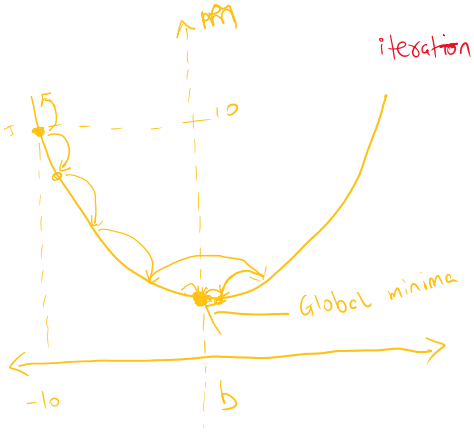
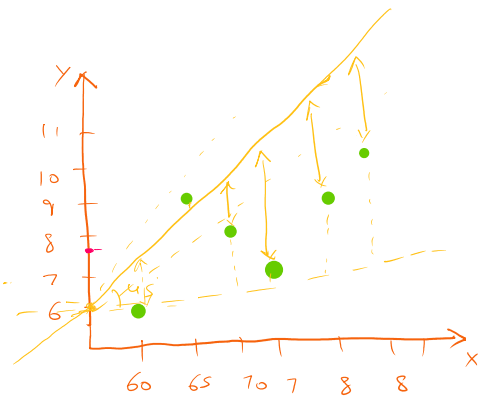
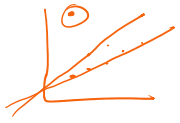



| | m | b | E |
|---|---|---|---|
| iteration 1 - | 45 | 1 | 100 |
| 2 - | 50 | 1 | 120 |
| 3 | 40 | 1 | 80 |
| | 30 | 1 | 40 |
| | 20 | 1 | 10 |
| | 10 | 1 | 2 |
| | 5 | | 20 |
| | 8 | | 10 |
| | 9 | | 5 |
| | 12 | | 6 |



## Assumption:

- Linearity
- Independence of Error
- Homoscedasticity
- No correlation b/w features

## Limitation :

- Linearity assumption
- outliers
- overfitting

## use cases!

## when to Apply

- Linear data
- simple & interpretable model
- moderate to high data

- **L1 & L2 Regularization :**

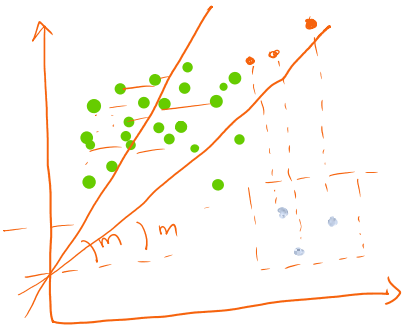
$$\text{Loss} = \left[\frac{1}{2}\sum (y - \hat{y})^2\right] + \lambda \sum (m)^2$$

(learning rate $(\cdot 1 - \cdot 005)$, Penalty)

$+ \lambda \sum (m)^2$

L1 Error - Lasso Regression

L2 Error - Ridge Regression




Trade off -

$y = m_1x_1 + m_2x_2 + m_3x_3 + m_0$ — Lasso

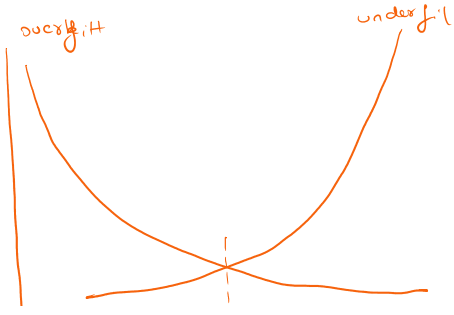
$y = m_1x_1$

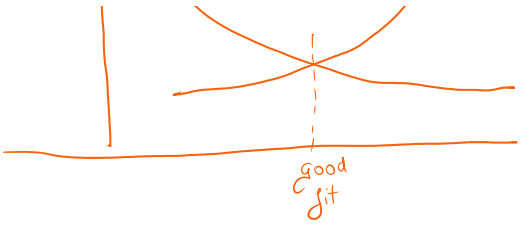
→ train error — overfitting

→ test error —

↑ bias & variance ↑

overfitting underfitting

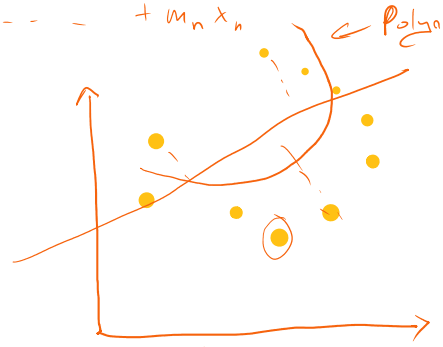





## Polynomical Regression

$$\hat{y} = m_0 + m_1 x_1 + m_2 x_2^{(2)} + m_3 x_3^{(3)} + \cdots + m_n x_n^{n}$$

← Polynomial

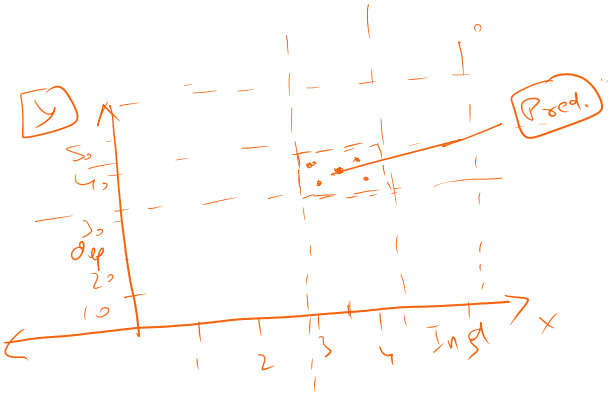


When to use

- Non linear

Assumptions :

- Linearity
- Independence




- Decision tree →
- Sum
- KNN

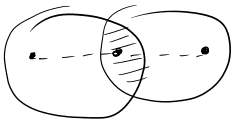
$x = 3.5$

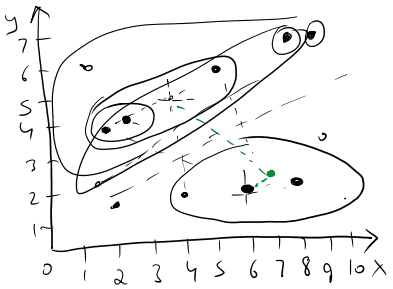
$y = 40$

# Unsupervised learning. – (Descriptive)

Sunday, September 8, 2024 8:06 PM

- K-Means

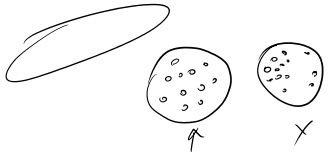




- Types of K-means:
  - K-Mean ++ :
  - Mini Batch K Mean :
  - Fuzzy K Mean :

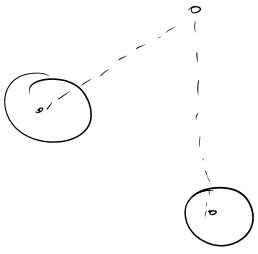
Assumption:

- K - number of cluster
- Cluster Shape
- Homogeneous density



Limitations :-

- Sensitivity to initial Centroids
- Number cluster
- Sensitive to outliers
- Cluster shape assumptions



Use Cases:

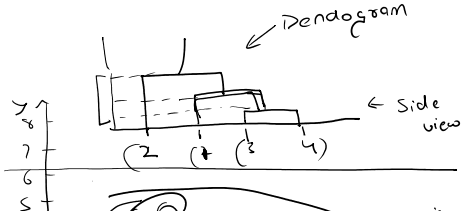
- Customer segmentation
- Document clustering
- Image compression

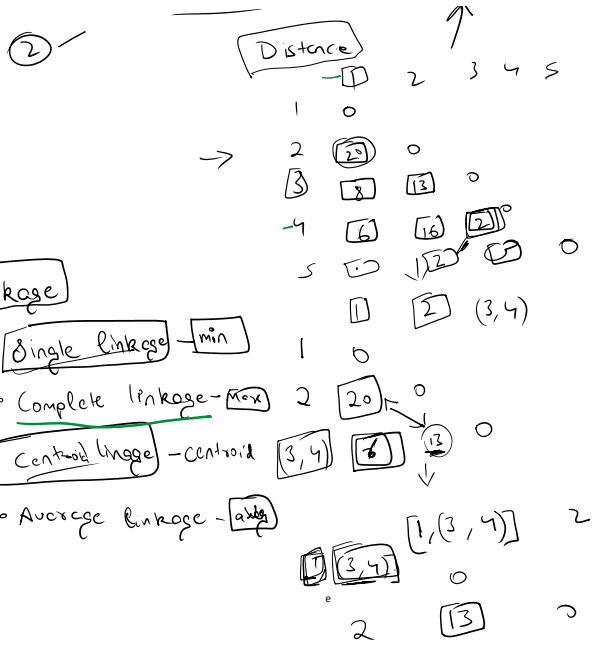
When to Apply:

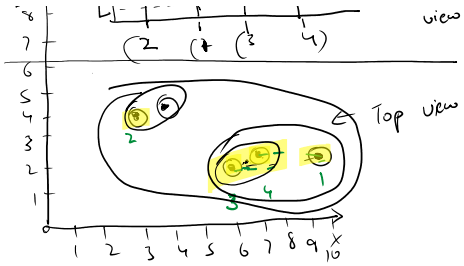
- Normal to big data
- estimation of no. of clusters
-

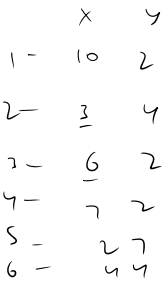
(2,5)
(3,7)

Agglomerative Clustering :-





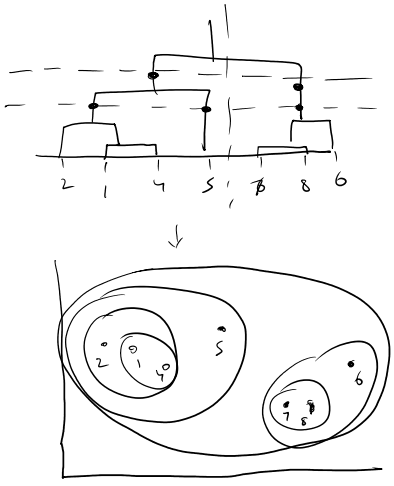



$$D_{23} = \sqrt{(3-6)^2 + (4-2)^2} = \sqrt{13}$$

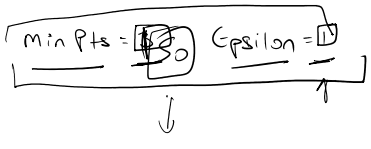
Linkage:

- Single: grouping of plant
- Complete:
- Average
- Centroid / ward

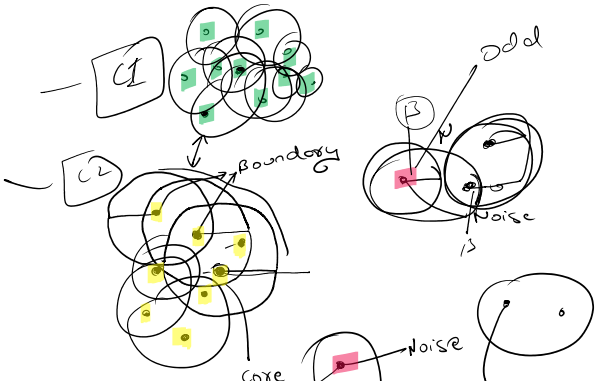


- Assumptions:
  - Distance / linkage choice
  - Scalable
- Use Case:
- When to Apply:
  - Explore hierarchical str. & granularity level.
  - No estimation
  - Intuitive to visualize.

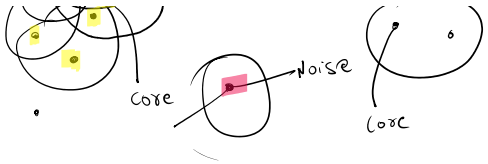
- DB Scan – (Anomaly detection)




- Core –



- Core
- Boundary
- Noise




Limitation :

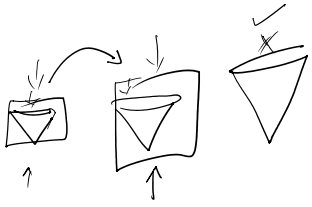
- Parameter Sensitive
- High Dimension data / very large data

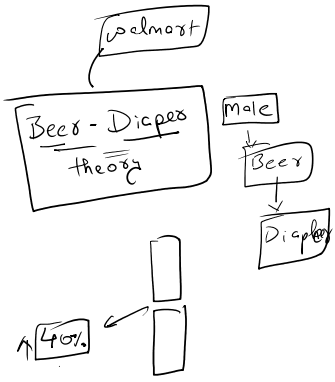
Use Case :

- Anomaly

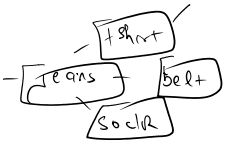
→ Association techniques → Market Basket Analysis

- FP-growth






- Upselling
- Down selling
- Cross selling



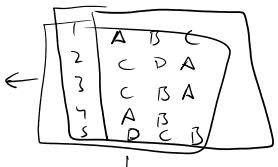

FP-growths

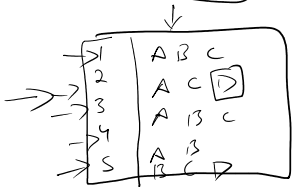
- threshold value = (2)
- Support = 2

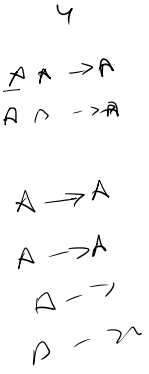
A = 4
B = 4
C = 4

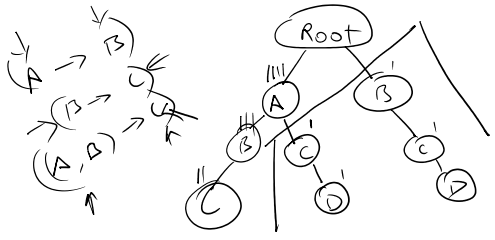
Rice
Flour
Pulse










Assumption :

- Structure pre-defined tabular data
- assume the existence of FP.

Use Case

Limitations:

- Memory usage
- Handling large dataset
- Handling numeric data

Use Case

- Web usage mining.
- Biology & genomic
- Network Intrusion Detect.

When to Apply:

- FP
- Processed. Data
- Threshold

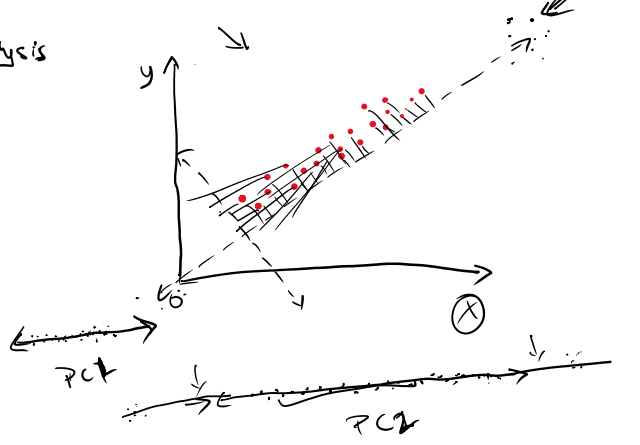
→ ~~Apriori~~ →

→ ~~Eclat~~

## Dimension Reduction

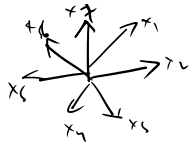
"Curse of Dimensionality"

- PCA :- Principle Component Analysis




⇒

No. of PC ≤ No. of features



| X | Y |
|---|---|
| 2.5 | 2.4 |
| .5 | .9 |
| 2.2 | 2.9 |
| 1.9 | 2.2 |
| 3.1 | 3.0 |
| 2.3 | 2.7 |
| 2 | 1.6 |
| 1 | 1.1 |
| 1.5 | 1.6 |
| 1.1 | 0.9 |

Step 1: Cal. the mean of each colum :

$\bar{X} = 1.81$ $\bar{Y} = 1.91$

Step:2 - Cal. the covariance matrix

1)

| x | $\bar{x}$ | $x-\bar{x}$ | $(x-\bar{x})(x-\bar{x})$ |
|---|---|---|---|
| 2.5 | 1.81 | .69 | .476 |
| .5 | 1.81 | -1.31 | 1.7161 |

$\downarrow$

$\left(\frac{5.5490}{9}\right) =$

2)

| x | y | $x-\bar{x}$ | $y-\bar{y}$ | $(x-\bar{x})(y-\bar{y})$ |
|---|---|---|---|---|
| 2.5 | 2.4 | .69 | .49 | .3381 |
| | | | | 1.5851 |
| | | | | ⋮ |
| | | | | 5.538 |

$\frac{5.538}{9}$

$C_{2\times2} = \begin{bmatrix} Cov(x,x) & Cov(x,y) \\ Cov(y,x) & Cov(y,y) \end{bmatrix}$

$Cov(x,x) = \frac{\sum_{1}^{n} (x-\bar{x})(x-\bar{x})}{n-1}$

$Cov(y,y) = \frac{6.449}{9}$

$C_{2\times2} = \begin{bmatrix} .6165 & .6154 \\ .6154 & .7165 \end{bmatrix}$

Step 1:- Eigen values & vector.

$$C - \lambda I = 0$$

$$\begin{bmatrix} .6165 & .6154 \\ .6154 & .7165 \end{bmatrix} - \lambda \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = 0$$

$$\begin{bmatrix} .6165 - \lambda & .6154 \\ .6154 & .7165 - \lambda \end{bmatrix} = 0$$

$$\lambda^2 - 1.3330\lambda + 0.0630 = 0$$

$\lambda_1 =$ 0.0490

$\lambda_2 =$ 1.2840 →

$$C\vec{v} = \lambda\vec{v}$$

$$\begin{bmatrix} .6165 & .6154 \\ .6154 & .7165 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = .049 \begin{bmatrix} x \\ y \end{bmatrix}$$

$$.6165x_1 + .6154y_1 = .049x_1$$

$$.6154x_1 + .7165y_1 = .049y_1$$

$$.6165x_1 - .049x_1 = -.6154y_1$$

$$.5675x_1 = -.6154y_1$$

$$x_1 = -1.0845y_1$$

Assuming $y_1 = 1$

$$\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} -1.0845 \\ 1 \end{bmatrix}$$

$$= (-1.0845)^2 + (1)^2$$

$$1.1764 + 1$$

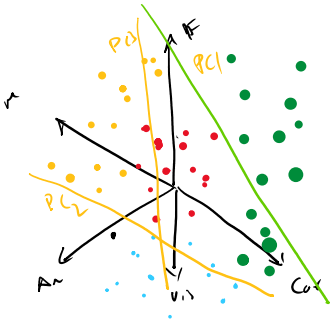
$$\sqrt{2.1764}$$

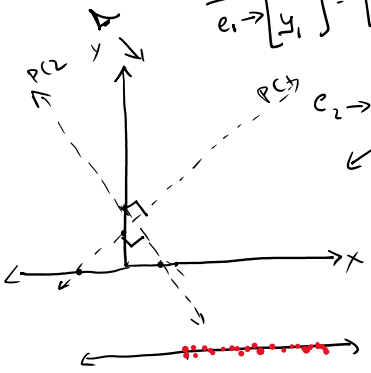
$$1.475$$

$$-1.0845$$

$$-1.475$$

$$\frac{1}{1.475}$$


$$PC1 \quad e_1 \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} -.7351 \\ .6778 \end{bmatrix}$$

$$C_2 \rightarrow \begin{bmatrix} .6778 \\ .7351 \end{bmatrix}$$

Step: 5

$$e_1^T \begin{bmatrix} x_i & \bar{x} \\ y_i & \bar{y} \end{bmatrix}$$

$$\begin{bmatrix} -.7351 & .6778 \end{bmatrix} \begin{bmatrix} 2.5 - 1.81 \\ 2.4 - 1.9 \end{bmatrix}$$

$$-.7351 [.69] + .6778 [.5]$$

$$= .166$$

| X | Y | PC1 |
|---|---|---|
| 2.5 | 2.4 | .166 |
| .5 | .9 | |
| 2.2 | 2.9 | |
| 1.9 | 2.2 | |
| 3.1 | 3. | |
| 2.3 | 2.7 | |
| 2 | 1.6 | |
| 1 | 1.1 | |
| 1.5 | 1.6 | |
| 1.1 | 0.9 | |

Assumptionst

- Linearity
- mean is zero
- uncorreloted PCA components

Use cases:

- Dimension Reduction
- Avoid overfitting
- Visualizing high dimension features

Limitation :

- Linearity

Limitation :-

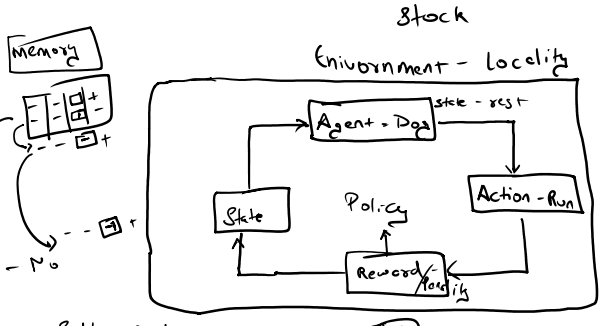
- Linearity
- Loss of interpretability
- Sensitive to scaling

Dimension Reduction :-

- Kernal PCA
- t-SNE
- UMAP

→ Reinforcement Learning :-

- Enviroment → Exchange
- Agent → Bot
- State → Current Position - No
- Action → Buy, Sell, hold
- Policy → ∘ Types
- Reward → Buy - Sell ∘ Learning by interaction
  +ve → Rew
  -ve → Pen ∘ Goal oriented learning




Sell → 100
→ Buy 90 = +ve
→ Sell 110 = -ve

Rel. -
90 - [S.]

1000 [1010 +ve → 1010 -ve 990-H]

-1
→ 1000 - Bu
1010 - H
1005 - H
990-H

- Semi - Supervised :- language translation

Eng → French
French → Eng
Supervised

- Transfer learning :-

model

# Deep learning :-

Friday, September 20, 2024 9:29 PM

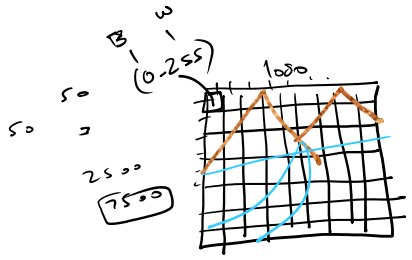
Text Image

ML → 10000

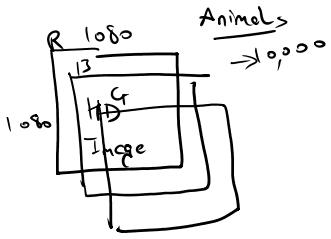
## Machine learning :

### Limitation :-

- High dimensionality data :-
- Feature Selection → Automatics
- Image, videos, text

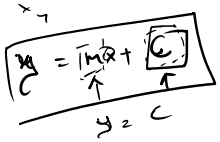







## Deep learning :-

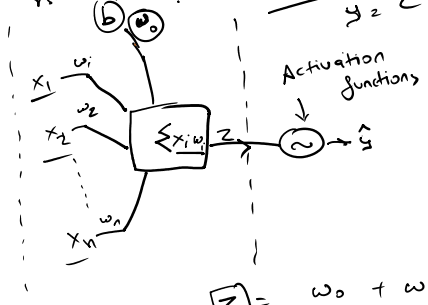
$x_1$ $x_2$ $x_3$ $x_4$

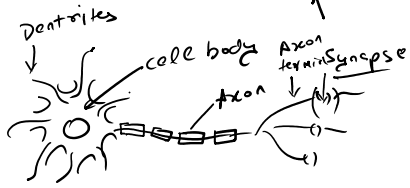



→ venom
- Blood thicken
Paralysis

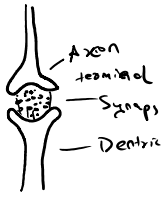
1080 × 1080 × 3

(10,000 row | 30,000,000 columns)




Biological Neuron

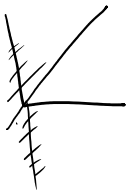



$$z = w_0 + w_1x_1 + w_2x_2 + \cdots + w_nx_n$$

- X = Input
- ω = weights
- b/$ω_0$ = bias
- Activation function
  - Sigmoid = $\frac{1}{1-e^{-x}}$
  - Relu
  - tanh

$z \leftrightarrow y$

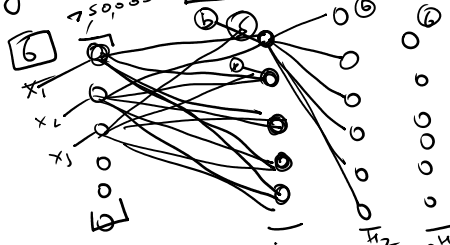
$$\hat{y} = \frac{1}{1-e^{-z}}$$




Network of Neurons :- Neural Networks

7,50,000

Fully connected NN




Input layer

Hidden layer

output layer

Dog
Cat
Mouse

6 × 5 + 5
30 + 5
35

Data Process

36 42 21

is NN ≤ 2 HL → Shallow Neural Network

If NN ≤ 2 HL → Shallow Neural Network

NN > 2 HL → Deep Neural Network

1 Neuron → Preceptron

How to train the Neural Network:

Step 1 — Feed Forward

Step 2: $[y - \hat{y}]$

Step 3 — Backpropogation

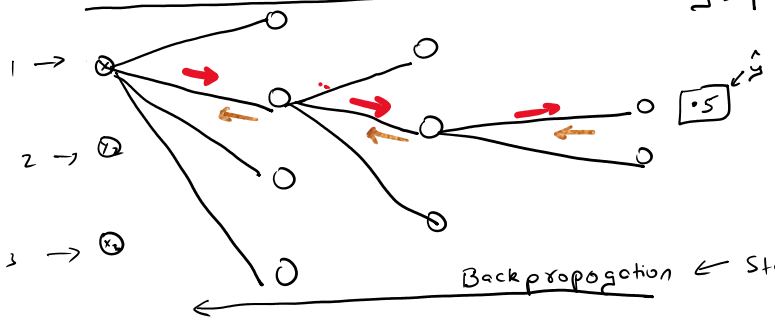
FCNN

↓

Feed Forward

Fully connected Neural Networks

$x_1 \rightarrow 1\ 2\ 3$




IL H1 H2 OL

FFFCNN

Mathematics:

Step 1:

$$z = \sum w_i x_i$$

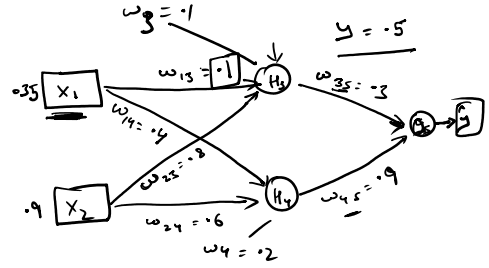
$$\hat{y} = \frac{1}{1+e^{-z}}$$

$$z_3 = (.35 \times .1) + (.9 \times .8) + b = .755$$

$$\hat{y}_3 = \frac{1}{1+e^{-.755}} = .68$$

$$z_4 = (.35 \times .4) + (.9 \times .6) + .2 = .68$$

$$\hat{y}_4 = \frac{1}{1+e^{-.68}} = .66$$




$$z_5 = (.68 \times .3) + (.66 \times .9) = .79$$

$$\hat{y}_5 = \frac{1}{1+e^{-.79}} = .687$$

Step 2: $y - \hat{y}$

$.5 - .687 = -.187$

Step 3:

$\Delta w_{ji} = \eta \cdot \delta_j O_i$

$\rightarrow \delta_j = O_j(1-O_j)(\epsilon_j - O_j)$ {j = output}

$\delta_j = O_j(1-O_j) \sum \delta_k w_{jk}$

$\eta$ = learning rate — 1 [0.1 — 0.001]

$\delta_j$ = error energy

$O_j$ = Output of neuron

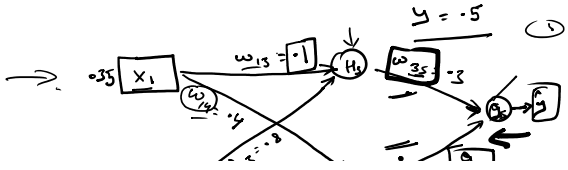
$\epsilon_j$ = correct output

For output

$$\delta_5 = \hat{y}(1-\hat{y})(y-\hat{y})$$

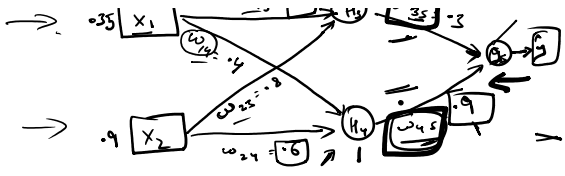
$$= .687(1-.687)(.5-.687) = -.198$$

Exploding gradient



$\delta_5 = \hat{y}(1-\hat{y})(y-\hat{y})$

$= .687(1-.687)(.5-.687) = -.1198$




$\delta_3 = y_3(1-y_3)(w_{35} \times \delta_5)$

$= .68(1-.68)(.3 \times -.178) = -.0265$

$\delta_4 = y_4(1-y_4)(w_{45} \times \delta_5)$

$= .66(1-.66)(.9 \times -.178) = -.0166$

$\Delta w_{ij} = \eta \delta_5 z_4$ — (2nd)

$= 1 \times -.178 \times .66$

$= -.11$

$w_{45}$ (new) $= \Delta w_{45} + w_{45}$ (old)

$= -.11 + .9$

$= .79$

$\Delta w_{14} = \eta \delta_4 x_1$

$= 1 \times -.166 \times .35$

$= -.058$

$w_{14}$ (new) $= \Delta w_{14} + w_{14}$ (old)

2nd Iteration

Step 1 →

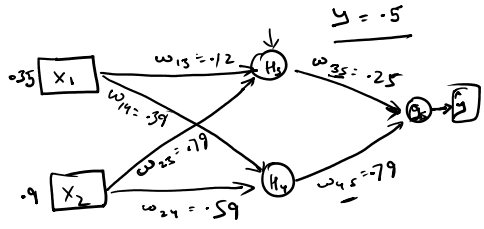
Feed Forword = .65

Step 2 →

Error = .5 − .65 = −.15

Step 3 →

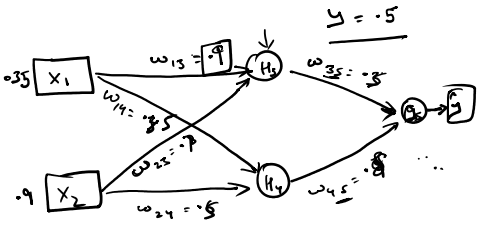
Backpoogyth →




3rd I

- Feed F
- Error → .01

Back-




Types of Activation Functions :

ssification

1) Sigmoid : $\frac{1}{1+e^{-z}}$

Range : [0, 1]

Use Cases:

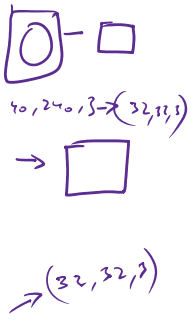
- binary Classification

Pros:

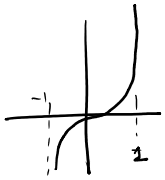
Prob based funct., easy implement

Cons:

- Vanishing Gradient




2) Hyperbolic tan : $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$




Range [-1, 1]

Use:

hidden layer

Pros:

- zero centered output help optimization
- less saturation

Cons:

Vanishing Gradient

3) Relu = max(0, x) Rectified Linear Unit

Range : [0, ∞]

Use cases

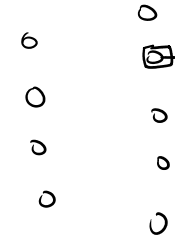
Ridden layers

Pros:

- Computationally inexpensive
- vanishing gradient won't work

Cons:

Dying Relu Problem



0, 255

4) Leaky Relu $\begin{cases} x & \text{if } x > 0 \\ \alpha x & \text{if } x < 0 \end{cases}$

$\alpha = (.01)$

Range: $(-\infty, \infty)$

Regression / General

- [Classification]

- Swish $= \frac{x}{1 + e^{-x}}$

- Swish $= \frac{x}{1+e^{-x}}$

  Range = $(-\infty, \infty)$

  Use Case:

  deep network

  Pros:

  - Avoid vanishing gradient
  - Smooth - & non-monotonic

  Cons:

  computationally expensive

- Softmax: $\frac{e^{x_i}}{\sum_i e^{x_j}}$ Range: (0, 1)

  3 class

  0 — .1
  0 — .2
  0 — .7
  1

  Use Case:

  multi-class classification

  Pros: Converts logits to prob. distribution

  Cons:

  - outlier data
  - class are mutually exclusive

optimizer:

- Gradient Desent (GD): $\theta_{new} = \theta_{old} - \eta \nabla \delta\theta$

  Use Cases

  - Small datasets

  Pros:

  Simple & effective on convex problem

  Cons:

  Slow convergence & computationally expensive for large dataset.

- SGD: $\theta = \theta - \eta \nabla \delta(\theta, x_i, y_i)$ $(x, y)$ - individual samples

  Use cases:

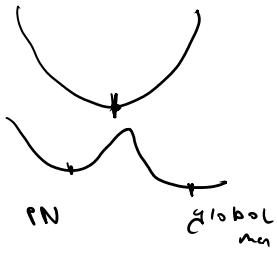
  - big datasets

  Pros:

- big datasets

Pros:
- Fast processing

Cons:
Noisy updates, may not reach global minima.




- Mini batch GD : $\theta_{new} = \theta_{old} - \eta \nabla \delta(\theta, batch)$

Use cases:
Large datasets, more stable than SGD

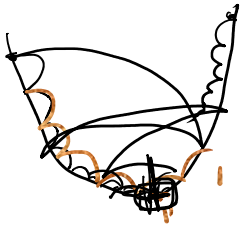
Pros:
- Reducing noise & variance SGD
- Faster than GD

Cons:
Very large dataset

.8 = 1 − .05 − .01 (.5)

- Momentum Optimizer: $\theta = \theta - (\gamma v) - \eta \nabla \delta(\theta)$ (momentum = v)

$\gamma$ = momentum factor = .9



Use Case:
use in training deep network

Pros:
- Speed up the convergence
- Reducing oscillation
- Big data also

Cons:
tuneable parameter of momentum.

Adaptive optimizer

Adagrad = $\theta = \theta - \frac{\eta}{\sqrt{G_t + \epsilon}} \cdot \nabla \delta(\theta)$

$G_t$ = sum of sq. of past gradients

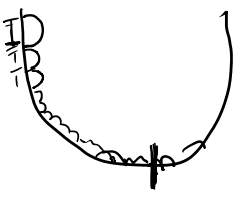
$\epsilon$ = constant small value to avoid zero division.

.01 (.1)$_{t+1}$

$\eta$ = .1 / .01

- Use Cases:
Sparse data, NLP, image

- Pros:
Adaptive learning rate



- Cons:

Pros:
Adaptive learning rate

• Cons:
learning rate keeps on decay

RMS Prop :- Root mean Sq. propagation

$$\therefore G_t = \beta G_t + (1-\beta)(\nabla \delta\theta)^2 \qquad \beta = \text{decay factor} = .9$$

$$\left[\theta = \theta - \frac{\eta}{\sqrt{G+\epsilon}} \cdot \nabla \delta(\theta)\right]$$

Use:-
NLP,

Pros:
Addressing adagrad cons.

Cons:
$\beta$ is sensitive

Adam: Adaptive momentum Estimation

- First momentum Estimation:

$$m_t = \beta_1 m_t + (1-\beta)\nabla J(\theta) \qquad \beta = .9$$

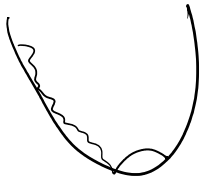
- second momentum Estimation:

$$v_t = \beta v_t + (1-\beta)\nabla \delta(\theta)$$

Bias correction: $\hat{m}_t = \frac{m_t}{1-\beta}$ , $\hat{v}_t = \frac{v_t}{1-\beta}$

update: $\theta = \theta - \frac{\eta}{\sqrt{\hat{v}_t + \epsilon}} \cdot \hat{m}_t$

Pros:
- Adaptive & momentum optimizer benefits
- Efficiently works with large & noisy data.
- Convergence faster.



Cons
. Chances of overshoots
. Computationally expensive

- Chances of overshoots
- computationally expensive

**Adomax** :-

$$v_t = \max\left(\beta v_{t-1}, |\nabla J(\theta)|\right)$$

$$\theta = \theta - \frac{\eta}{v_t} \cdot \hat{m}_t$$