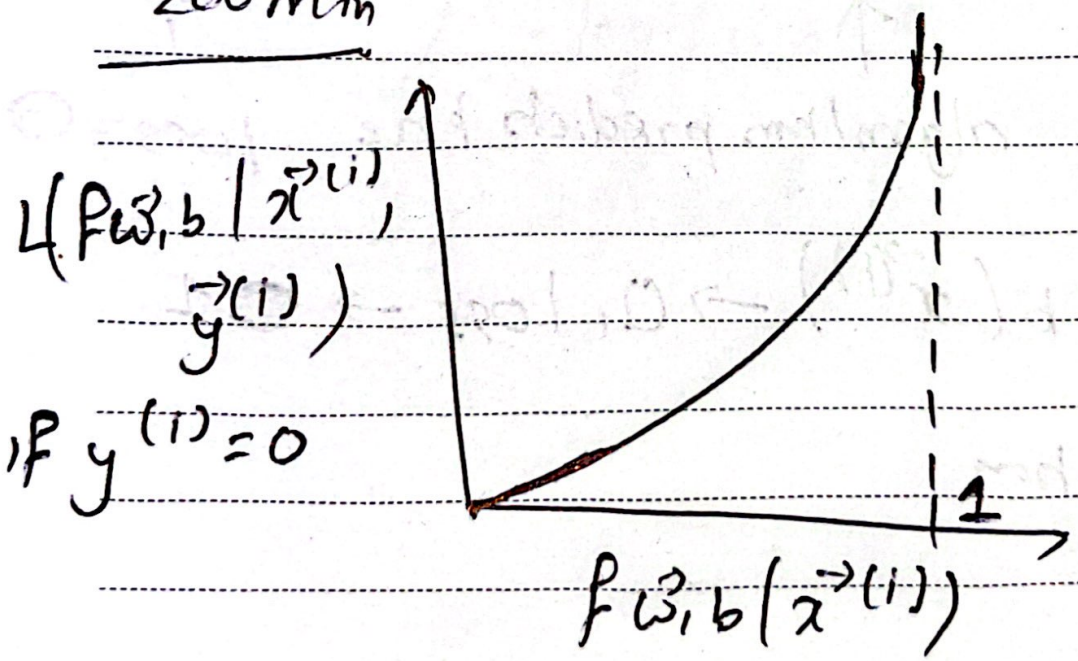
If $y = 1$ and model predicts 0.8,
loss fn becomes 0.0969
i.e. near to 0.

Memo Date: . .

$-\log(1-f)$



zoom in



ie. For actual value = 0,
if predictn = 1,
loss fn = infinity
and cost fn = ∞.

Memo Date: . .

But if prediction = 2, for actual val = 1,
loss fn = 0, so, cost fn = 0

Thus,

Cost fn :

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

making convex,

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

loss is measure of the difference of a single example to its target value while;
Cost is measure of losses over training set.

Memo Date: . .

Simplified loss fn:

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

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

Gradient Descent

repeat {

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

$$b = b - \alpha \left[\text{''} \right] \text{ no } x_j^{(i)}.$$

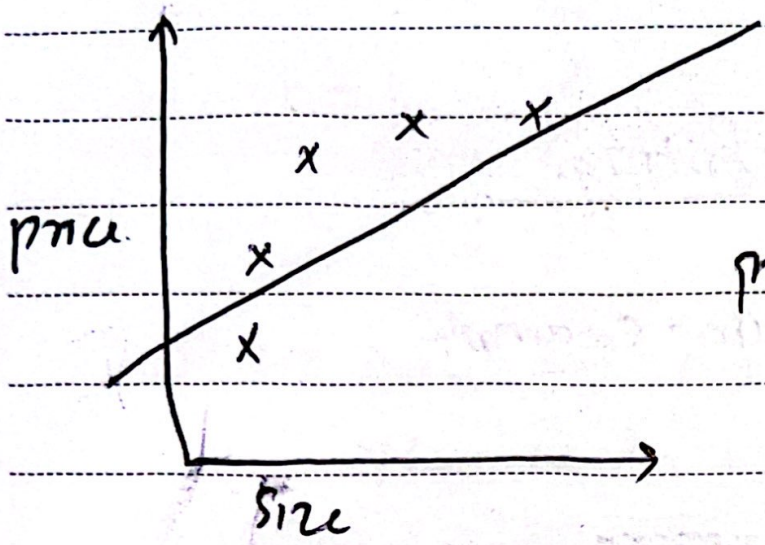
linear reg: $f_{\vec{w},b}(\vec{x}) = \vec{w} \cdot \vec{x} + b$

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

Memo Date:

The problem of overfitting.

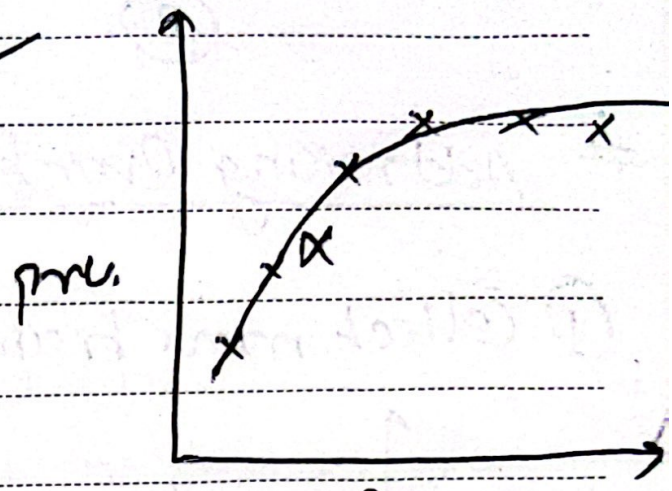
Just right



$w_1 x + b$

→ Underfit

High bias.

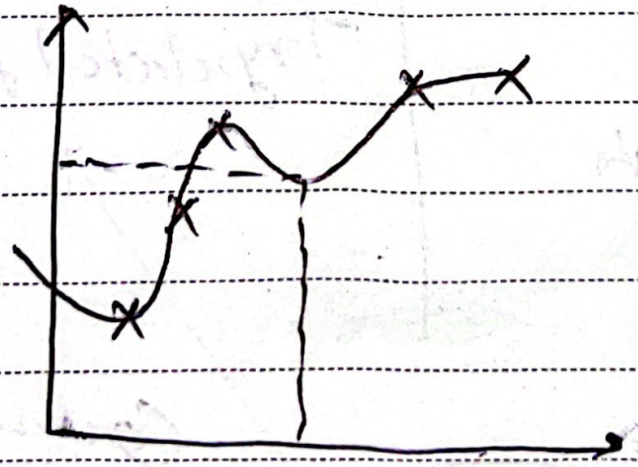


$w_1 x + w_2 x^2 + b$

Set well

generalization.

pretty well



$w_1 x + w_2 x^2 + w_3 x^3 + w_4 x^4 + b$

Overfit

High variance

fit ~~pretty~~ extremely well.

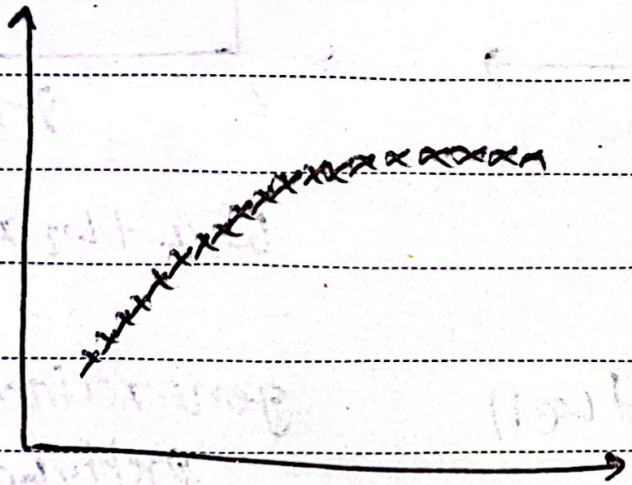
Memo Date: . .

⇒ neither high bias nor high variance

(2).

Addressing Overfitting.

① Collect more train example



② all feature + insufficient data ↓ overfit.

Try selected features.

Course 2

Regularization.

③. Reduce size of parameters

Memo Date: . .

Regularization to reduce Overfitting

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

If we have 100 features

Then we have 100 parameters.

$w_1 x_1 + w_2 x_2^2 + \dots$

Now Cost fn is, (mean sq. Error)

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

} regularization term.

λ = regularization parameter.

$\lambda > 0$.

lesser value parameter, to new will be fn.

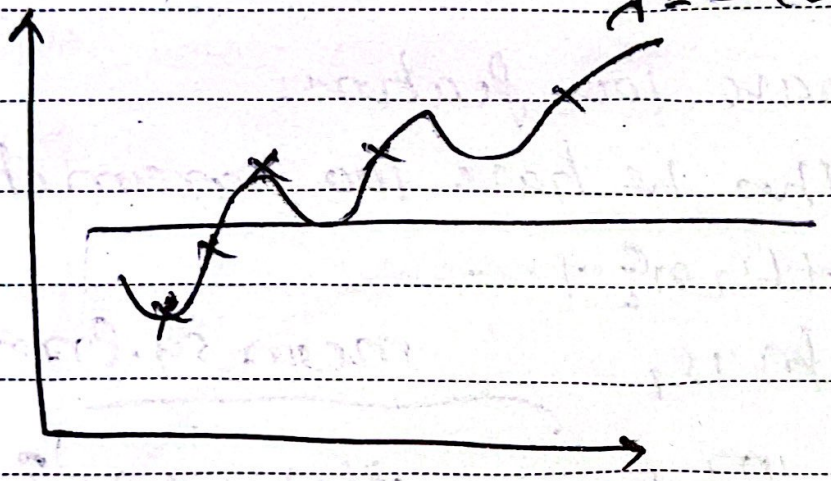
(b) is not regularized but some do

Memo Date.

If λ is larger, it will be underfit

If λ is small, it will be overfit

$\lambda = 0$ (overfit).



{ # Increasing λ, reduces overfitting by reducing size of parameters }

Implementing Gradient descent

repeat {

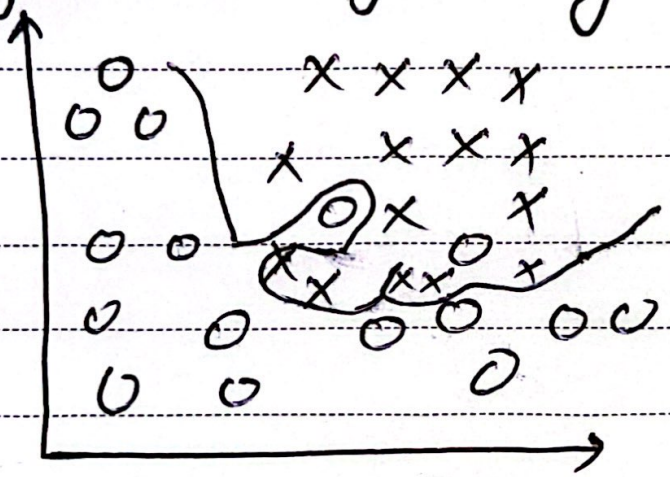
$$w_j = w_j - \alpha \left[\frac{1}{m} \sum_{i=1}^{m} \left[f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)} \right) x_j^{(i)} + \frac{\lambda}{m} w_j \right]$$

Memo Date: . .

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

}.

Regularized logistic Regression.



$$z = w_1 x_1 + w_2 x_2 + w_3 x_1^2 x_2 + w_4 x_1^2 x_2^2 + \cdots + b$$

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

Cost fn:

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

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

Memo Date: . .

Gradient descent:

Same as logistic.