Supervised

Regression:- Predict a number / infinitely many possible outputs

Classification:- predict categories / small number of possible outputs.

Unsupervised.

Data only comes with inputs x, but not Output labels y

Algorithm has to find structure in the data

Clustering
grp similar data pts together

Dimensionality reduction
Compress data using fewer numbers.

Anomaly detection
Find unusual data pts.

Linear Regression:-

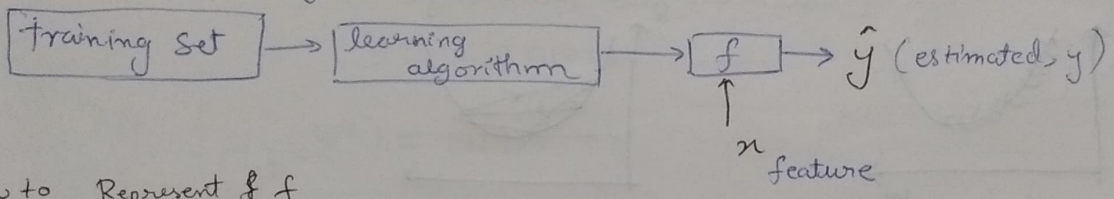
$x \rightarrow$ input variable / feature

$y \rightarrow$ output variable / target variable

$m \rightarrow$ no. of training examples

$(x, y) \rightarrow$ single training eg.

$(x^i, y^i) \rightarrow i^{th}$ training example.

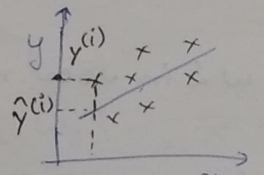


How to Represent f

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

$f(x)$

$w, b \rightarrow$ parameters, coefficients



Cost function :-

Squared Error Cost function

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

$m \rightarrow$ no. of training eg.

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

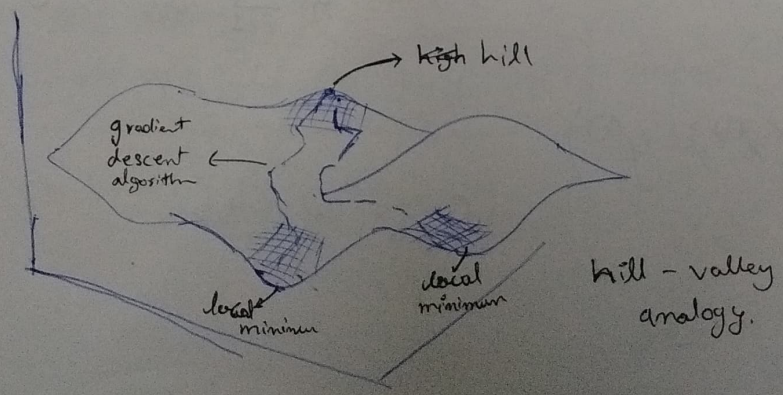
goal of linear Regression $\Rightarrow$ minimise $J(w)$ or $J(w,b)$

Gradient Descent

Have $J(w,b)$ Want $\min_{w,b} J(w,b)$

or Have $J(w_1, w_2, w_3 \dots, b)$ Want $\min_{w,b} J(w_1, w_2 \dots b)$

Outline → • Start with some w, b → Keep changing w, b to reduce $J(w,b)$



gradient descent algorithm

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

- $:=$ assignment operation
- α learning rate — determine how big step u take down hill

repeat until convergence : {

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

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

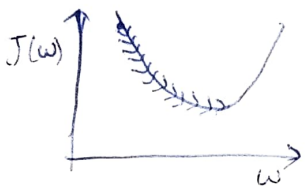
$w = tmp\_w$

$b = tmp\_b$

}

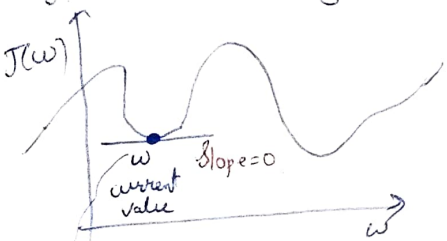
tends to find w, b for which $J(w,b)$ is min

\# if α is too small



if α is too large

\# if current value of w is in local minimum



$$w = w - \alpha \frac{\partial J(w)}{\partial w} \Rightarrow \text{update remains unchanged}$$

$w = w$

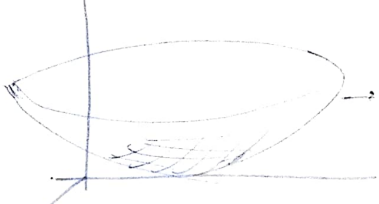
Traing Linear Regression for Gradient descent

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

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

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

Squared error cost



→ bowl shaped

~~feature Scaling~~ { Mutli-feature Regression }

$x_j \rightarrow j^{th}$ feature $n \rightarrow$ no. of features $\vec{x}^{(i)}$ = features of i^{th} training eg.

eg:-

$$f_{w,b}(x) = w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + b$$

price ← $f_{w,b}(x)$; x_1 → #size; x_2 → #bedroom; x_3 → #floor; x_4 → #years old; b → Base price

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

$b \rightarrow$ is a number

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

Vectorization

$\vec{w} = [w_1\ w_2\ w_3]$, $\vec{x} = [x_1\ x_2\ x_3]$, b

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

code ⟶ `f = np.dot(w,x) + b`

without vectorizat^n:-

without Vectorizat^n ↓

```
for j in range(16):
    w[j] = w[j] - 0.1*d[j]
```

with Vectorizat^n

$w = w - 0.1*d$

Gradient Descent for Multi-feature Regression

One feature

repeat {

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

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

}

multi-feature. $\frac{\partial J(\vec{w},b)}{\partial w_1}$

repeat {

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

$$\vdots$$

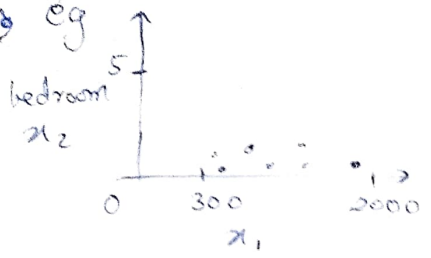
$$w_n = w_n - \alpha\left(\frac{1}{m}\sum_{i=1}^{m}\left(f_{w,b}(\vec{x}^{(i)}) - y^{(i)}\right)x_n^{(i)}\right)$$

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

feature Scaling

(for optimal maths, involved / no abrupt data scenarios in)

→ eg

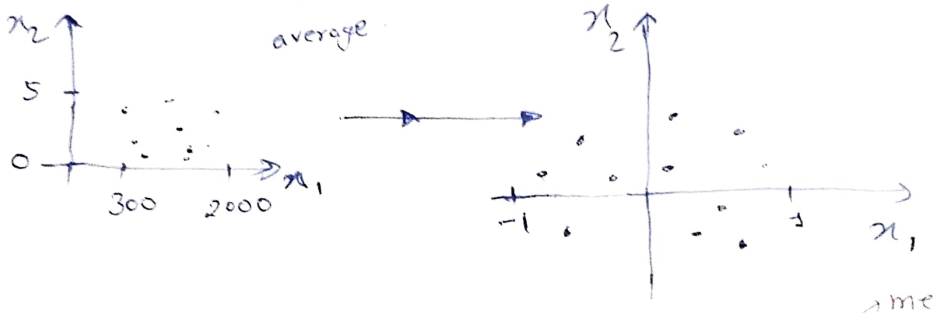


$300 \le x_1 \le 2000$ $\qquad$ $0 \le x_2 \le 5$

$x_{1,\text{scaled}} = \frac{x_1}{2000}$ $\qquad$ $x_{2,\text{scaled}} = \frac{x_2}{5}$

$0.15 \le x_{1,\text{scaled}} \le 1$ $\qquad$ $0 \le x_2 \le 1$

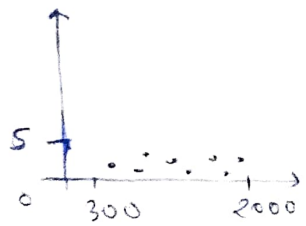
mean normalisatn

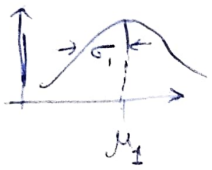


$$x_{1,\text{scaled}} = \frac{x_1 - \mu_1}{2000 - 300} \quad (\mu_1 = \text{mean},\ \text{max} - \text{min})$$

$\rightarrow -0.18 \le x_1 \le 0.82$

Z-score normalisatn





$300 \le x_1 \le 2000$

$$x_{1,\text{scaled}} = \frac{x_1 - \mu_1}{\sigma_1}$$

(μ_1 = mean; σ_1 = std. deviatn)

acceptable range

- $-1 < x < 1$
- $-3 < x < 3$
- $-0.3 < x < 0.3$

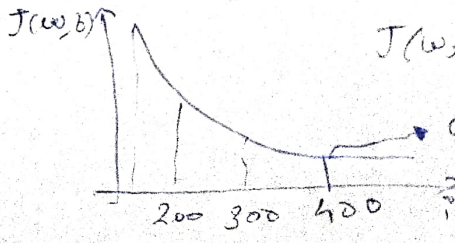
$-100 < x_3 < 100$

$-0.001 < x < 0.001$

$98.6 < x < 105$

---

make sure gradient descent is working correctly



$J(w,b)$ → should decrease after every iteratn

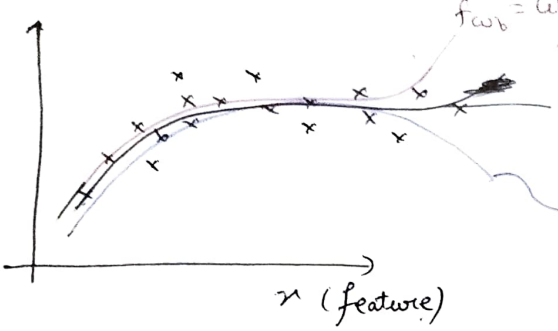
→ converges after 400 iteratn

Feature Engineering → Using intutions to design transforming or combining original features.

$f_{w,b} = w_1 x_1 + w_2 x_2 + b$ → x_3 → new feature → $f_{w,b} = w_1 x_1 + w_2 x_2 + w_3 x_3 + b$ (x_3: new feat)

$x_3 = x_1 x_2$

Polynomial Regression

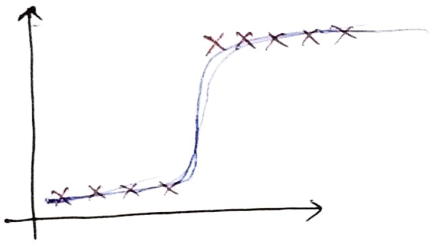


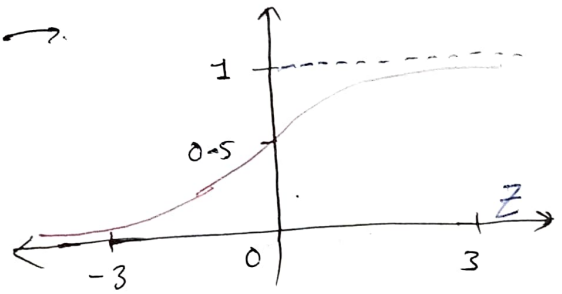
$f_{w,b} = w_1 x + w_2 x^2 + w_3 x^3 + b$

$f_{w,b} = w_1 x + w_2 \sqrt{x} + b$

$f_{w,b} = w_1 x + w_2 x^2 + b$

Logistic Regression





sigmoid fxn
logistic fxn

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

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

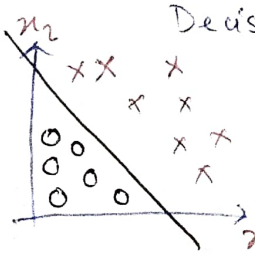
$z \rightarrow g(z) = \frac{1}{1+e^{-z}}$

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

Decision Boundary

$f_{w,b} = g(z) = g(w_1 x_1 + w_2 x_2 + b)$

$w_1 = 1,\ w_2 = 1,\ b = -3$



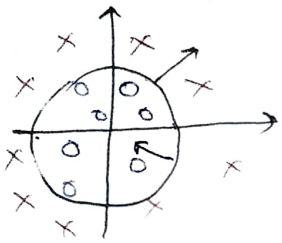
Decision boundary

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

$z = x_1 + x_2 - 3 = 0$

$x_1 + x_2 = 3$

non-linear decision boundaries



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

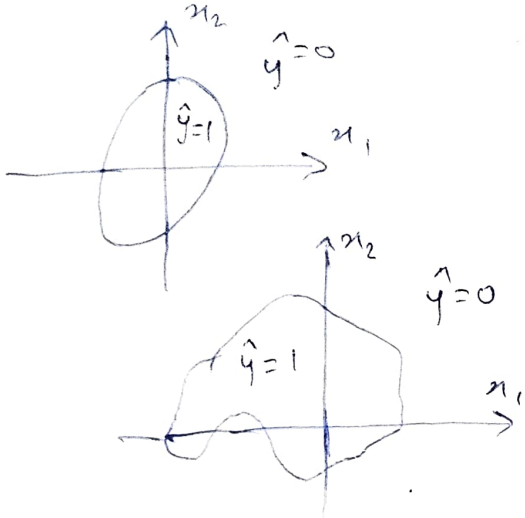
$w_1 = 1,\ w_2 = 1,\ b = -1$

decision boundary: $z = x_1^2 + x_2^2 - 1 = 0$

$x_1^2 + x_2^2 = 1$

$x_1^2 + x_2^2 \geq 1 \Rightarrow \hat{y} = 1$

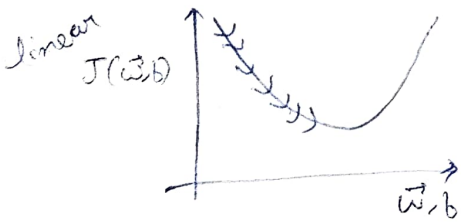
$x_1^2 + x_2^2 < 1 \Rightarrow \hat{y} = 0$

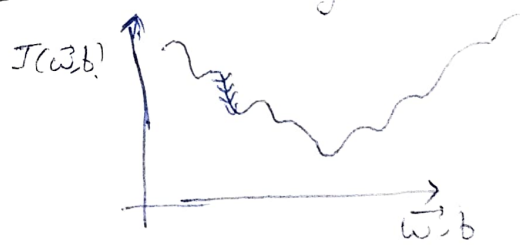


$$f_{\vec{w},b}(\vec{x}) = g(z) = g(w_1 x_1 + w_2 x_2 + w_3 x_1^2 + w_4 x_1 x_2 + w_5 x_2^2 + w_6 x_1^3 + \dots + b)$$

Cost fn for logistic Regression

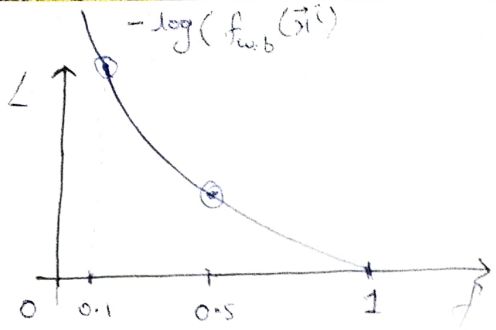
$$J(\vec{w},b) = \frac{1}{m}\sum_{i=1}^{m} \underbrace{\frac{1}{2}\left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)}\right)^2}_{\text{loss}} \quad \Big\} \text{ loss per training}$$

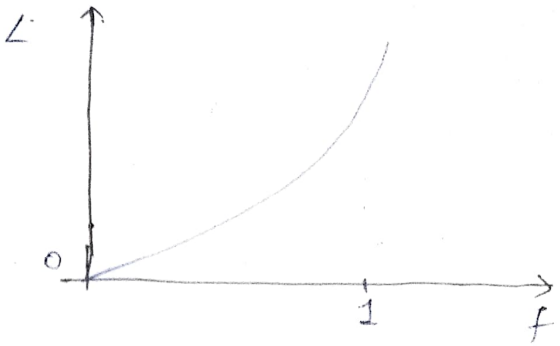




logistic loss funct^n:-

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





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

$$\text{cost} = \frac{1}{m} \sum L\left(f_{w,b}(x^{(i)}), y^{(i)}\right)$$

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

Training logistic Regression → Find $\vec{w}, b$

gradient descent

repeat {

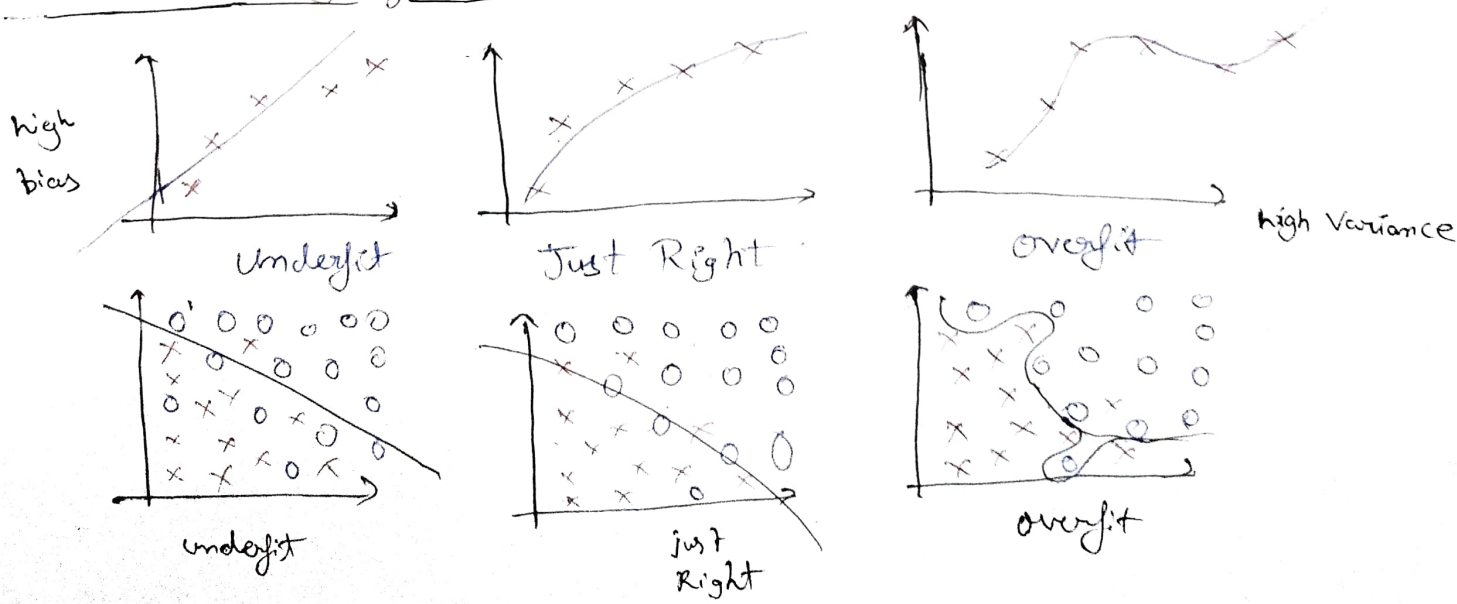
$$w_j = w_j - \alpha \left[\frac{1}{m} \sum_{i=1}^{m} \left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)}\right) x_j^{(i)}\right]$$

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

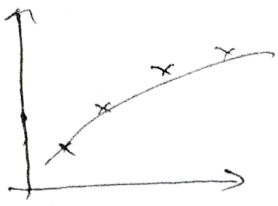
$$f_{\vec{w},b}(\vec{x}) = \frac{1}{1 + e^{-(\vec{w} \cdot \vec{x} + b)}}$$

ek training example ka error har feature k sath multiply hoga

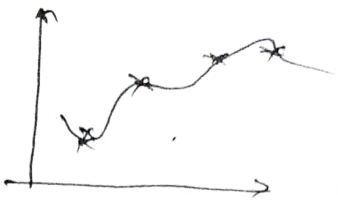
Overfitting



Regularizatⁿ



$w_1 x + w_2 x^2 + b$



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

make w_3, w_4 really small

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

mean sq^red error

regularisat^r term

choose $\lambda = 10^6$
then algorithm will choose the w_j small

fits data ↘ ↙ keep w_j small

λ balances both

gradient descent with Regularizatⁿ

$$f_{\vec{w},b}(\vec{x}) = w_1 x + w_2 x^2 + w_3 x^3 + w_4 x^4 + b$$

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

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

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

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

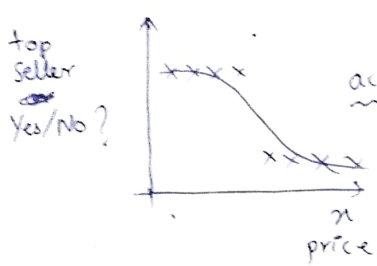
update rule of w, b →

$$w_j = w_j \left(1 - \alpha \frac{\lambda}{m}\right) - \underbrace{\frac{\alpha}{m} \sum \left(f_{w,b}(\vec{x}^i) - y^{(i)} \right) x_j^{(i)}}_{\text{usual update}}$$

b = usual update

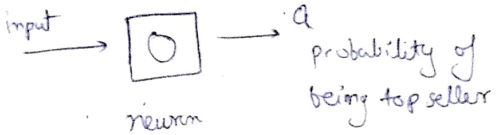
Neural Network

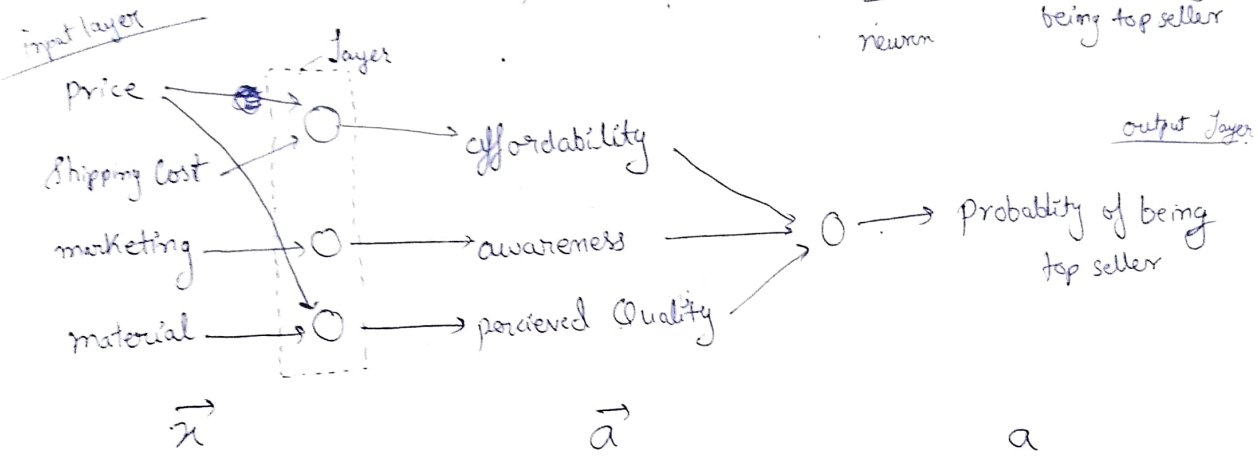
→ Demand Predictn

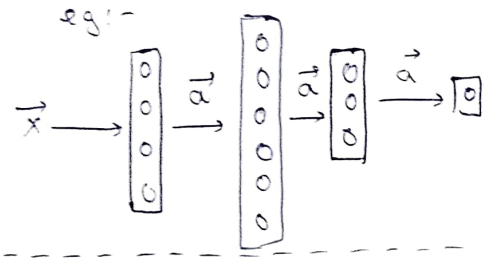


$x \rightarrow$ price

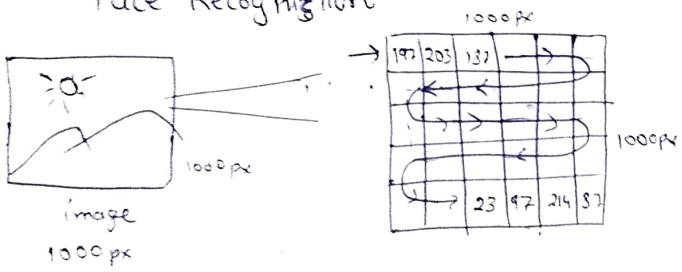
activation = $a = f(x) = \frac{1}{1 + e^{-(wx+b)}}$

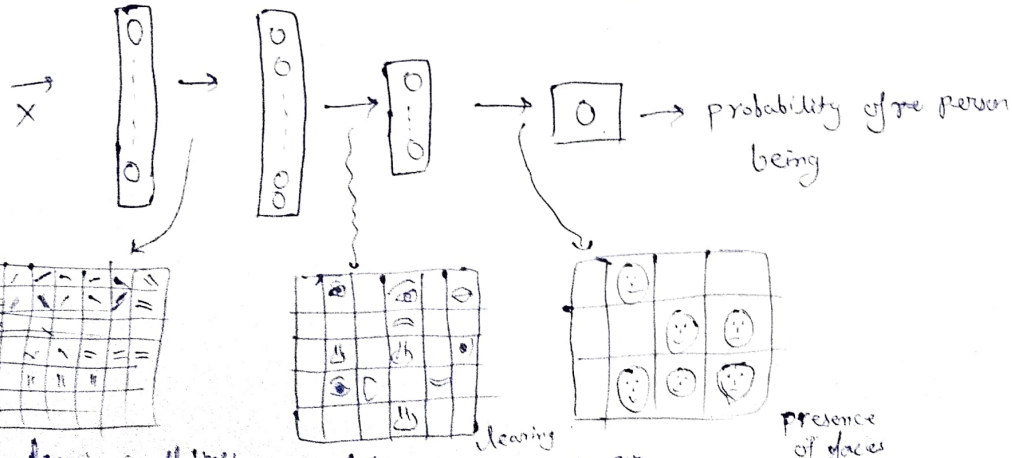




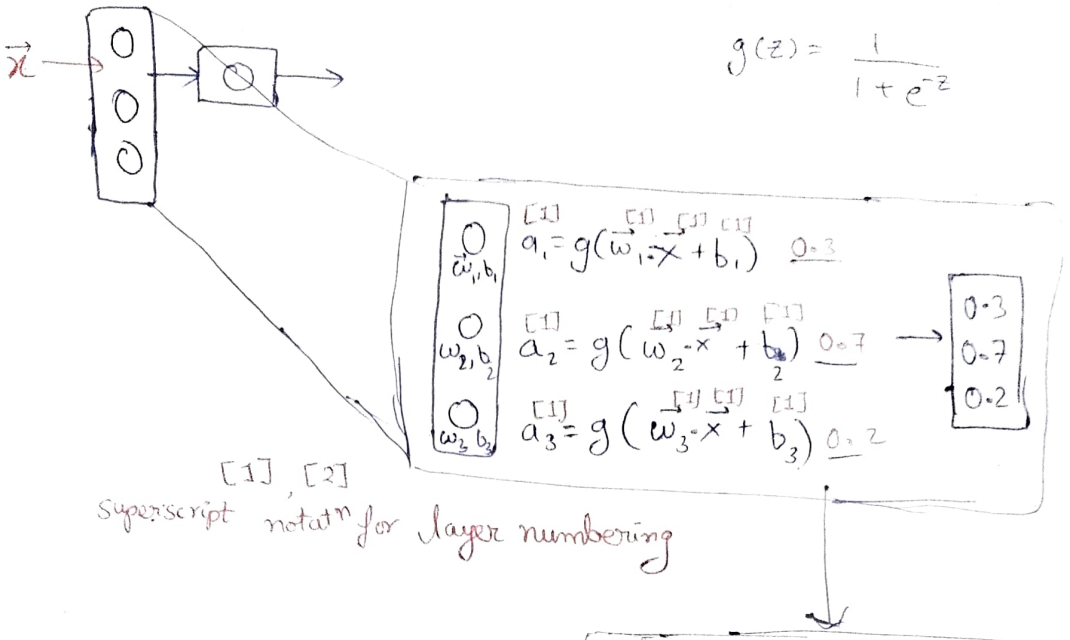


Face Recognition





Neural network layer



[1], [2] Superscript notation for layer numbering

activation value of layer 'l' unit (neuron) j

$$a_j^{[l]} = g(\vec{w}_j^{[l]} \cdot \vec{a}^{[l-1]} + b_j^{[l]})$$

Sigmoid Activation function

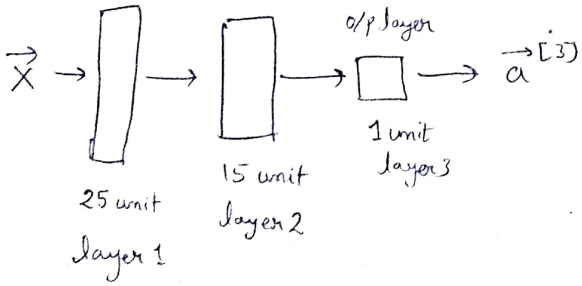
Parameters w, & b of layer l, unit j

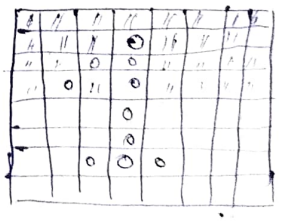
is $a^{[2]} \geq 0.5$?

yes → $\hat{y} = 1$

no → $\hat{y} = 0$

Hand Written Digit Recognition

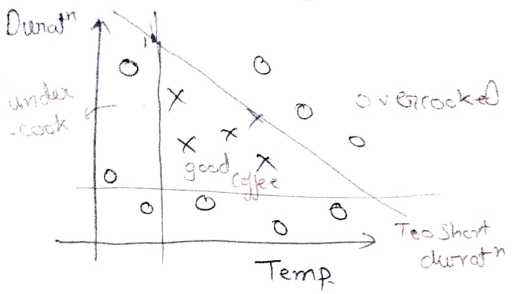


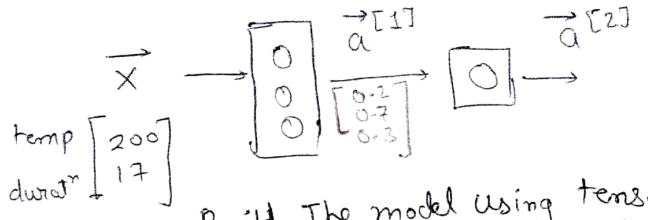


1 → 255 (white)

0 → Black

Coffee Roasting





Build The model using tensorflow

```
x = np.array([[200.0, 17.0]])
layer_1 = Dense(units=3, activatⁿ='sigmoid')
a1 = layer_1(x)
layer_2 = Dense(units=1, activation='sigmoid')
a2 = layer_2(a1)
```

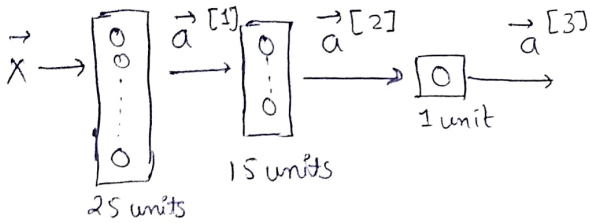
Tensor ← [[0.2, 0.7, 0.3]] ← a1 = layer_1(x)

1×3 matrix

if u print out a1 → tf.Tensor([[0.2, 0.7, 0.3]], shape=(1,3), dtype=float32)

a1.numpy → print array([[0.2, 0.7, 0.3]], dtype=float32)

model for digit classification




```
x = np.array([[0.0, ..... 255, ..240..., 0]])
layer_1 = Dense(units=25, activation='sigmoid')
a1 = layer_1(x)
layer_2 = Dense(units=15, activation='sigmoid')
a2 = layer_2(a1)
layer_3 = Dense(units=1, activation='sigmoid')
a3 = layer_3(a2)
```

if a3 >= 0.5 ↓ $\hat{y}=1$ else $\hat{y}=0$

---

numpy arrays / Matrices

~~x = np.array([200, 17])~~ x = np.array([[200, 17]]) → [200, 17] (row) — 1×2 (↓column, row × column)

x = np.array([[200], [17]]) → $\begin{bmatrix}200\\17\end{bmatrix}$ 2×1

x = np.array([200, 17]) → 1D vector

Building a neural network architecture

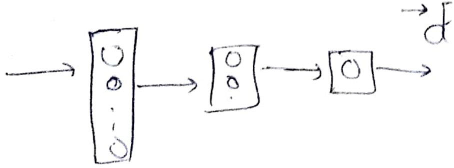
```
model = Sequential([
    Dense(units=3, activation="sigmoid"),
    Dense(units=1, activation="sigmoid")])
x = np.array([[200, 17]
              [120, 5]
              [425, 20]
              [212, 18]])
```

		y
200	17	1
120	5	0
425	20	0
212	18	1

```
y = np.array([1,0,0,1])
    model.compile(......)
    model.fit(x,y)

    model.predict(x_new)
```

→ **Digit Classification model:-**




```
layer_1 = Dense(units=25, activation="sigmoid")
layer_2 = " (" = 15, " ")
layer_3 = " (" = 1, " ")

model = Sequential([layer_1, layer_2, layer_3])
model.compile(-----)
x = np.array([[0....., 245,....., 17..],
              [0....., 200,....., 184..]])
y = np.array([1,0])
model.fit(x,y)
model.predict(x_new)
```

Coffee roasting model (using numpy/py)
forward Prop

```
x = np.array([200,17])
```

$a_1^{[1]} = g(\vec{w}_1^{[1]} \cdot \vec{x} + b_1^{[1]})$

```
w1_1 = np.array([1,2])
b1_1 = np.array([-1])
z1_1 = np.dot(w1_1, x) + b1_1

a1_1 = sigmoid(z1_1)
```

Similarly a1_2 ✓
a1_3 ✓

```
a1 = np.array([a1_1, a1_2, a1_3])
```
✓

forward Prop in Numpy

W → matrix

```
def dense (a_in, W, b):
    units = W.shape[1]
    a_out = np.zeros(units)        [0,0,0]
    for j in range(units):         0,1,2   → pulling out column from 2D array
        w = W[:, j]
        z = np.dot(w, a_in) + b[j]
        a_out[j] = g(z)
    return a_out
```

eg:-

$w_1^{[1]} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$, $w_2^{[1]} = \begin{bmatrix} -3 \\ 4 \end{bmatrix}$, $w_3^{[1]} = \begin{bmatrix} 5 \\ -6 \end{bmatrix}$

```
W = np.array([
    [1, -3, 5],
    [2,  4, -6]])
```

$b_1^{[1]} = -1$, $b_2^{[1]} = 1$, $b_3^{[1]} = 2$

```
b = np.array([-1, 1, 2])
```

$\vec{a}^{[0]} = \vec{x}$

```
a_in = np.array([-2, 4])
```

```
def sequential(x):
    a1 = dense(x, W1, b1)
    a2 = dense(a1, W2, b2)
    a3 = dense(a2, W3, b3)
    a4 = dense(a3, W4, b4)
    f_x = a4
    return f_x
```

Dot prod

$\begin{bmatrix} \\ \vec{a} \\ \\ \end{bmatrix} \cdot \begin{bmatrix} \\ \vec{w} \\ \\ \end{bmatrix}$

$z = \vec{a} \cdot \vec{w}$

$\begin{bmatrix} \leftarrow \vec{a} \rightarrow \end{bmatrix} \begin{bmatrix} \\ \vec{w} \\ \\ \end{bmatrix}$

$z = \vec{a}^T \vec{w}$

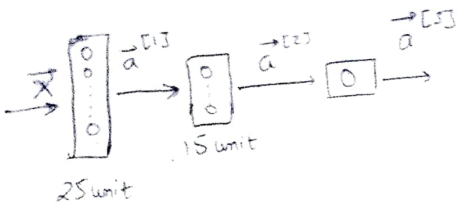
vector vector multiplication

vectorisation of above code

```
X = np.array([[200, 17]])
W = np.array([[1, -3, 5],
              [-2, 4, -6]])
B = np.array([[-1, 1, 2]])

def dense(A_in, W, B):
    Z = np.matmul(A_in, W) + B
    A_out = g(Z)
    return A_out
```

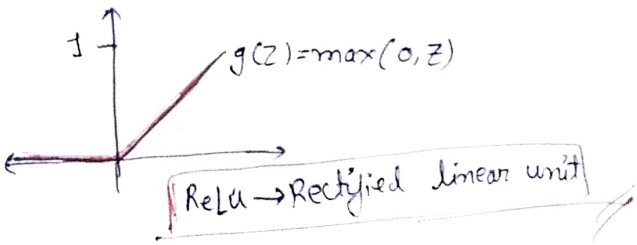
Training NN in Tensorflow




```
import tensorflow as tf
from tensorflow.keras.import Sequential
from tensorflow.keras.layers import Dense
    model = Sequential([
Dense(units=25, activation='sigmoid'),
Dense(units=15, activation='sigmoid'),
Dense(units=1, activation='sigmoid'),
    ])
```

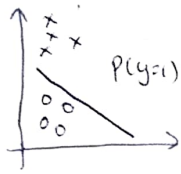
```
from tensorflow.keras.losses import
BinaryCrossentropy
    model.compile(loss=BinaryCrossentropy())
    model.fit(X, Y, epochs=100)
```

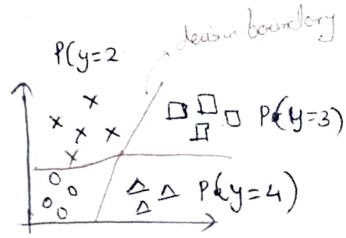
→ alternatives to sigmoid fxn in Activat^{n} fxn



ReLU → Rectified linear unit

Multiclass classificat^{n} Problem:-





Softmax Regression ↳ Binary → Multi class classificat^{n} problem.

× $z_1 = \vec{w}_1 \cdot \vec{x} + b_1$

○ $z_2 = \vec{w}_2 \cdot \vec{x} + b_2$

□ $z_3 = \vec{w}_3 \cdot \vec{x} + b_3$

△ $z_4 = \vec{w}_4 \cdot \vec{x} + b_4$

$$a_1 = \frac{e^{z_1}}{e^{z_1} + e^{z_2} + e^{z_3} + e^{z_4}} = P(y=1|\vec{x})$$

$$a_2 = \frac{e^{z_2}}{e^{z_1} + e^{z_2} + e^{z_3} + e^{z_4}} = P(y=2|\vec{x})$$

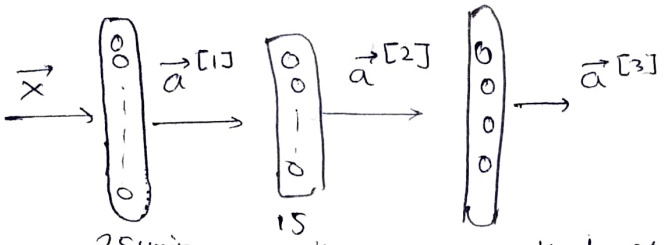
Softmax Regression

Crossentropy loss

$$\text{loss}(a_1, \dots a_N, y) = \begin{cases} -\log a_1 & \text{if } y=1 \\ -\log a_2 & \text{if } y=2 \\ \vdots & \vdots \\ -\log a_N & \text{if } y=N \end{cases}$$

Neural Network with Softmax Output

Digit Recognition Model.



$$z_1 = \vec{w}_1^{[3]} \cdot \vec{a}^{[2]} + b_1$$

$$a_1^{[3]} = \frac{e^{z_1^{[3]}}}{e^{z_1^{[3]}} + \dots e^{z_{10}^{[3]}}} = P(y=1 \mid \vec{x})$$

MNIST with Softmax

better (recommended) version later.

Don't use the code version here!

```
import tensorflow as tf
from tensorflow.keras import Sequential
from tensorflow.keras ~~import~~ layers import Dense
model = Sequential([
        Dense(units=25, activation='relu'),
        D------=15, ----------),
        De-------=10, activation='softmax')])
```

'linear' → replacements for more numerically accurate implementation

```
from tensorflow.keras.losses import
    SparseCategoricalCrossentropy
model.compile(loss=SparseCategoricalCrossentropy())
model.fit(X, Y, epochs=100)
```

(from_logits=True)

Additional Concepts

- ~~ADA~~ Adam Optimisation
- Convolutional layer

Debugging Regression model :-

→ get more training examples , → Try smaller sets of features
→ try getting additional feature , → Try adding polynomial features (x_1^2, x_2^2 ---)
→ try decreasing , increasing λ

Evaluating and Chosing model :-

- model selection

$d=1$ $f_{wb} = w_1 x_1 + b$ → $J_{test}^{<1>}$

$d=2$ $f_{wb} = w_2 x_2^2 + w_1 x_1 + b$ → $J_{test}^{<2>}$

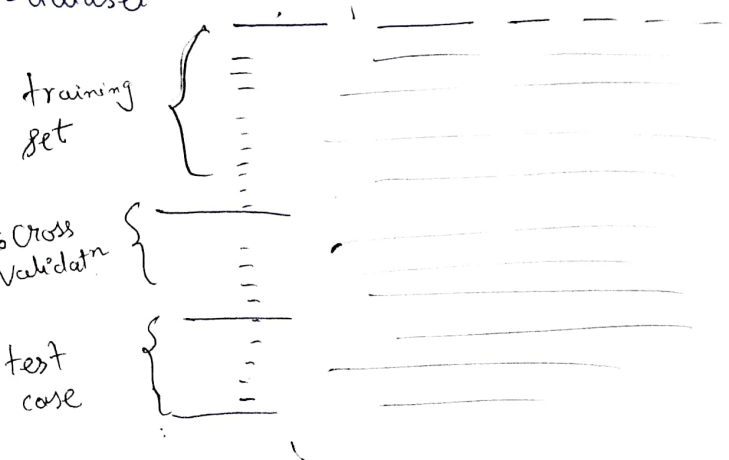
$d=3$ $f_{wb} = w_3 x_3^3 + w_2 x^2 + w_1 x_1 + b$ → $J_{test}^{<3>}$

$d=10$ ⇒ $f_{wb} = w_{10} x^{10} + \cdots + w_1 x_1 + b$ → $J_{test}^{<10>}$

$\leftarrow w_1 x_1 + x_1^2 w_2 + \cdots w_5 x_5^5 + b$

lowest $J_{test}^{<5>}$ ✓ (works well).

- dataset



1. $J_{cv}(w^{(1)}, b^{(1)})$
2. $J_{cv}(w^{(2)}, b^{(2)})$ ← low error → model Selected
3. $J_{cv}(w^{(3)}, b^{(3)})$

Train CV

Pick $w^{(2)}, b^{(2)}$

↳ generalisatn error using test set

$J_{test}(w^{(2)}, b^{(2)})$

→ regularisation and bias/variance

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

regularizing parameter

Try $\lambda = 0$

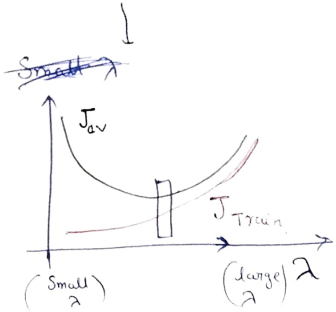
Try $\lambda = 0.01$

⋮ eg.

→ $\lambda = 0.08$ (goes well)

⋮

Try $\lambda \approx 10$

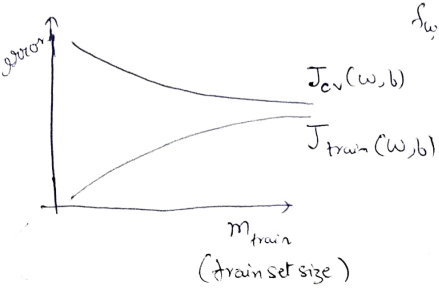


→ Establishing base line level of Performance.

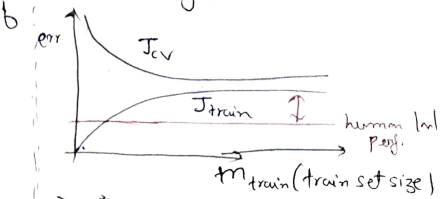
- Human lvl performance
- Competing algorithms —"—
- Ones based on experience

→ Learning Curves.

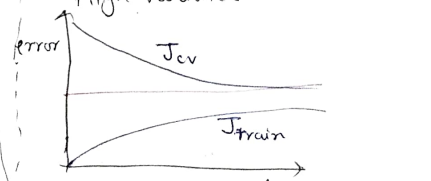
$f_{\vec{w},b} = w_1 x + x_2^2 w_2 + b$



→ with High bias



→ with High variance



Debugging Algo - revisited.

- get more training set → fixes high variance
- try smaller set of feature → fixes high variance
- adding additional features → fixes high bias.
- try adding polyno. features (x_1, x_1^2, x_2, x_2, etc) → fixes high bias
- decreasing λ → fixes high bias
- increasing λ → fixes high variance

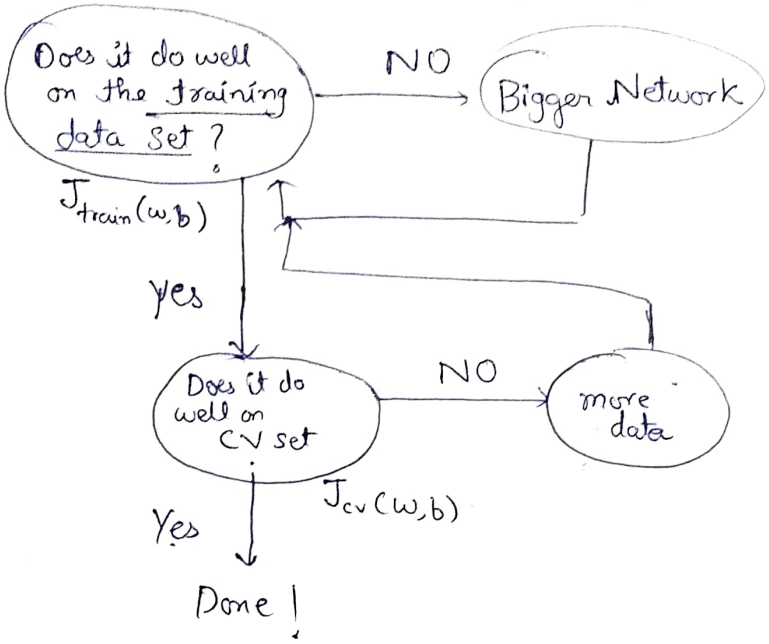
helps to do better on the training set

Simple model — High bias ⟷ trade off ⟷ Complex model — High variance

$f = wx + b$

$f = w_1x + w_2x^2 + w_3x^3 + w_4x^4 + b$

→ large N^2 are low bias machine.



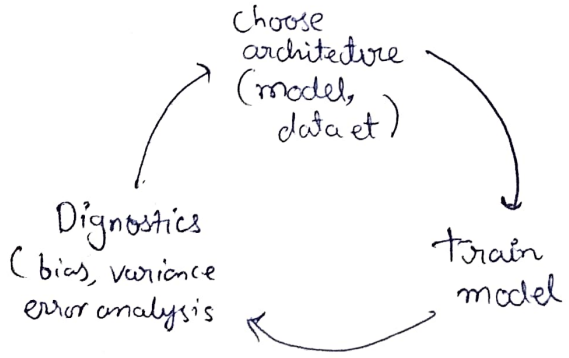
N. N. regularizat^n

$$J(W,B) = \frac{1}{m}\sum^{m} L(f(\vec{x}^{(i)}), y^{(i)}) + \frac{\lambda}{2m}\sum(w^2)$$

all weight W

ML Devlopment process :-

iterative loop of ML dev.

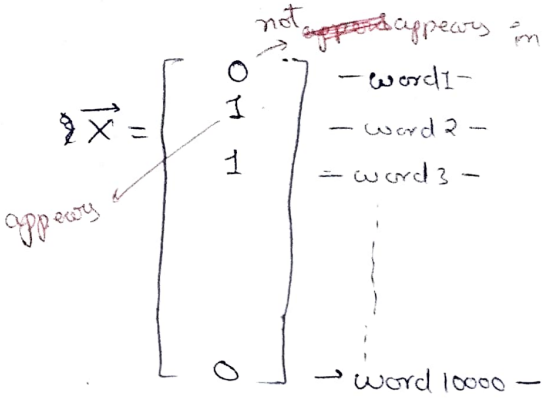


eg:- Building a spam classifier

Supervised learning : $\vec{x}$ = features of email

y = Spam(1) or not Spam(0)

Features: list the top 10,000 words to compute. $x_1, x_2, x_3 \ldots x_{10,000}$



~~Set~~ Set example of an email

→ train NN / or LR → try to reduce error

- collect more data
- develop features based on email routing
- Define features from email body
- Design algo. to detect misspellings eg: w4tches, m0rtgage

→ Error Analysis

$m_{cv} = 500$ examples in CV set

Algorithm misclassifies 100 of them

manually examine 100 eg. and categorize them & on common traits.

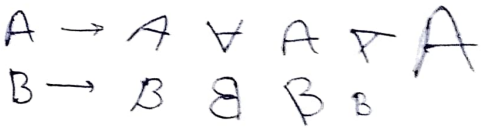
eg of common traits:
- ~ deliberate misspellings (w4tches, etc)
- ~ Unusual routing
- ~ Steal passwords
- ~ Spam-message in embedded image.

→ Adding Data:

- Artificial Data Synthesis

1. Data augmentation → create new data by modifying ~~exist~~ previous data

eg,



→ by introducing distortions.

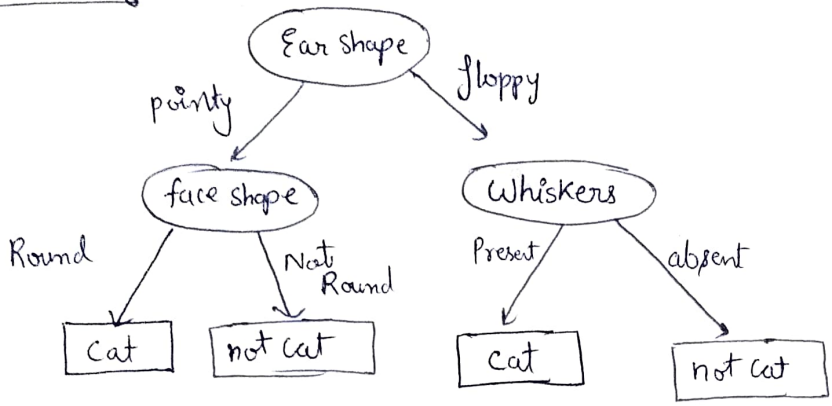
• → used for speech regonition by adding different-different background noise / effects.

AI = Code + Data!

Decision Tree Model

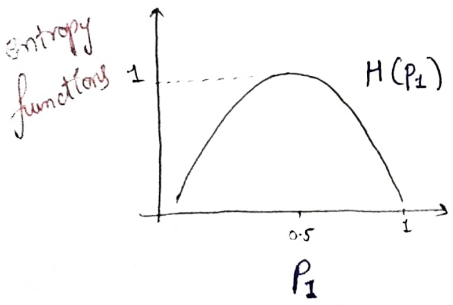
eg

cat identificatn



- How to choose what feature to split on at each node?
  → ensure ~~no~~ maximum purity.
- When to stop splitting?
  → when a node is 100% one class
  → When splitting a node will result in the tree exceeding a maximum depth.
  → when no. of egs in a node is below a threshhold.
  → when improvements in purity score are below a threshold

Measuring purity



p_1 → fract^n of example that is desired

$$H(p_1) = -p\log_2(p_1) - (1-p_1)\log_2(1-p_1)$$

informatn Gain

$$\text{info. Gain} = H(p_1^{root}) - \left(w^{left} H(p_1^{left}) + w^{right} H(p_1^{right})\right)$$

p_1^{root} of true situation of desired

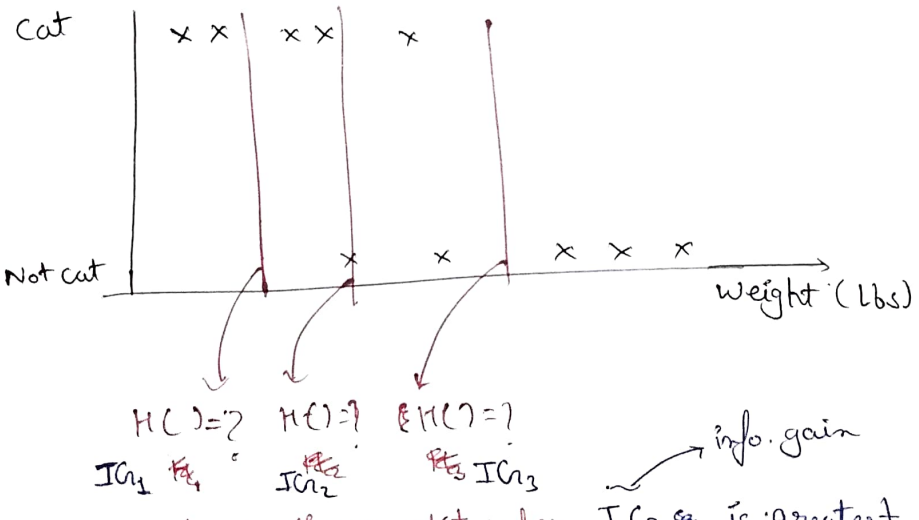
w^{left} Weighted avg fraction $\left(\frac{\text{no. of eg in this}}{\text{Total no. of eg.}}\right)$

→ Decision tree learning:-

- Start with all examples at the root node.
- Calculate information gain for all possible features, and pick the one with the highest information gain.
- Split dataset according to selected feature, and create left and right branches of tree.
- Keep repeating, splitting process until stopping criteria is met:-

One-hot encoding:- if a feature take 'K' values
→ Create K features with (0 or 1).

Splitting a ~~con~~ continous Variable:-



XGr Boost

classifatn

```
from xgboost import XGrBClassifier
model = XGrBClassifier()
model.fit(X_train, Y_train)
Y_pred = model.predict(X_test)
```

Regression

```
from xgboost import XGrBRegressor
model = XGrBRegressor()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
```

Decision trees Vs Nueral Network

- works well on tabular Data
- not recommended for unstructured & data (image, audio, txt)
- fast
- small decision are human interpretable

- works well on all types of data
- may be slower than a decision tree.
- works with transfer learning