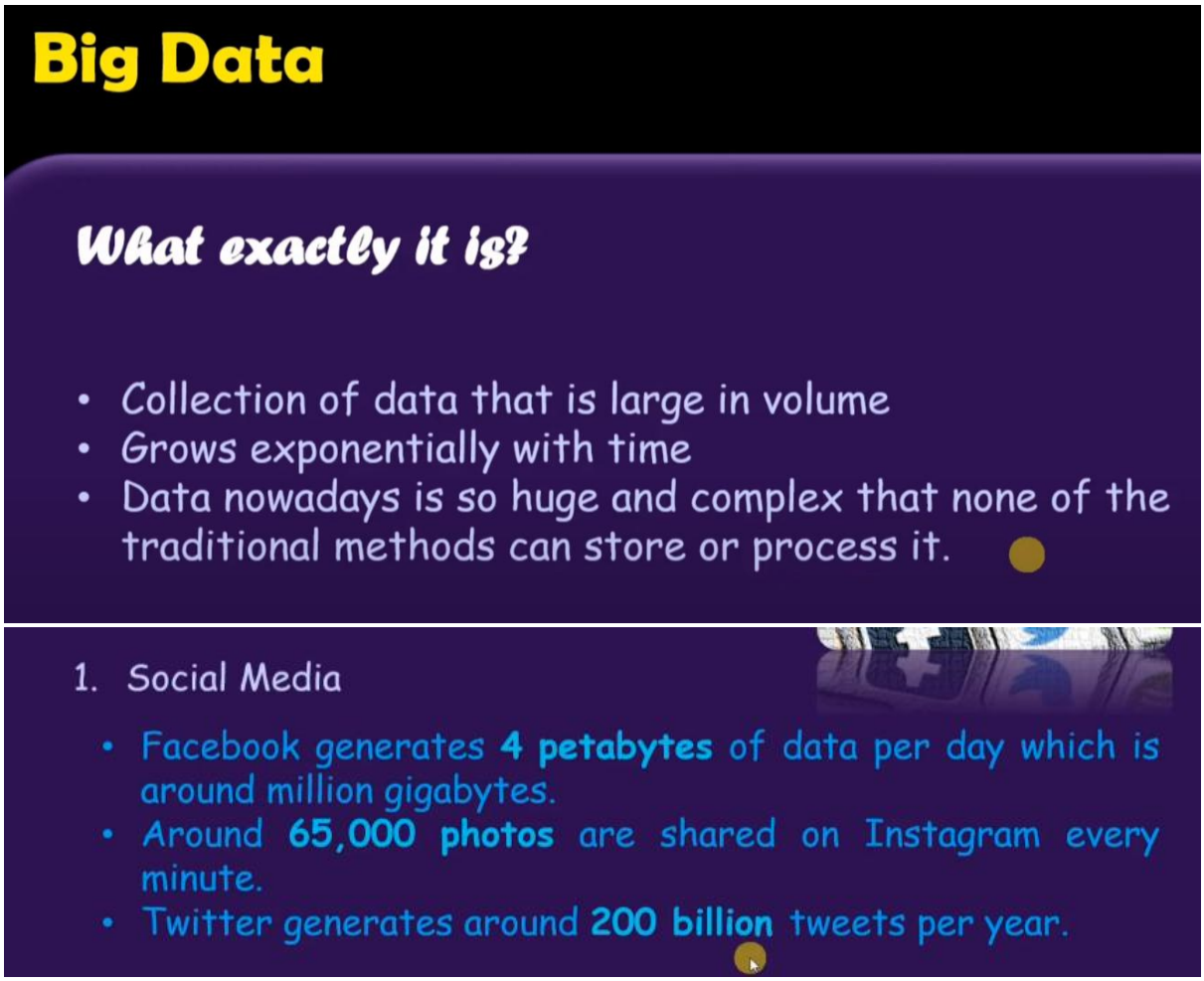
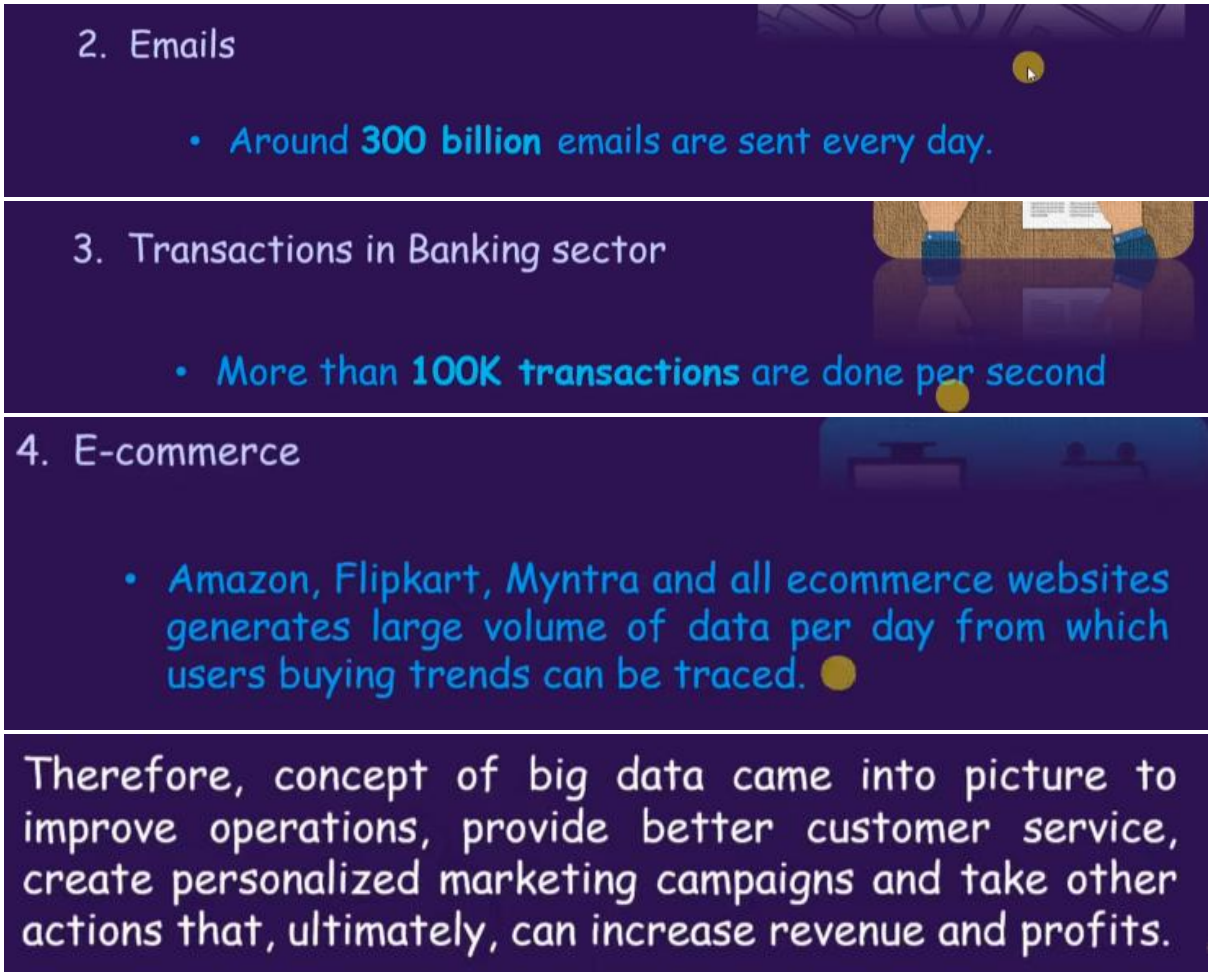
Notes Made by Dare-Marvel

BIG DATA ANALYTICS

Introduction to Big Data





Characteristics / 7 V's of Big Data

https://www.includehelp.com/big-data-analytics/7-vs-of-big-data.aspx

Types of Big Data

https://www.geeksforgeeks.org/types-of-big-data/

Traditional vs Big Data Business Approach

https://www.geeksforgeeks.org/difference-between-traditional-data-and-big-data/

Applications of Big Data

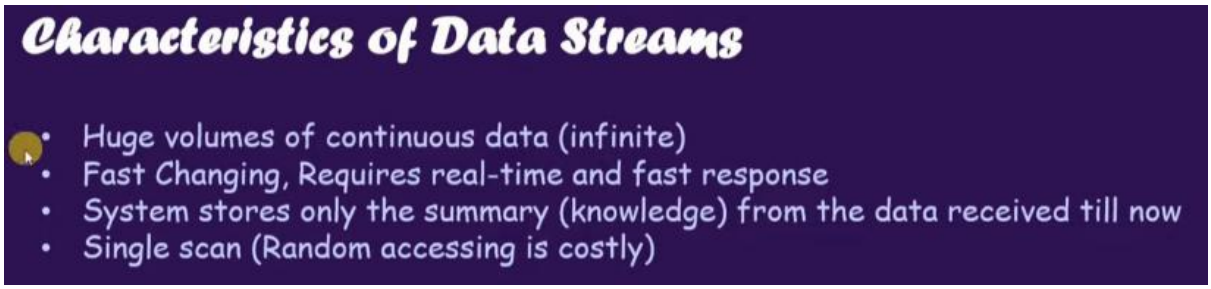
https://www.geeksforgeeks.org/applications-of-big-data/

Big Data Challenges

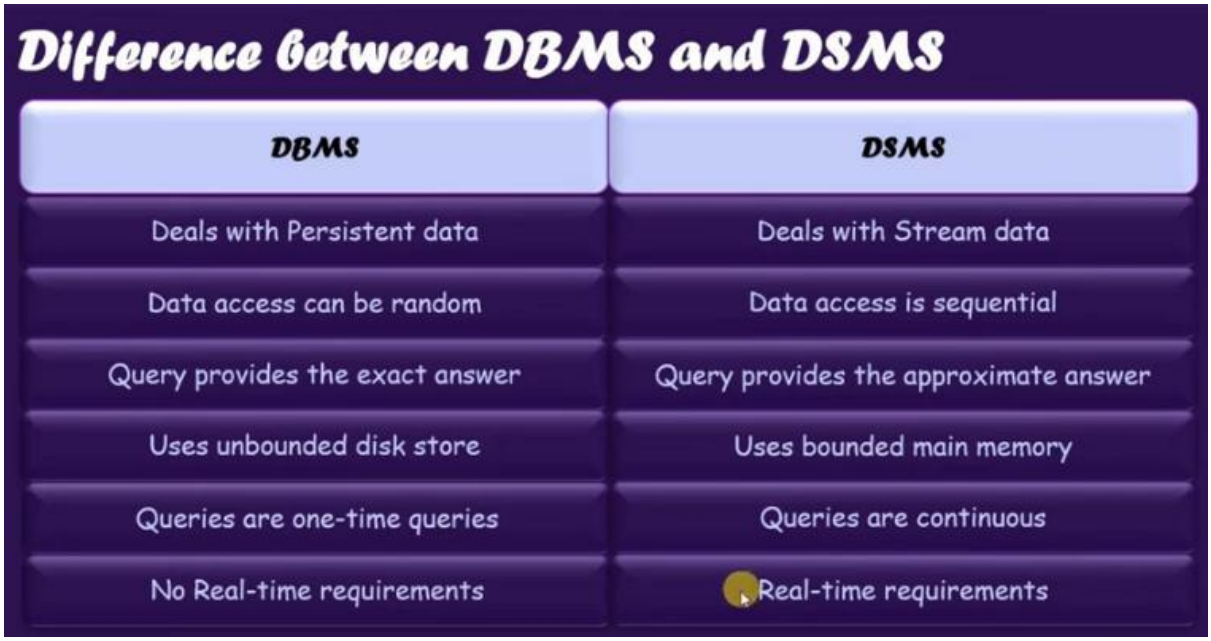
https://www.geeksforgeeks.org/big-challenges-with-big-data/

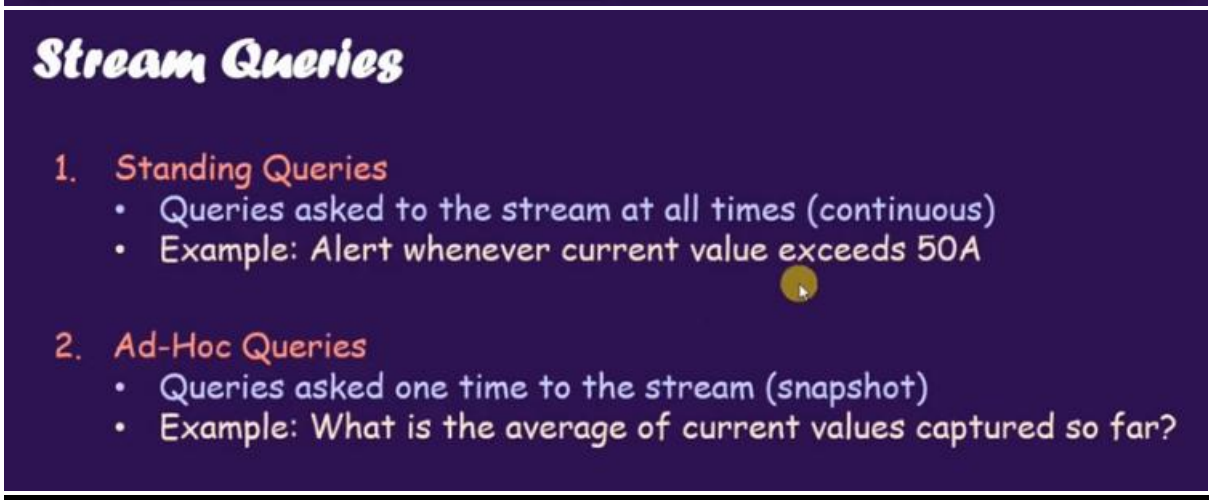
Memory efficient data structures

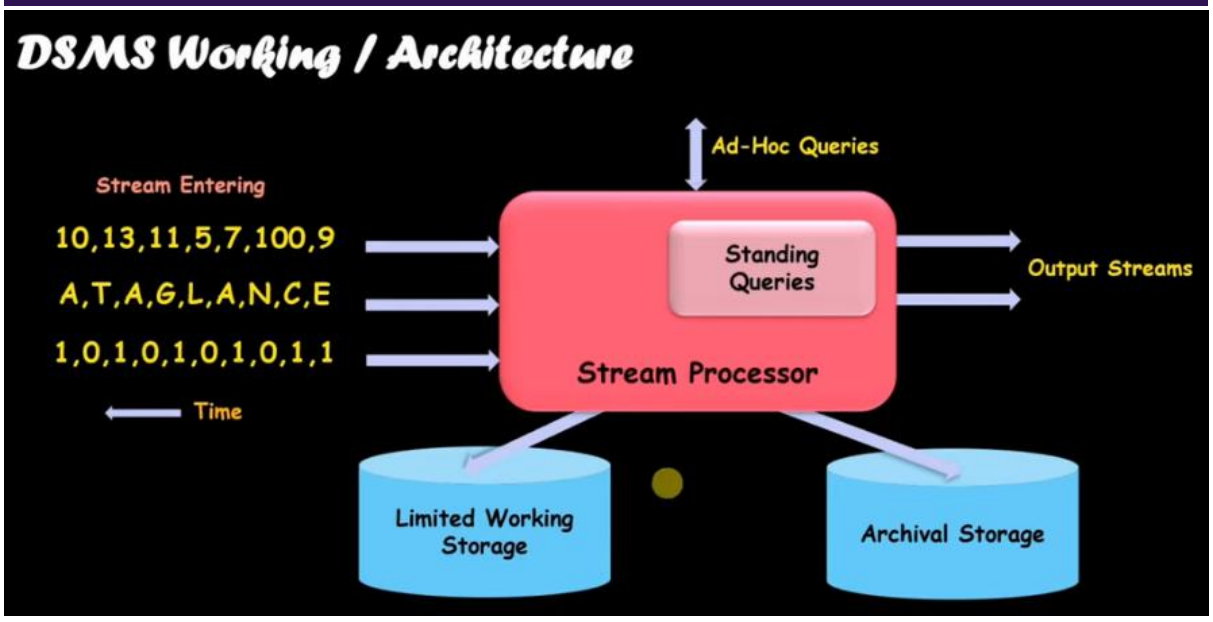
Introduction to Big Data Streams

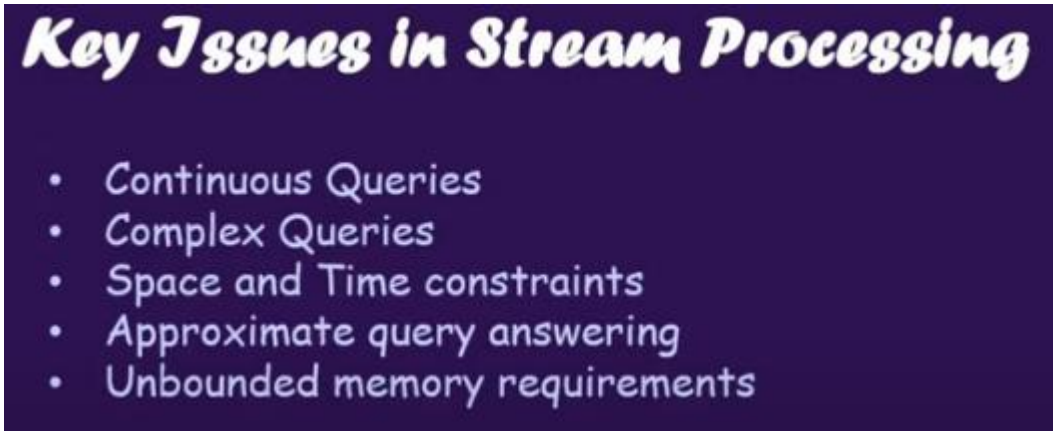


Data Stream Management System

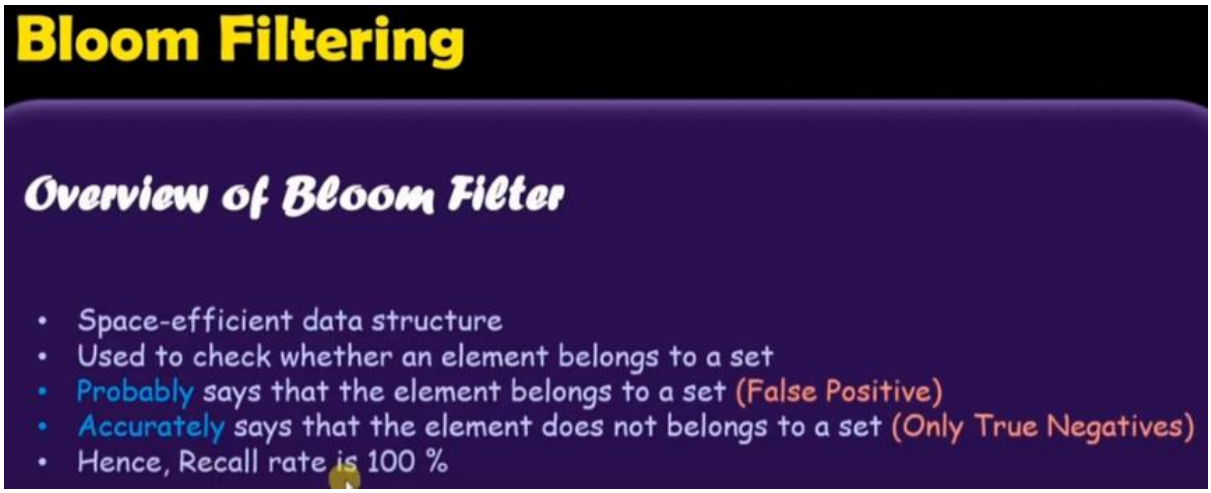


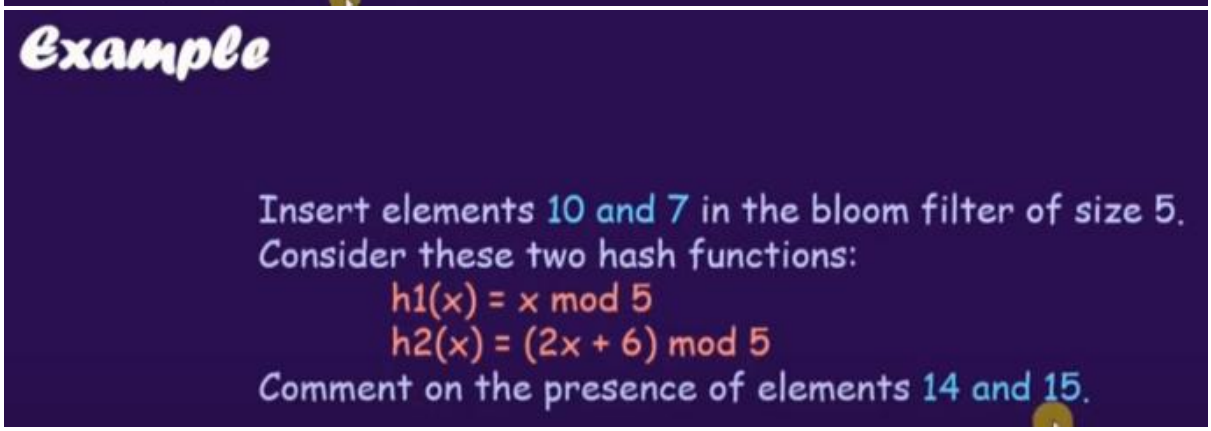






The Bloom Filter





Bloom Filtering

m (Size of Bloom filter) = 5

Hash Functions:
1. h1(x) = x mod 5
2. h2(x) = (2x + 6) mod 5

Insert elements:
- 10
- 7

Element to insert	h1(x)	h2(x)	Bloom Filter
10	10 mod 5 0	(2(10) + 6) mod 5 26 mod 5 1	
7	7 mod 5 2	(2(7) + 6) mod 5 20 mod 5 0	

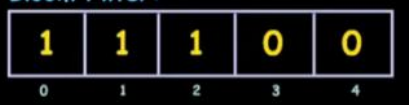
Bloom Filtering

Present elements:
- 10
- 7

Hash Functions:
1. h1(x) = x mod 5
2. h2(x) = (2x + 6) mod 5

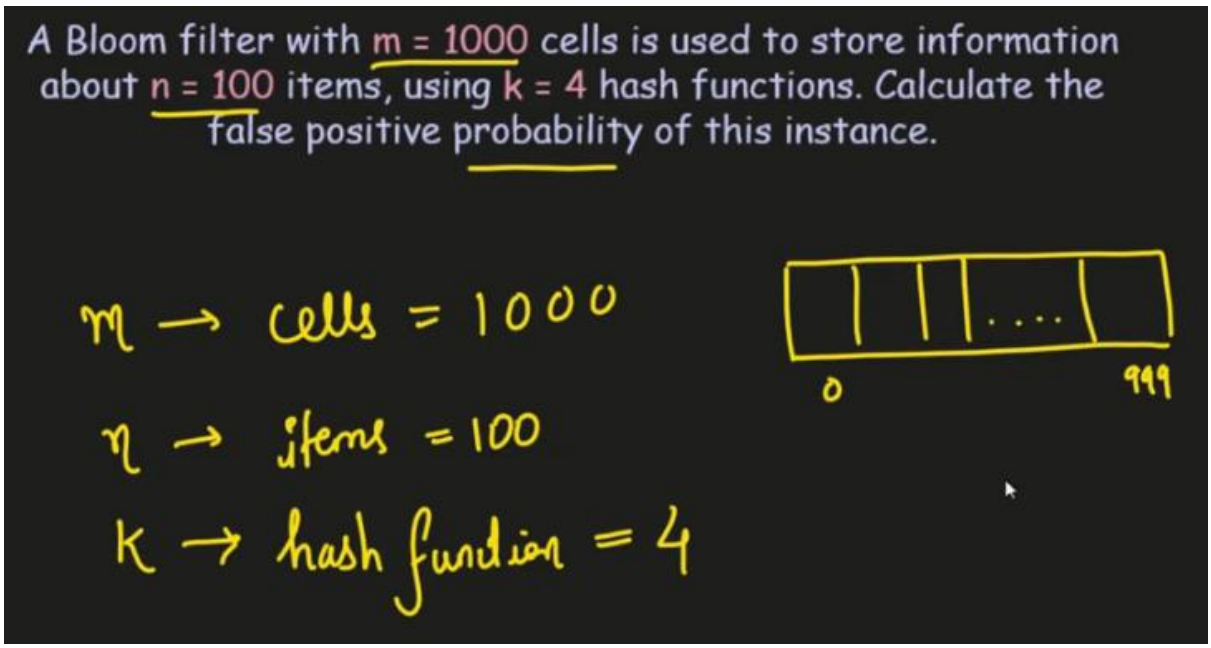
Check for elements:
- 14
- 15

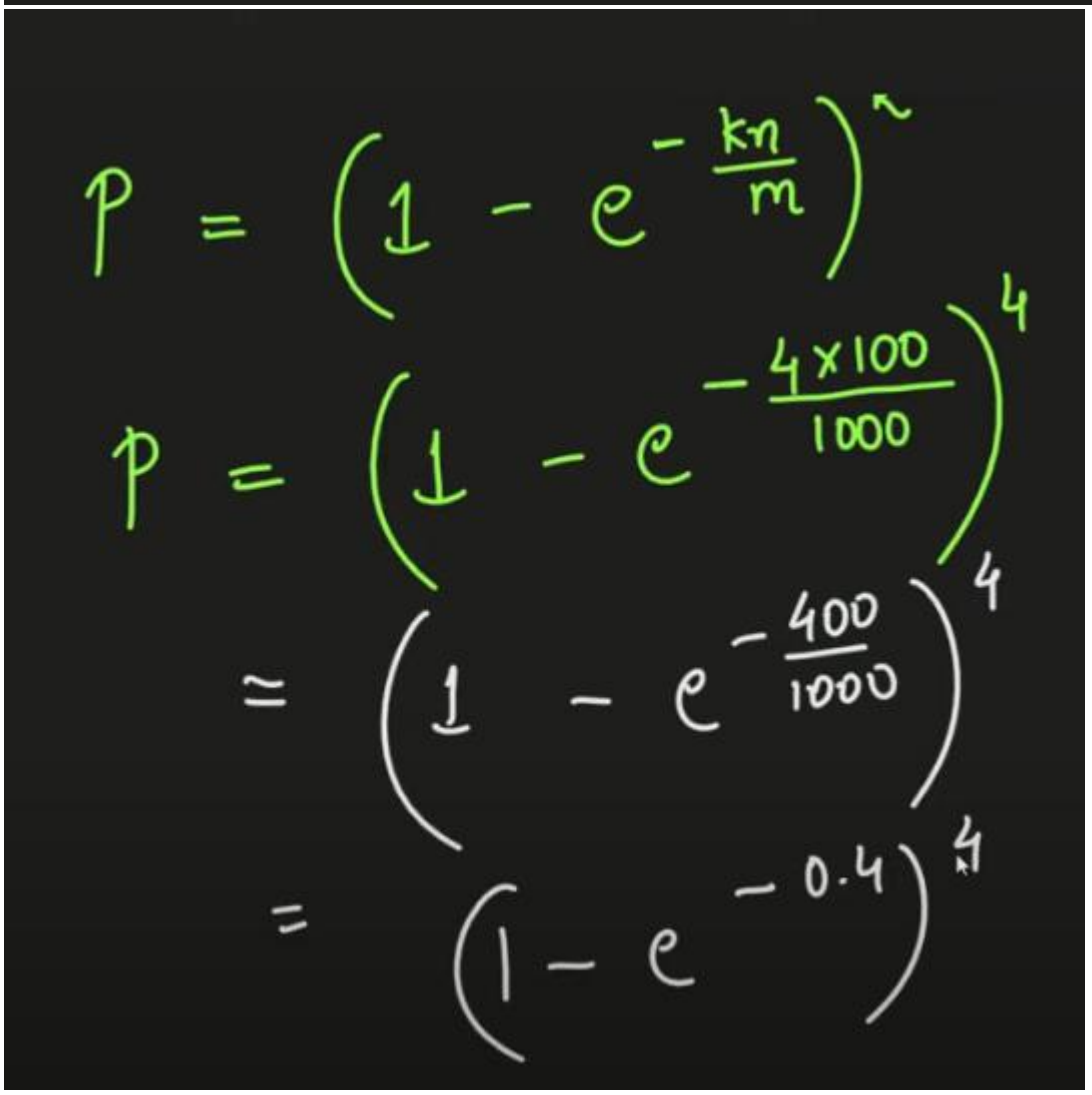
Bloom Filter:



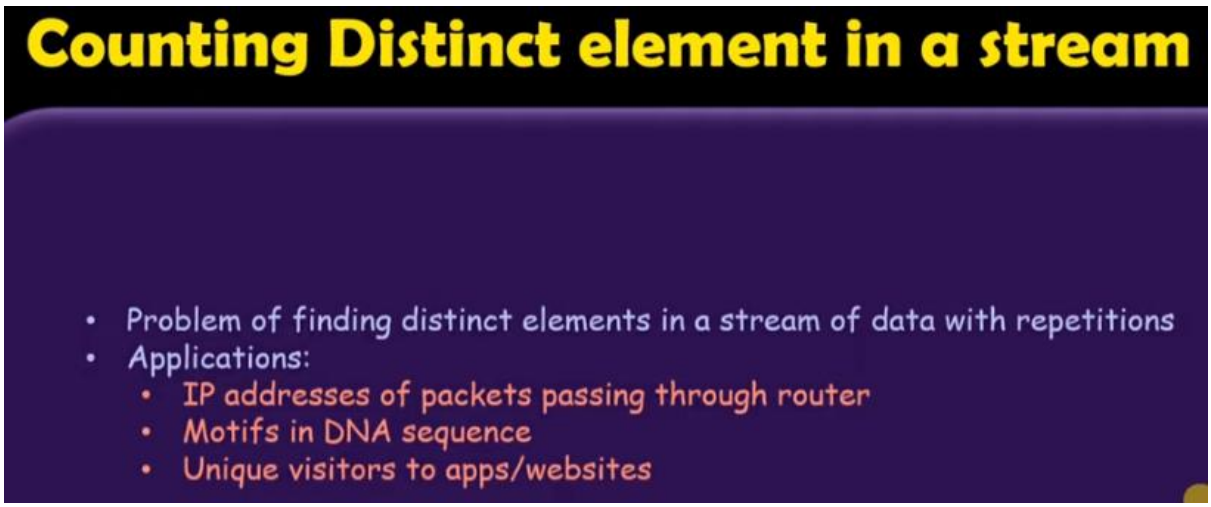
Element	h1(x)	h2(x)	Presence
14	14 mod 5 4	(2(14) + 6) mod 5 34 mod 5 4	At index 4, the value is 0 which means that the element is not present in the set. (Accurately says absence)
15	15 mod 5 0	(2(15) + 6) mod 5 36 mod 5 1	The value is 1 at both the index 0 and 1, which depicts that element is present in the set. But in reality it is not. (False Positive) (Only provides probability of presence)

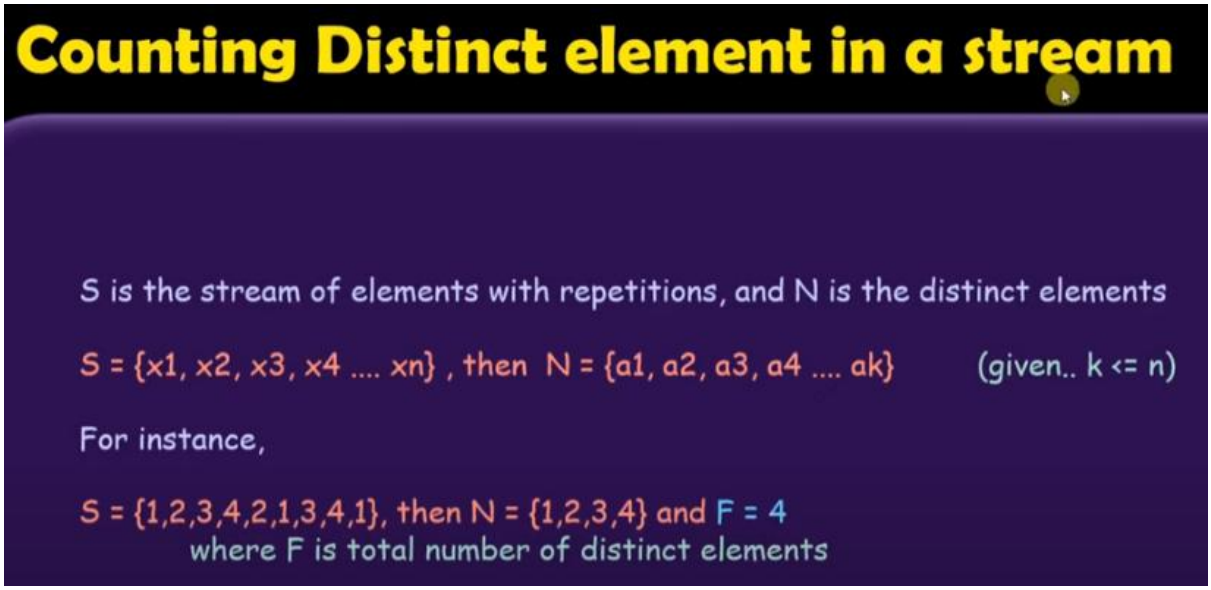
How to find false positive probability of instance in Bloom Filter





Flajolet-Martin Algorithm | Counting distinct elements in a stream





Counting Distinct element in a stream

Naïve Solution/Algorithm

```
SET COUNTER = 0
SET UNIQUE_SET = []
WHILE COUNTER NOT EQUALS LAST ELEMENT INDEX:
        IF CURRENT ELEMENT NOT PRESENT IN UNIQUE_SET:
                ADD CURRENT ELEMENT IN UNIQUE_SET
        INCREMENT THE COUNTER
DISPLAY COUNT OF DISTINCT ELEMENTS : LENGTH(UNIQUE_SET)
```

Flajolet-Martin Algorithm

Overview

- To find the approximate number of distinct elements in a stream
- In a single pass
- Uses very less memory space while executing
- Hence, efficient and robust
- Note: This algorithm is meant to be used when the stream of elements as well as the expected distinct element count is very very large

Pseudocode / Algorithm

1. Select a hash function h(x) so each element in the set is mapped to a value to at least $\log_2 n$ bits
2. Convert this h(x) output to binary_value
3. For each binary_value, find r(binary_value) = length of the trailing zeroes in binary_value
4. Find R = max(r(binary_value))
5. Finally, Approximate count of distinct elements will be 2^R

Flajolet-Martin Algorithm

Pseudocode / Algorithm:

```
SET COUNTER = 0
SET MAX_R = 0
WHILE COUNTER NOT EQUALS LAST ELEMENT INDEX:
        VAL = BINARY OF HASH OUTPUT OF CURRENT ELEMENT
        COUNT NO. OF TRAILING ZEROES IN VAL
        IF COUNT > MAX_R:
                MAX_R = COUNT
        INCREMENT THE COUNTER

DISPLAY APPX. COUNT OF DISTINCT ELEMENTS : 2**(MAX_R)
```

FM Algo uses very less memory as compared to Naive algorithm.

Example

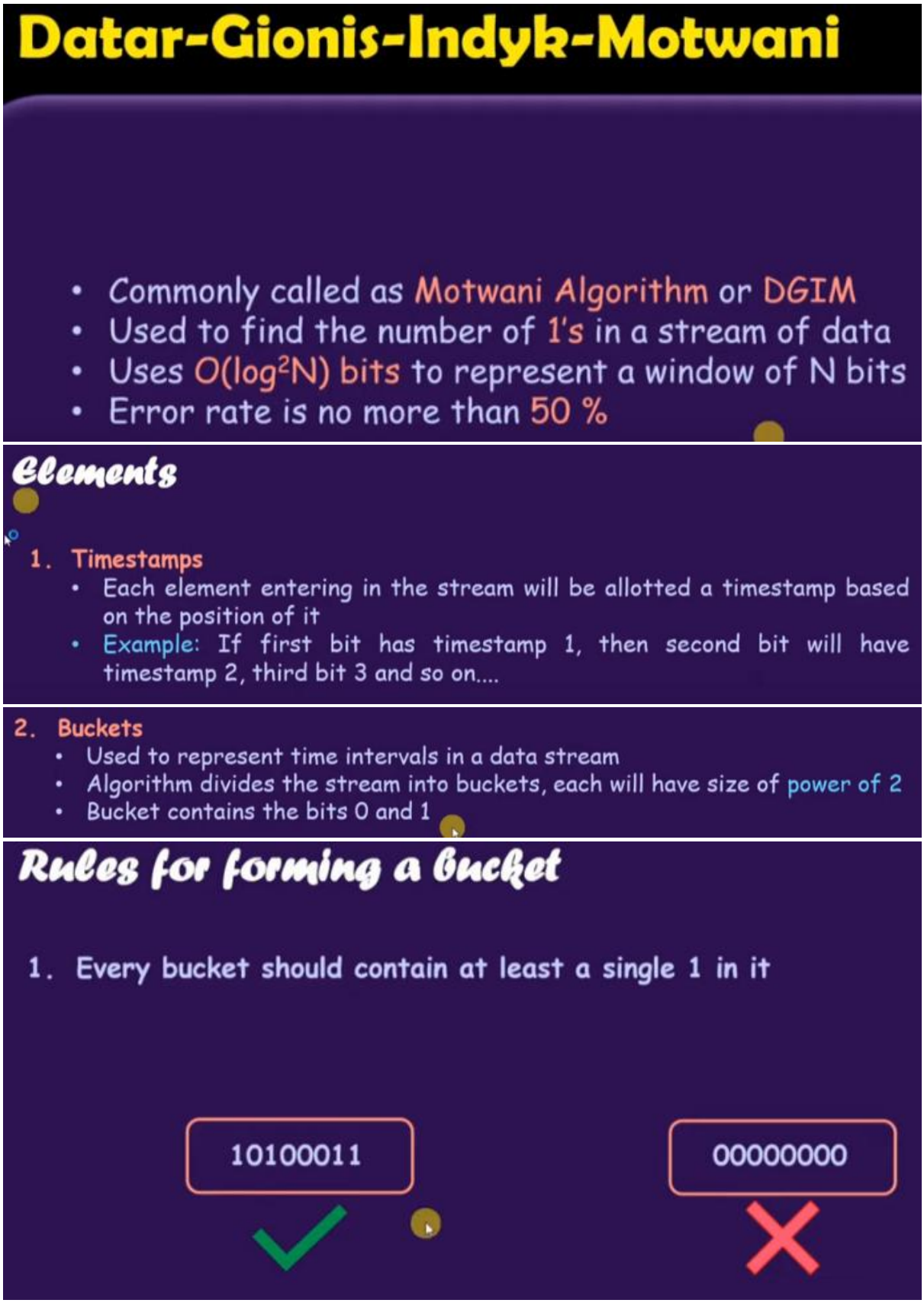
Consider stream, x = [1, 5, 10, 5, 15, 1] and hash function h(x) = x mod 11
Find the count of distinct/unique elements using the FM Algorithm

Flajolet-Martin Algorithm

Consider stream, x = [1, 5, 10, 5, 15, 1] and hash function h(x) = x mod 11
Find the count of distinct/unique elements using the FM Algorithm

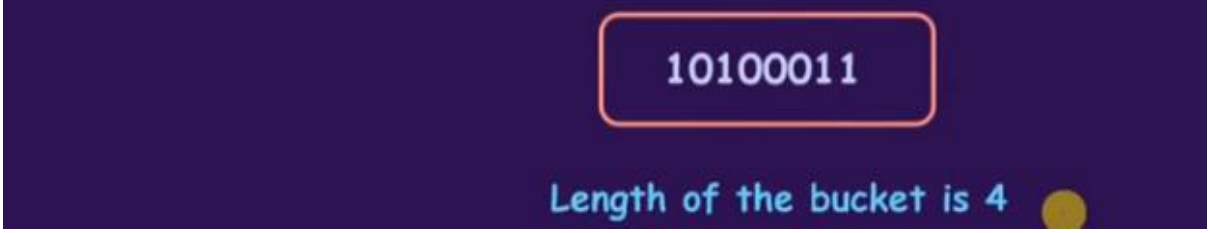
x	h(x)	Binary	Count trailing 0	R = Max of count of trailing 0	Distinct elements count = 2^R
1	1 mod 11 = 1	1	0	2	2^2 = 4
5	5 mod 11 = 5	101	0		
10	10 mod 11 = 10	1010	1		
5	5 mod 11 = 5	101	0		
15	15 mod 11 = 4	100	2		
1	1 mod 11 = 1	1	0		

Datar-Gionis-Indyk-Motwani (DGIM) Algorithm | Counting 1's in a stream



2. Right side of the bucket should strictly start from 1

3. Length of the bucket is equal to the number of 1's in it

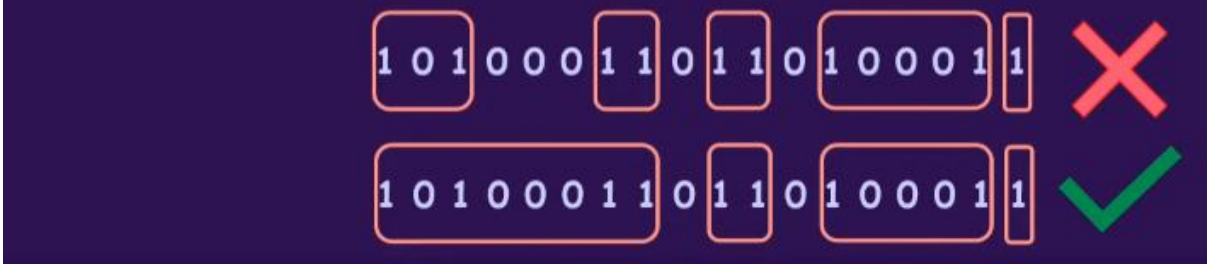


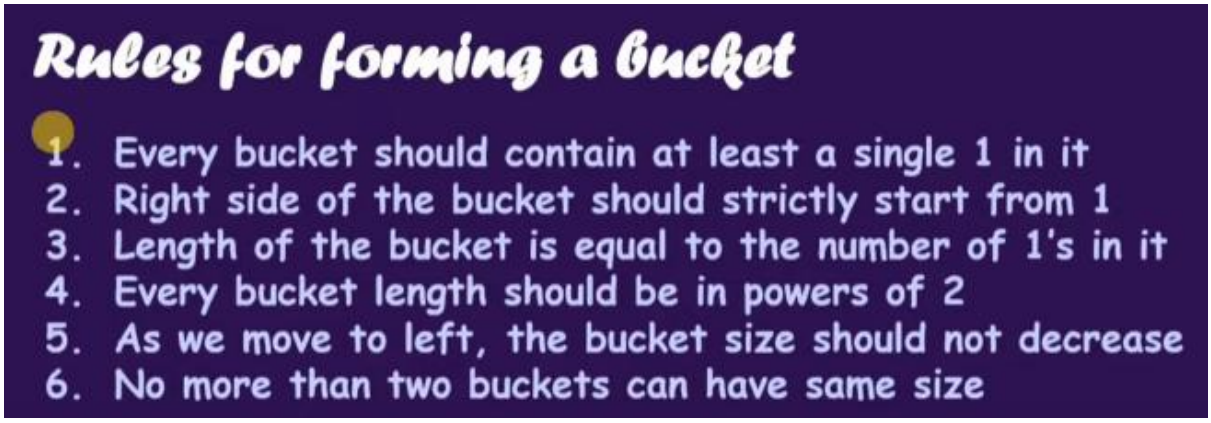
4. Every bucket length should be in powers of 2

$2^0 = 1, 2^1 = 2, 2^2 = 4, 2^3 = 8, 2^4 = 16$ .....

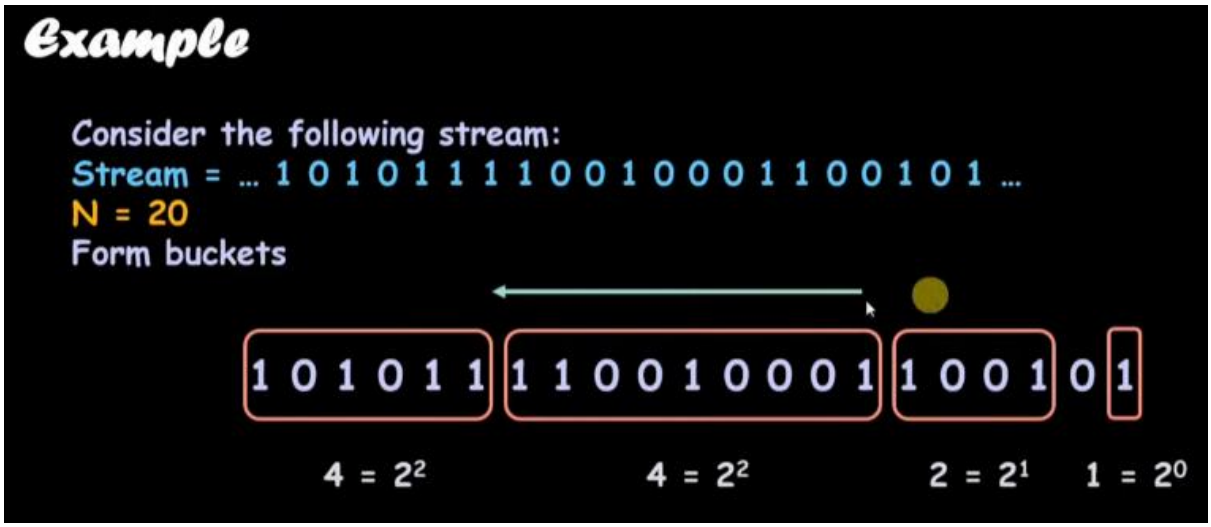
5. As we move to left, the bucket size should not decrease

6. No more than two buckets can have same size

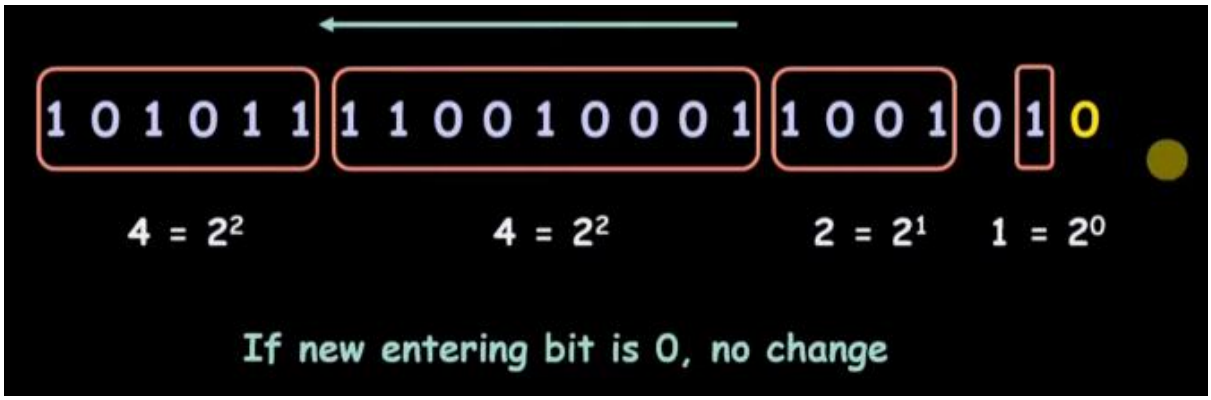




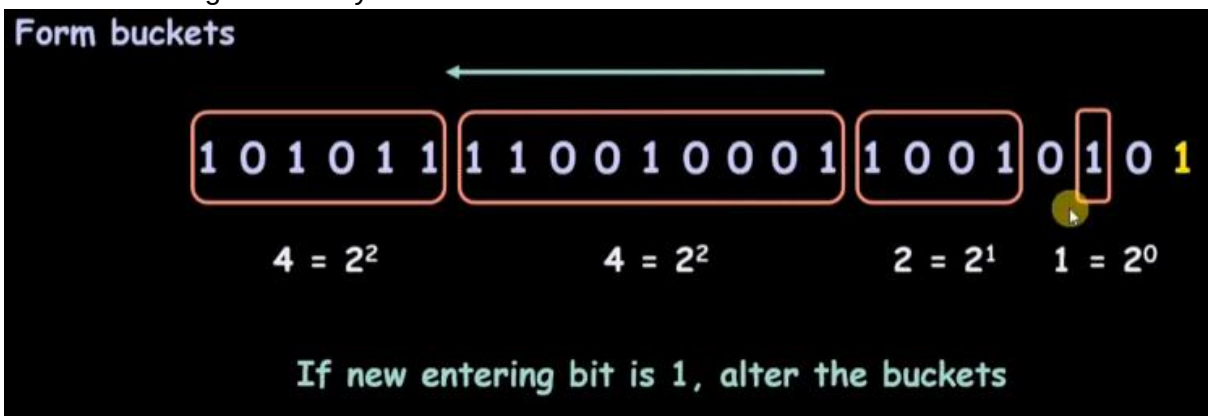
In the 6th rule size and length mean one and the same thing.

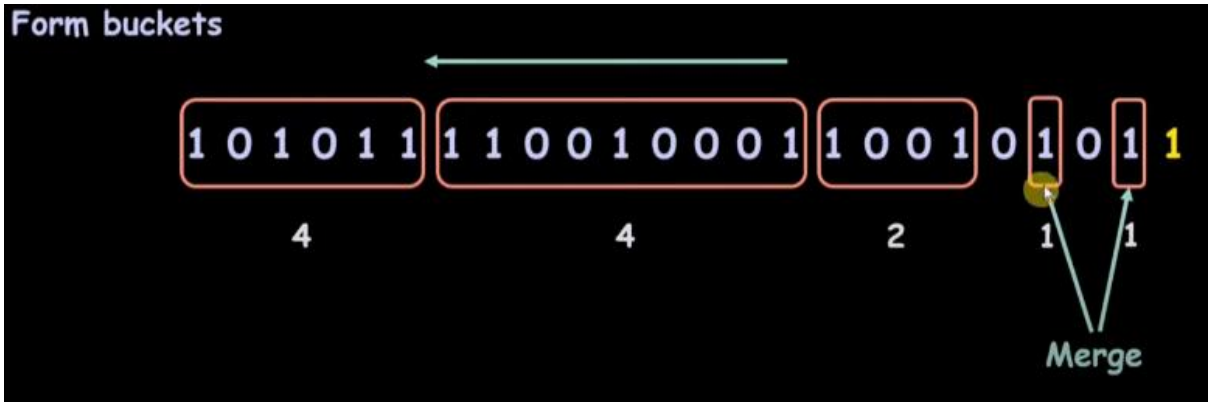


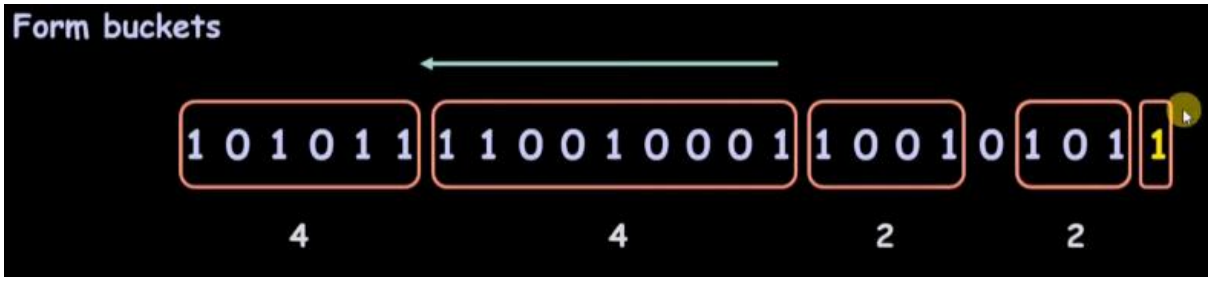
If a new bit enters in the stream

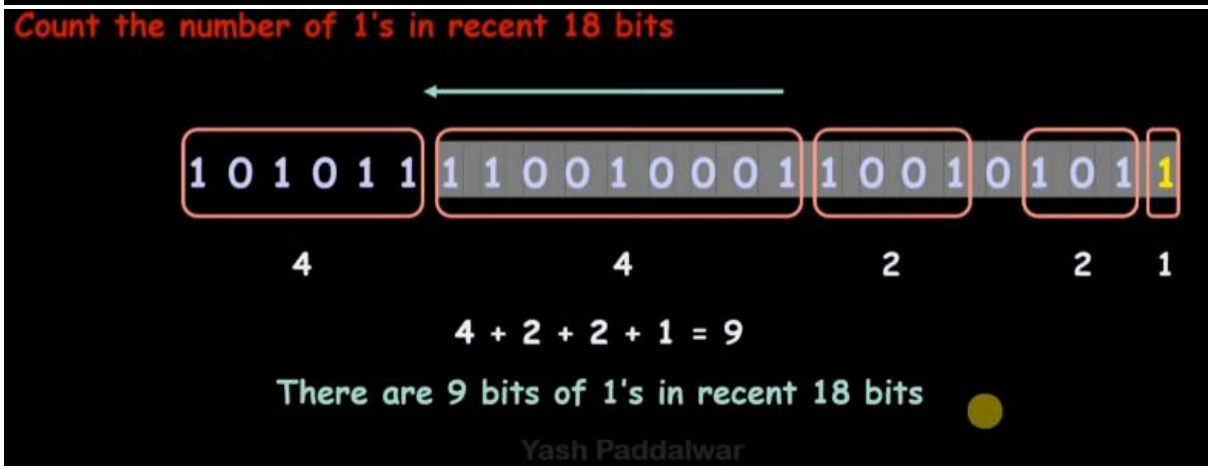


If a new entering bit is one you need to alter the buckets.



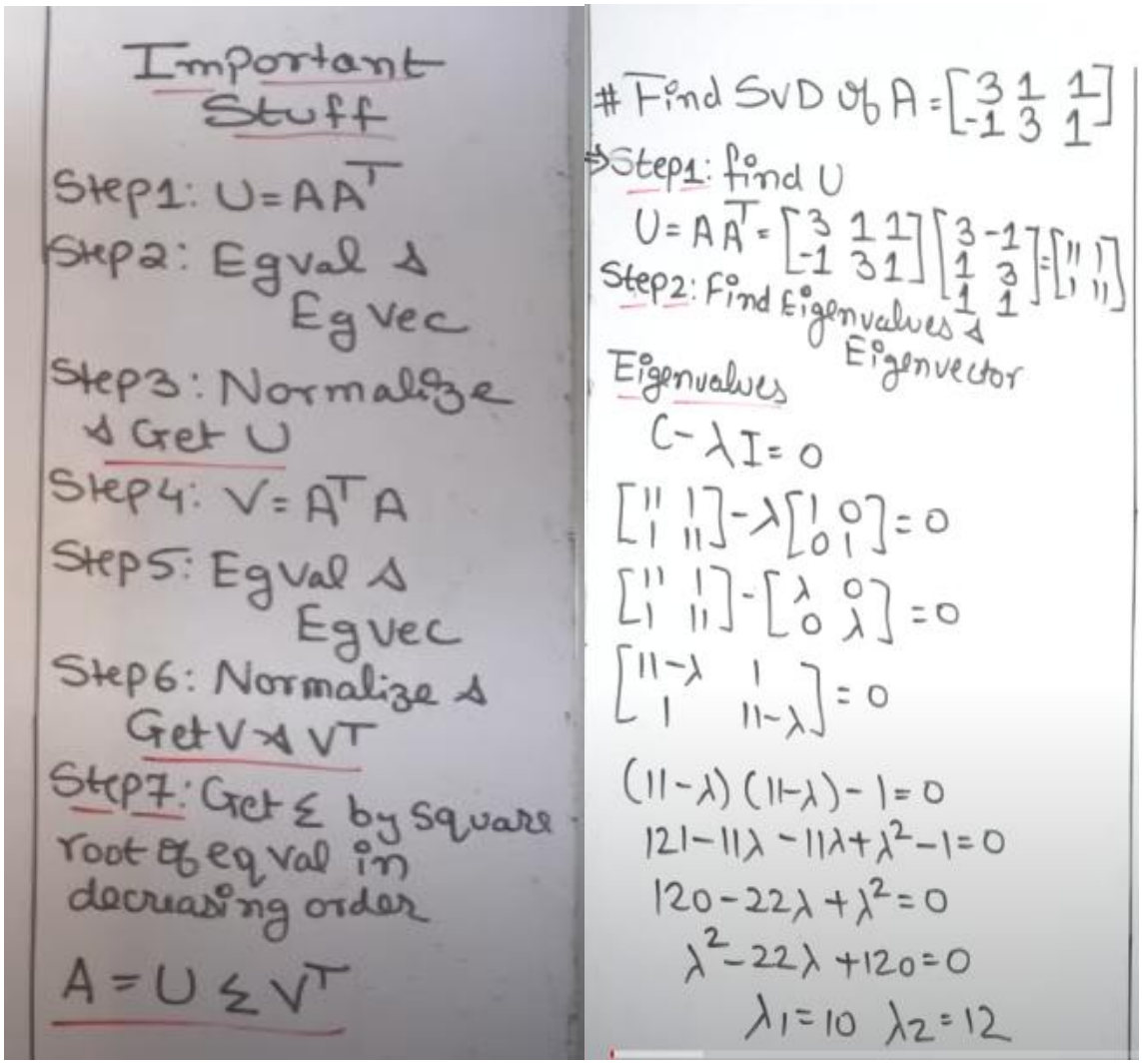


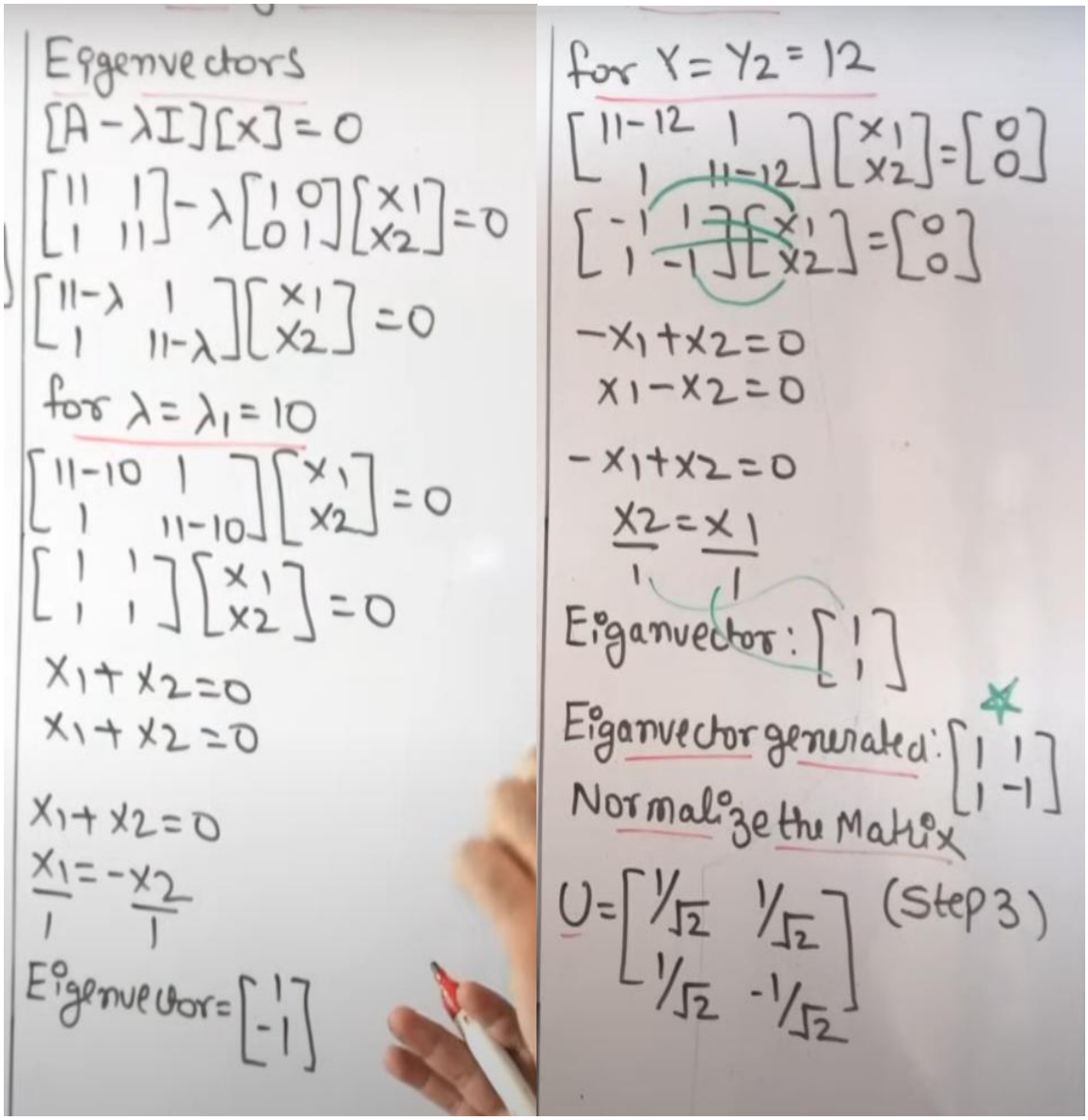




Count the number of buckets and then simply add it.

Singular Value Decomposition





It's lambda and not Y in the above pic.

Now for clubbing these two eigen vectors we first look at their respective eigen values and write the vector with the higher eigen value first and then the vector with lower eigen value. Hence we write the eigen vector of 12 before 10.

Find SVD of $A = \begin{bmatrix} 3 & 1 & 1 \\ -1 & 3 & 1 \end{bmatrix}$

$$U = \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & -1/\sqrt{2} \end{bmatrix}$$

Step4: $V = A^T A$

$$= \begin{bmatrix} 3 & -1 \\ 1 & 3 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 3 & 1 & 1 \\ -1 & 3 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 10 & 0 & 2 \\ 0 & 10 & 4 \\ 2 & 4 & 2 \end{bmatrix}$$

Step5:

$\lambda^3 - (\text{Sum of DE})\lambda^2 + (\text{Sum of minor of DE})\lambda - |A| = 0$

$\lambda^3 - 22\lambda^2 + 120\lambda - 0 = 0$

$\lambda_1 = 0 \quad \lambda_2 = 10 \quad \lambda_3 = 12$

Case1: Eigen Vector for $\lambda_1 = 0$

$$\begin{bmatrix} 10 & 0 & 2 \\ 0 & 10 & 4 \\ 2 & 4 & 2 \end{bmatrix} - \lambda_1 \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = 0$$

$$[A - \lambda_1 I][x] = 0$$

$$\begin{bmatrix} 10 & 0 & 2 \\ 0 & 10 & 4 \\ 2 & 4 & 2 \end{bmatrix} - \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_1 & 0 \\ 0 & 0 & \lambda_1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = 0$$

$$\begin{bmatrix} 10 & 0 & 2 \\ 0 & 10 & 4 \\ 2 & 4 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = 0$$

$10x_1 + 0x_2 + 2x_3 = 0$

$0x_1 + 10x_2 + 4x_3 = 0$

$$\frac{x_1}{\begin{vmatrix} 0 & 2 \\ 10 & 4 \end{vmatrix}} = \frac{-x_2}{\begin{vmatrix} 10 & 2 \\ 0 & 4 \end{vmatrix}} = \frac{x_3}{\begin{vmatrix} 10 & 0 \\ 0 & 10 \end{vmatrix}}$$

$$= \frac{x_1}{-20} = \frac{-x_2}{40} = \frac{x_3}{100}$$

$x_1 = 1 \quad x_2 = 2 \quad x_3 = -5 \quad \therefore X_1 = \begin{bmatrix} 1 \\ 2 \\ -5 \end{bmatrix}$

Case2: Eigenvector for $\lambda_2 = 10$

$[A - \lambda I][x] = 0$

$$\begin{bmatrix} 10 & 0 & 2 \\ 0 & 10 & 4 \\ 2 & 4 & 2 \end{bmatrix} - \begin{bmatrix} \lambda_2 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = 0$$

$$\begin{bmatrix} 0 & 0 & 2 \\ 0 & 0 & 4 \\ 2 & 4 & -8 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = 0$$

$0x_1 + 0x_2 + 4x_3 = 0$

$2x_1 + 4x_2 - 8x_3 = 0$

$$\frac{x_1}{\begin{vmatrix}0&4\\4&-8\end{vmatrix}} = \frac{-x_2}{\begin{vmatrix}0&4\\2&-8\end{vmatrix}} = \frac{x_3}{\begin{vmatrix}0&0\\2&4\end{vmatrix}}$$

$$\frac{x_1}{-16} = \frac{-x_2}{-8} = \frac{x_3}{0}$$

$x_1 = 2 \quad x_2 = -1 \quad x_3 = 0 \quad x_2 = \begin{bmatrix}2\\-1\\0\end{bmatrix}$

Case 3: Eigen vec for $\lambda_3 = 12$

$$\begin{bmatrix}10&0&2\\0&10&4\\2&4&2\end{bmatrix} - \begin{bmatrix}\lambda_3&0&0\\0&\lambda_3&0\\0&0&\lambda_3\end{bmatrix}\begin{bmatrix}x_1\\x_2\\x_3\end{bmatrix} = 0$$

$$\begin{bmatrix}-2&0&2\\0&-2&4\\2&4&-10\end{bmatrix}\begin{bmatrix}x_1\\x_2\\x_3\end{bmatrix} = 0$$

$-2x_1 + 0x_2 + 2x_3 = 0$

$0x_1 - 2x_2 + 4x_3 = 0$

$$\frac{x_1}{\begin{vmatrix}0&2\\-2&4\end{vmatrix}} = \frac{-x_2}{\begin{vmatrix}-2&2\\0&4\end{vmatrix}} = \frac{x_3}{\begin{vmatrix}-2&0\\0&-2\end{vmatrix}}$$

$$\frac{x_1}{4} = \frac{-x_1}{-8} = \frac{x_3}{4}$$

$x_1 = 1 \quad x_2 = 2 \quad x_3 = 1$

$X_3 = \begin{bmatrix}1\\2\\1\end{bmatrix}$

$$V = \begin{bmatrix}1&2&1\\2&-1&2\\1&0&-5\end{bmatrix}$$

Step 6: Normalize

$$V = \begin{bmatrix}1/\sqrt{6}&2/\sqrt{5}&1/\sqrt{30}\\2/\sqrt{6}&-1/\sqrt{5}&2/\sqrt{30}\\1/\sqrt{6}&0&-5/\sqrt{30}\end{bmatrix}$$

$$V^T = \begin{bmatrix}1/\sqrt{6}&2/\sqrt{6}&1/\sqrt{6}\\2/\sqrt{5}&-1/\sqrt{5}&0\\1/\sqrt{30}&2/\sqrt{30}&-5/\sqrt{30}\end{bmatrix}$$

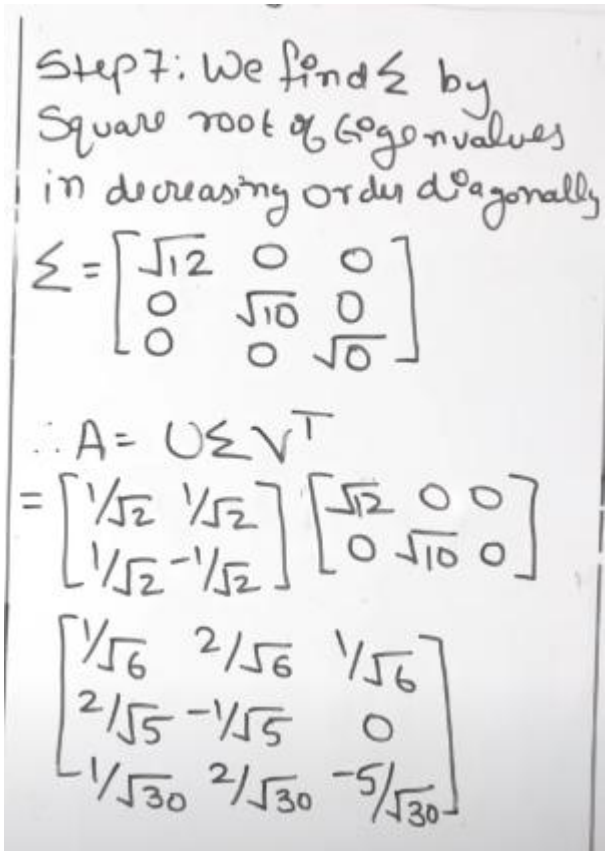
Step 4: $V = A^T A$

Step 5: Eg Val Δ
Eg vec

Step 6: Normalize Δ
Get $V \Delta V^T$

Step 7: Get Σ by square root of eg val in decreasing order

$A = U \Sigma V^T$



Principal Component Analysis Algorithm

Principal Component Analysis – Algorithm

Step 1. Data

- We consider a dataset having n features or variables denoted by $X_1; X_2; \ldots ; X_n$.
- Let there be N examples.
- Let the values of the i^{th} feature X_i be $X_{i1}; X_{i2}; \ldots ; X_{iN}$

Features	Example 1	Example 2	⋯	Example N
X_1	X_{11}	X_{12}	⋯	X_{1N}
X_2	X_{21}	X_{22}	⋯	X_{2N}
⋮				
X_i	X_{i1}	X_{i2}	⋯	X_{iN}
⋮				
X_n	X_{n1}	X_{n2}	⋯	X_{nN}

Principal Component Analysis – Algorithm

Step 2. Compute the means of the variables

Features	Example 1	Example 2	⋯	Example N
X_1	X_{11}	X_{12}	⋯	X_{1N}
X_2	X_{21}	X_{22}	⋯	X_{2N}
⋮				
X_i	X_{i1}	X_{i2}	⋯	X_{iN}
⋮				
X_n	X_{n1}	X_{n2}	⋯	X_{nN}

$$\bar{X}_i = \frac{1}{N}(X_{i1} + X_{i2} + \cdots + X_{iN})$$

Principal Component Analysis – Algorithm

Step 3. Calculate the covariance matrix

Features	Example 1	Example 2	⋯	Example N
X_1	X_{11}	X_{12}	⋯	X_{1N}
X_2	X_{21}	X_{22}	⋯	X_{2N}
⋮				
X_i	X_{i1}	X_{i2}	⋯	X_{iN}
⋮				
X_n	X_{n1}	X_{n2}	⋯	X_{nN}

$$\mathrm{Cov}(X_i, X_j) = \frac{1}{N-1}\sum_{k=1}^{N}(X_{ik} - \bar{X}_i)(X_{jk} - \bar{X}_j)$$

$$S = \begin{bmatrix} \mathrm{Cov}(X_1, X_1) & \mathrm{Cov}(X_1, X_2) & \cdots & \mathrm{Cov}(X_1, X_n) \\ \mathrm{Cov}(X_2, X_1) & \mathrm{Cov}(X_2, X_2) & \cdots & \mathrm{Cov}(X_2, X_n) \\ \vdots & & & \\ \mathrm{Cov}(X_n, X_1) & \mathrm{Cov}(X_n, X_2) & \cdots & \mathrm{Cov}(X_n, X_n) \end{bmatrix}$$

Principal Component Analysis – Algorithm

Step 4. Calculate the eigenvalues and eigenvectors of the covariance matrix

i. **Set up the equation:** This is a polynomial equation of degree **n** in. It has **n** real roots and these roots are the eigenvalues of **S**

$$\det(S - \lambda I) = 0$$

ii. **If $\lambda = \lambda'$ is an eigenvalue**, then the corresponding eigenvector is a vector

$$U = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{bmatrix} \quad \text{such that} \quad (S - \lambda' I)U = 0$$

Principal Component Analysis – Algorithm

Step 4. Calculate the eigenvalues and eigenvectors of the covariance matrix

iii. **We now normalize the eigenvectors**. Given any vector X we normalize it by dividing X by its length. The length (or, the norm) of the vector

$$X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

is defined as

$$\|X\| = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}$$

We compute the n normalised eigenvectors $e_1, e_2, \ldots, e_n$ by

$$e_i = \frac{1}{\|U_i\|} U_i, \quad i = 1, 2, \ldots, n.$$

Principal Component Analysis – Algorithm

Step 5. Derive new data set

- Order the eigenvalues from highest to lowest.
- The unit eigenvector corresponding to the largest eigenvalue is the first principal component.

i) Let the eigenvalues in descending order be $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$ and let the corresponding unit eigenvectors be $e_1, e_2, \ldots, e_n$.

ii) Choose a positive integer p such that $1 \leq p \leq n$.

iii) Choose the eigenvectors corresponding to the eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_p$ and form the following $p \times n$ matrix (we write the eigenvectors as row vectors):

$$F = \begin{bmatrix} e_1^T \\ e_2^T \\ \vdots \\ e_p^T \end{bmatrix}$$

Principal Component Analysis – Algorithm

Step 5. Derive new data set

iv) We form the following $n \times N$ matrix:

$$X = \begin{bmatrix} X_{11} - \bar{X}_1 & X_{12} - \bar{X}_1 & \cdots & X_{1N} - \bar{X}_1 \\ X_{21} - \bar{X}_2 & X_{22} - \bar{X}_2 & \cdots & X_{2N} - \bar{X}_2 \\ \vdots & & & \\ X_{n1} - \bar{X}_n & X_{n2} - \bar{X}_n & \cdots & X_{nN} - \bar{X}_n \end{bmatrix}$$

v) Next compute the matrix:

$$X_{new} = FX.$$

Note that this is a p × N matrix. This gives us a dataset of N samples having p features.

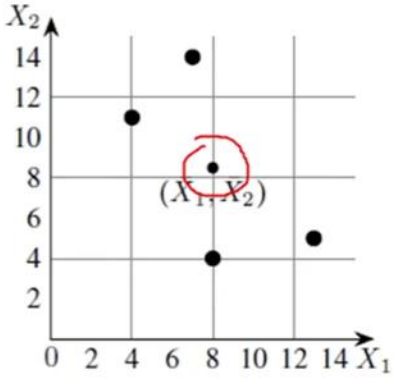
- Given the data in Table, reduce the dimension from 2 to 1 using the Principal Component Analysis (PCA) algorithm.

Feature	Example 1	Example 2	Example 3	Example 4
X_1	4	8	13	7
X_2	11	4	5	14

Step 1: Calculate Mean

F	Ex 1	Ex 2	Ex 3	Ex 4
X_1	4	8	13	7
X_2	11	4	5	14

$$\bar{X}_1 = \frac{1}{4}(4+8+13+7) = 8,$$
$$\bar{X}_2 = \frac{1}{4}(11+4+5+14) = 8.5.$$



Step 2: Calculation of the covariance matrix.

F	Ex 1	Ex 2	Ex 3	Ex 4
X_1	4	8	13	7
X_2	11	4	5	14

$$S = \begin{bmatrix} \mathrm{Cov}(X_1, X_1) & \mathrm{Cov}(X_1, X_2) \\ \mathrm{Cov}(X_2, X_1) & \mathrm{Cov}(X_2, X_2) \end{bmatrix}$$

$$\overline{X_1} = 8$$
$$\overline{X_2} = 8.5$$

$$\begin{aligned} \mathrm{Cov}(X_1, X_1) &= \frac{1}{N-1}\sum_{k=1}^{N}(X_{1k} - \bar{X}_1)(X_{1k} - \bar{X}_1) \\ &= \tfrac{1}{3}\left((4-8)^2 + (8-8)^2 + (13-8)^2 + (7-8)^2\right) \\ &= 14 \end{aligned}$$

Step 2: Calculation of the covariance matrix.

F	Ex 1	Ex 2	Ex 3	Ex 4
X_1	4	8	13	7
X_2	11	4	5	14

$$S = \begin{bmatrix} \mathrm{Cov}(X_1, X_1) & \mathrm{Cov}(X_1, X_2) \\ \mathrm{Cov}(X_2, X_1) & \mathrm{Cov}(X_2, X_2) \end{bmatrix}$$

$$\overline{X_1} = 8$$
$$\overline{X_2} = 8.5$$

$$\begin{aligned} \mathrm{Cov}(X_1, X_2) &= \frac{1}{N-1}\sum_{k=1}^{N}(X_{1k} - \bar{X}_1)(X_{2k} - \bar{X}_2) \\ &= \tfrac{1}{3}((4-8)(11-8.5) + (8-8)(4-8.5) \\ &\quad + (13-8)(5-8.5) + (7-8)(14-8.5) \\ &= -11 \end{aligned}$$

Step 2: Calculation of the covariance matrix.

F	Ex 1	Ex 2	Ex 3	Ex 4
X_1	4	8	13	7
X_2	11	4	5	14

$$\overline{X_1} = 8$$

$$\overline{X_2} = 8.5$$

$$S = \begin{bmatrix} \text{Cov}(X_1, X_1) & \text{Cov}(X_1, X_2) \\ \text{Cov}(X_2, X_1) & \text{Cov}(X_2, X_2) \end{bmatrix}$$

$$\text{Cov}(X_2, X_1) = \text{Cov}(X_1, X_2)$$
$$= -11$$

$$\text{Cov}(X_2, X_2) = \frac{1}{N-1} \sum_{k=1}^{N} (X_{2k} - \bar{X}_2)(X_{2k} - \bar{X}_2)$$
$$= \tfrac{1}{3}\left((11-8.5)^2 + (4-8.5)^2 + (5-8.5)^2 + (14-8.5)^2\right)$$
$$= 23$$

$$S = \begin{bmatrix} \text{Cov}(X_1, X_1) & \text{Cov}(X_1, X_2) \\ \text{Cov}(X_2, X_1) & \text{Cov}(X_2, X_2) \end{bmatrix}$$

$$= \begin{bmatrix} 14 & -11 \\ -11 & 23 \end{bmatrix}$$

Step 3: Eigenvalues of the covariance matrix

F	Ex 1	Ex 2	Ex 3	Ex 4
X_1	4	8	13	7
X_2	11	4	5	14

The characteristic equation of the covariance matrix is,

$$0 = \det(S - \lambda I)$$
$$= \begin{vmatrix} 14-\lambda & -11 \\ -11 & 23-\lambda \end{vmatrix}$$
$$= (14-\lambda)(23-\lambda) - (-11) \times (-11)$$
$$= \lambda^2 - 37\lambda + 201$$

$$\lambda = \tfrac{1}{2}(37 \pm \sqrt{565})$$
$$= 30.3849,\ 6.6151$$
$$= \lambda_1,\ \lambda_2 \quad \text{(say)}$$

$$\overline{X_1} = 8$$

$$\overline{X_2} = 8.5$$

$$S = \begin{bmatrix} 14 & -11 \\ -11 & 23 \end{bmatrix}$$

Step 4: Computation of the eigenvectors

F	Ex 1	Ex 2	Ex 3	Ex 4
X_1	4	8	13	7
X_2	11	4	5	14

$$U = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \qquad \begin{bmatrix} 0 \\ 0 \end{bmatrix} = (S - \lambda I)\, U$$

$$= \begin{bmatrix} 14-\lambda & -11 \\ -11 & 23-\lambda \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

$$= \begin{bmatrix} (14-\lambda)u_1 - 11u_2 \\ -11u_1 + (23-\lambda)u_2 \end{bmatrix}$$

$$(14-\lambda)u_1 - 11u_2 = 0$$
$$-11u_1 + (23-\lambda)u_2 = 0$$

$$\frac{u_1}{11} = \frac{u_2}{14-\lambda} = t$$

$$\overline{X_1} = 8$$

$$\overline{X_2} = 8.5$$

$$S = \begin{bmatrix} 14 & -11 \\ -11 & 23 \end{bmatrix}$$

$$\lambda_1 = 30.3849$$

$$\lambda_2 = 6.6151$$

Step 4: Computation of the eigenvectors

F	Ex 1	Ex 2	Ex 3	Ex 4
X_1	4	8	13	7
X_2	11	4	5	14

$$U = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

$$\frac{u_1}{11} = \frac{u_2}{14-\lambda} = t$$

$$u_1 = 11t, \qquad u_2 = (14-\lambda)t$$

$$U_1 = \begin{bmatrix} 11 \\ 14-\lambda \end{bmatrix}.$$

$$\overline{X_1} = 8$$

$$\overline{X_2} = 8.5$$

$$S = \begin{bmatrix} 14 & -11 \\ -11 & 23 \end{bmatrix}$$

$$\lambda_1 = 30.3849$$

$$\lambda_2 = 6.6151$$

Step 4: Computation of the eigenvectors

F	Ex 1	Ex 2	Ex 3	Ex 4
X_1	4	8	13	7
X_2	11	4	5	14

$$U_1 = \begin{bmatrix} 11 \\ 14-\lambda_1 \end{bmatrix}.$$

- To find a unit eigenvector, we compute the length of U_1 which is given by,

$$\|U_1\| = \sqrt{11^2 + (14-\lambda_1)^2}$$
$$= \sqrt{11^2 + (14-30.3849)^2}$$
$$= 19.7348$$

$$e_1 = \begin{bmatrix} 11/\|U_1\| \\ (14-\lambda_1)/\|U_1\| \end{bmatrix}$$
$$= \begin{bmatrix} 11/19.7348 \\ (14-30.3849)/19.7348 \end{bmatrix}$$
$$= \begin{bmatrix} 0.5574 \\ -0.8303 \end{bmatrix}$$

$$e_2 = \begin{bmatrix} 0.8303 \\ 0.5574 \end{bmatrix}$$

$$\overline{X_1} = 8$$

$$\overline{X_2} = 8.5$$

$$S = \begin{bmatrix} 14 & -11 \\ -11 & 23 \end{bmatrix}$$

$$\lambda_1 = 30.3849$$

$$\lambda_2 = 6.6151$$

Step 5: Computation of first principal components

F	Ex 1	Ex 2	Ex 3	Ex 4
X_1	4	8	13	7
X_2	11	4	5	14

$$e_1^T \begin{bmatrix} X_{1k} - \bar{X}_1 \\ X_{2k} - \bar{X}_2 \end{bmatrix}$$

$$e_1^T \begin{bmatrix} X_{1k} - \bar{X}_1 \\ X_{2k} - \bar{X}_2 \end{bmatrix} = \begin{bmatrix} 0.5574 & -0.8303 \end{bmatrix} \begin{bmatrix} X_{11} - \bar{X}_1 \\ X_{21} - \bar{X}_2 \end{bmatrix}$$

$$= 0.5574(X_{11} - \bar{X}_1) - 0.8303(X_{21} - \bar{X}_2)$$

$$= 0.5574(4 - 8) - 0.8303(11 - 8,5)$$

$$= -4.30535$$

$$e_1 = \begin{bmatrix} 0.5574 \\ -0.8303 \end{bmatrix}$$

$$e_2 = \begin{bmatrix} 0.8303 \\ 0.5574 \end{bmatrix}$$

$$\overline{X_1} = 8$$

$$\overline{X_2} = 8.5$$

$$S = \begin{bmatrix} 14 & -11 \\ -11 & 23 \end{bmatrix}$$

$$\lambda_1 = 30.3849$$

$$\lambda_2 = 6.6151$$

Step 5: Computation of first principal components

F	Ex 1	Ex 2	Ex 3	Ex 4
X_1	4	8	13	7
X_2	11	4	5	14

Feature	Ex 1	Ex 2	Ex 3	Ex 4
X_1	4	8	13	7
X_2	11	4	5	14
First Principle Components	-4.3052	3.7361	5.6928	-5.1238

$$\overline{X_1} = 8$$

$$\overline{X_2} = 8.5$$

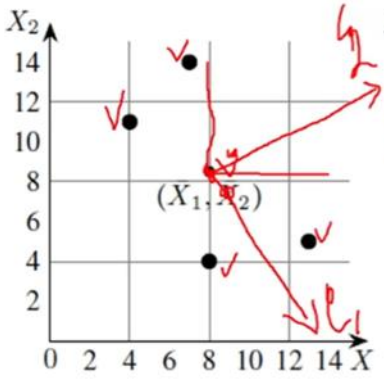
$$S = \begin{bmatrix} 14 & -11 \\ -11 & 23 \end{bmatrix}$$

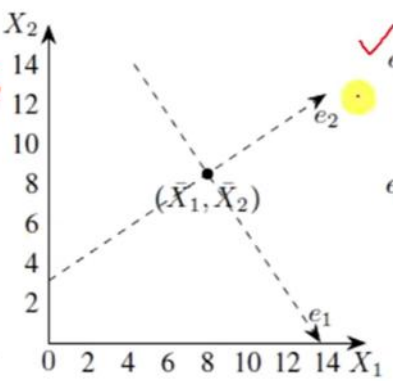
$$\lambda_1 = 30.3849$$

$$\lambda_2 = 6.6151$$

Step 6: Geometrical meaning of first principal components

F	Ex 1	Ex 2	Ex 3	Ex 4
X_1	4	8	13	7
X_2	11	4	5	14





$$e_1 = \begin{bmatrix} 0.5574 \\ -0.8303 \end{bmatrix}$$

$$e_2 = \begin{bmatrix} 0.8303 \\ 0.5574 \end{bmatrix}$$

$$\overline{X_1} = 8$$

$$\overline{X_2} = 8.5$$

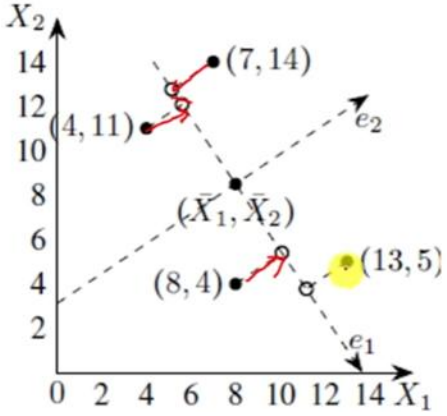
$$S = \begin{bmatrix} 14 & -11 \\ -11 & 23 \end{bmatrix}$$

$$\lambda_1 = 30.3849$$

$$\lambda_2 = 6.6151$$

Step 6: Geometrical meaning of first principal components

F	Ex 1	Ex 2	Ex 3	Ex 4
X_1	4	8	13	7
X_2	11	4	5	14



$$e_1 = \begin{bmatrix} 0.5574 \\ -0.8303 \end{bmatrix}$$

$$\overline{X_1} = 8$$

$$\overline{X_2} = 8.5$$

$$e_2 = \begin{bmatrix} 0.8303 \\ 0.5574 \end{bmatrix} \quad S = \begin{bmatrix} 14 & -11 \\ -11 & 23 \end{bmatrix}$$

$$\lambda_1 = 30.3849$$

$$\lambda_2 = 6.6151$$

PlanetOjas

Principle Component Analysis

→ Apply PCA on the following data
Δ find the Principle Component.

X	2.5	0.5	2.2	1.9	3.1	2.3	2	1	1.5	1.1	$\bar{X} = 1.81$ (1)
Y	2.4	0.7	2.9	2.2	3	2.7	1.6	1.1	1.6	0.9	$\bar{Y} = 1.91$

Calculate the Covariance Matrix

X	$X-\bar{X}$	$(X-\bar{X})(X-\bar{X})$	Y	$Y-\bar{Y}$	$(Y-\bar{Y})(Y-\bar{Y})$
2.5	0.69	0.4761 (2)	2.4	0.49	0.2401 (3)
0.5	-1.31	1.7161	0.7	-1.21	1.4641
2.2	0.39	0.1521	2.9	0.99	0.9801
1.9	0.09	0.0081	2.2	0.29	0.0841
3.1	1.29	1.6641	3	1.09	1.1881
2.3	0.49	0.2401	2.7	0.79	0.6241
2	0.19	0.0361	1.6	-0.31	0.0961
1	-0.81	0.6561	1.1	-0.81	0.6561
1.5	-0.31	0.0961	1.6	-0.31	0.0961
1.1	-0.71	0.5041	0.9	-1.01	1.0201
		5.549			6.449

nent

(4)

X	Y	$(x-\bar{x})$	$(y-\bar{y})$	$(x-\bar{x})(y-\bar{y})$
2.5	2.4	0.69	0.49	0.3381
0.5	0.7	-1.31	-1.21	1.5851
2.2	2.9	0.39	0.99	0.3861
1.9	2.2	0.09	0.29	0.0261
3.1	3	1.29	1.09	1.4061
2.3	2.7	0.49	0.79	0.3871
2	1.6	0.19	-0.31	-0.0589
1	1.1	-0.81	-0.81	0.6561
1.5	1.6	-0.31	-0.31	0.0961
1.1	0.9	-0.71	-1.01	0.7171
				5.539

(1)

1.81
1.91

$$Cov(X,Y) = \sum_{i=1}^{n} \frac{(x_i-\bar{x})(y_i-\bar{y})}{n-1}$$

(5)

$$C = \begin{bmatrix} Cov(x,x) & Cov(x,y) \\ Cov(y,x) & Cov(y,y) \end{bmatrix}$$

$$= \begin{bmatrix} 0.6165 & 0.6154 \\ 0.6154 & 0.7165 \end{bmatrix}$$

PlanetOjas

Principle Ana

→ Apply PCA on the following data
& find the Principle Component.

X	2.5	0.5	2.2	1.9	3.1	2.3	2	1	1.5
Y	2.4	0.7	2.9	2.2	3	2.7	1.6	1.1	1.6

$$\begin{bmatrix} 0.6165 & 0.6154 \\ 0.6154 & 0.7165 \end{bmatrix}$$

$$C - \lambda I = 0$$

$$\begin{bmatrix} 0.6165 & 0.6154 \\ 0.6164 & 0.7165 \end{bmatrix} - \lambda \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = 0$$

$$\begin{bmatrix} 0.6165-\lambda & 0.6154 \\ 0.6154 & 0.7165-\lambda \end{bmatrix} = 0$$

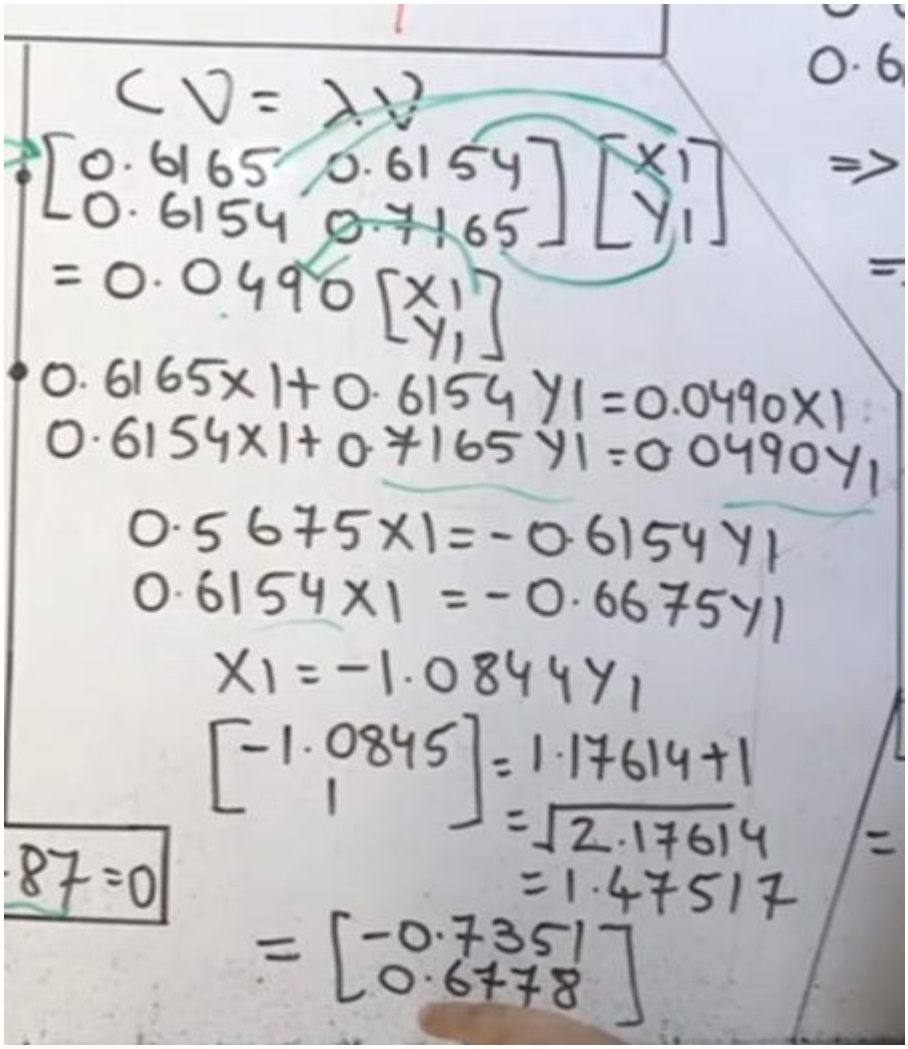
$$(0.6165-\lambda)(0.7165-\lambda) - 0.6154 \times 0.6154 = 0$$

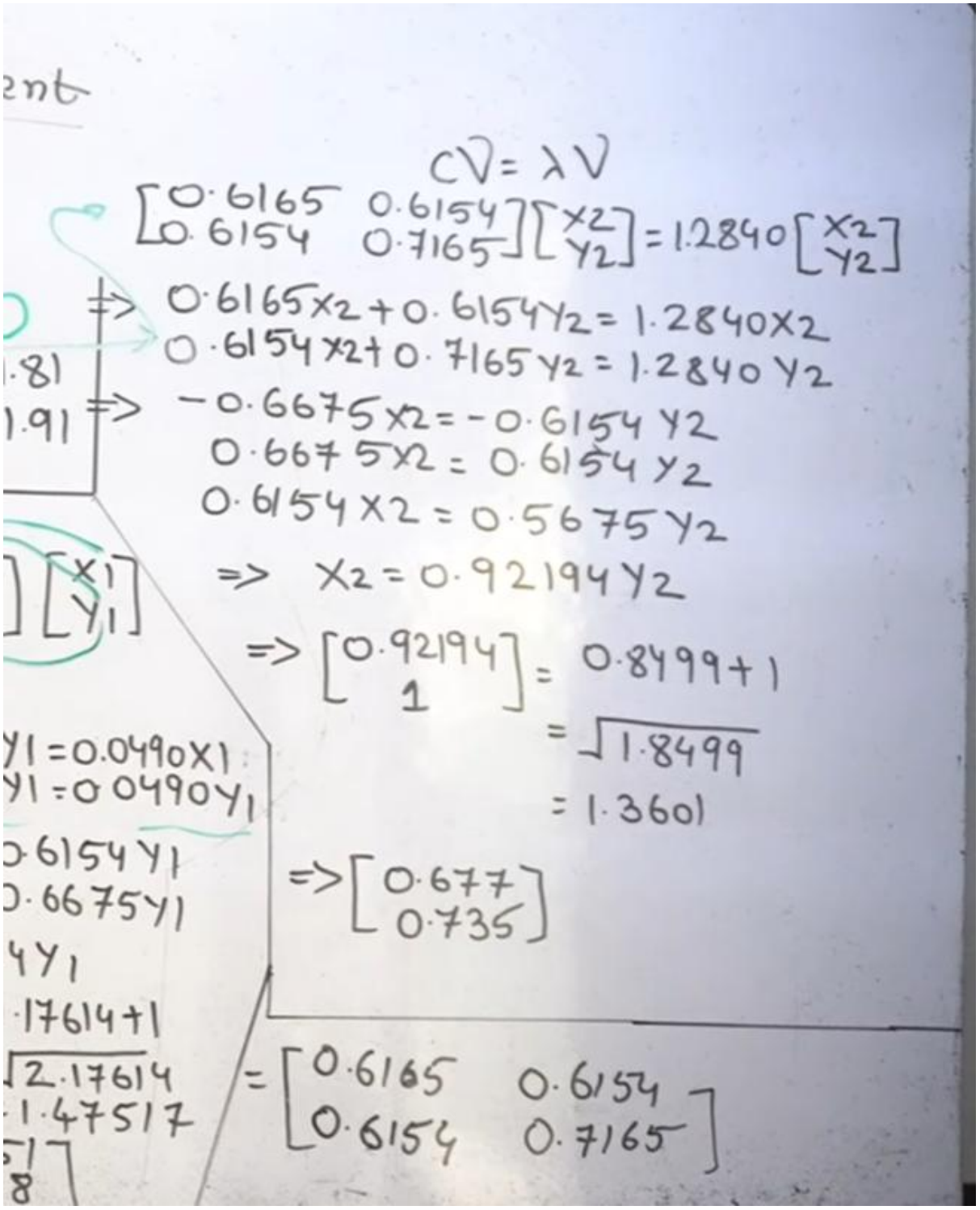
$$0.4417 - 0.6165\lambda + \lambda^2 - 0.7165\lambda - 0.3787 = 0$$

$$\lambda^2 - 1.333\lambda + 0.063 = 0$$

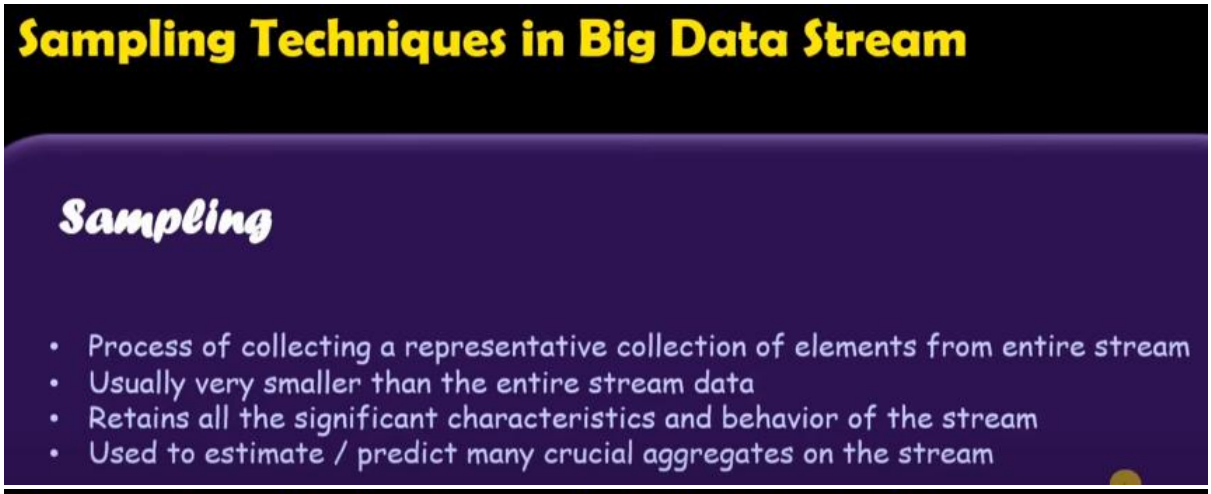
$$\lambda 1 = 1.2840 \quad \lambda 2 = 0.0490$$

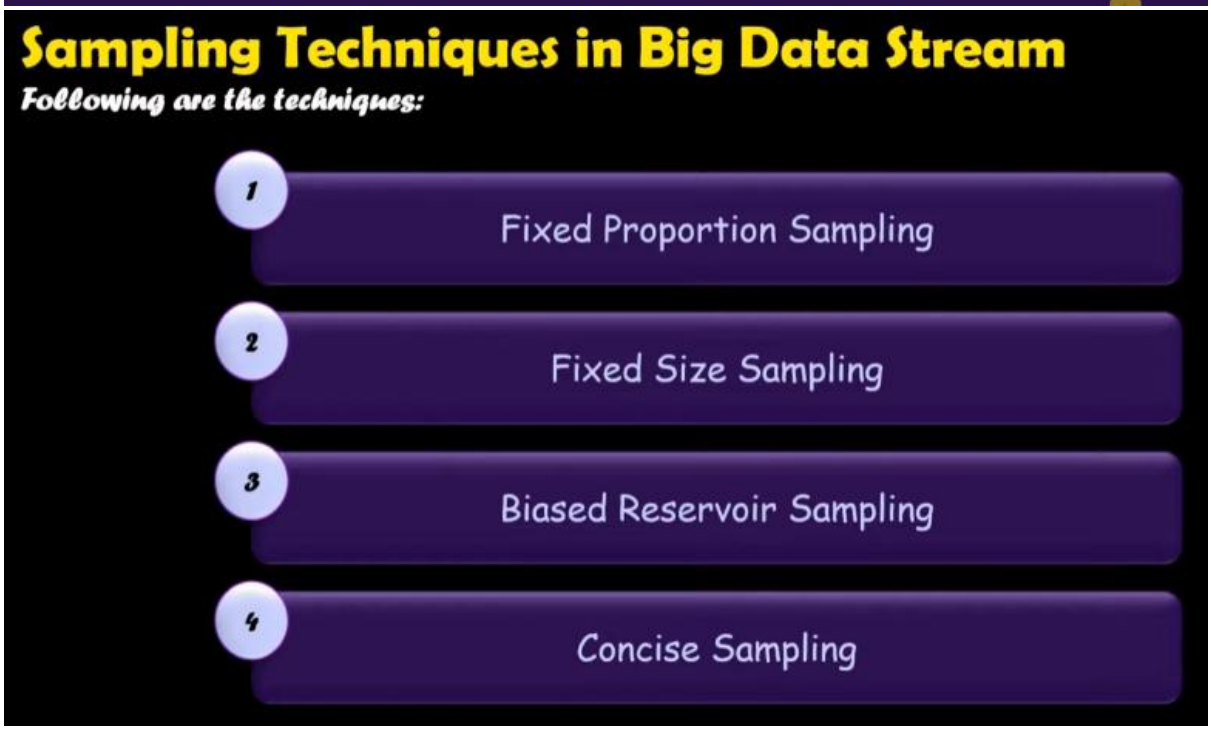
CV

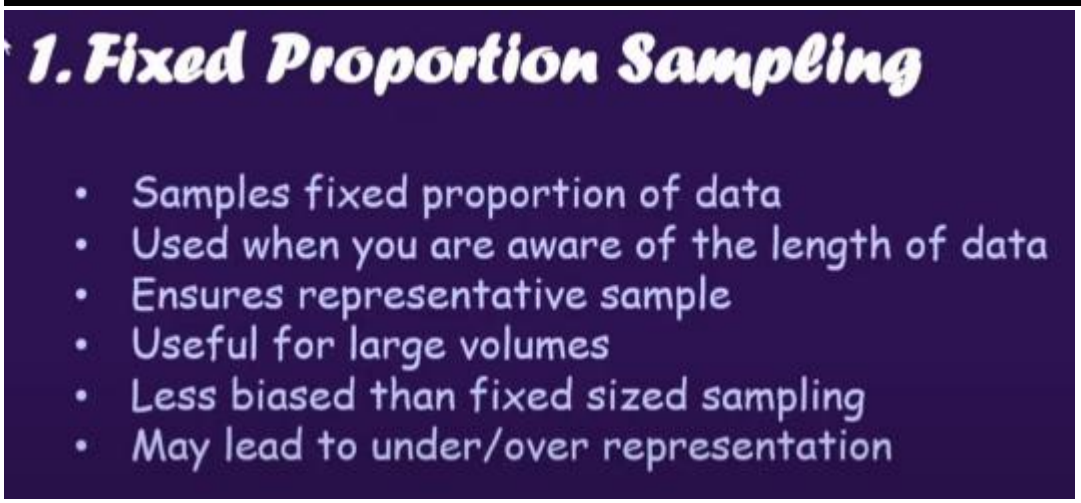

CV = λV
0.6165 0.6154
0.6154 0.7165
X1
Y1
= 0.0490
0.6165X1 + 0.6154Y1 = 0.0490X1
0.6154X1 + 0.7165Y1 = 0.0490Y1
0.5675X1 = -0.6154Y1
0.6154X1 = -0.6675Y1
X1 = -1.0844Y1
-1.0845
1
= 1.17614 + 1
= √2.17614
= 1.47517
= -0.7351
0.6778
=>
0.6
87 = 0


CV = λV
[0.6165 0.6154; 0.6154 0.7165] [X2; Y2] = 1.2840 [X2; Y2]
=> 0.6165X2 + 0.6154Y2 = 1.2840X2
0.6154X2 + 0.7165Y2 = 1.2840Y2
=> -0.6675X2 = -0.6154Y2
0.6675X2 = 0.6154Y2
0.6154X2 = 0.5675Y2
=> X2 = 0.92194Y2
=> [0.92194; 1] = 0.8499 + 1
= √1.8499
= 1.3601
=> [0.677; 0.735]
= [0.6165 0.6154; 0.6154 0.7165]

Sampling Techniques in a Data Stream







Example

A social media platform wants to analyze the sentiments of its users towards a topic. They receive millions of tweets per day and use fixed proportion sampling to select a representative sample. They randomly select 1% of the tweets received each hour, ensuring a representative sample for statistical analysis of user sentiments towards the topic.

2. Fixed Size Sampling

- Samples fixed number of data points.
- Does not guarantee representative sample.
- Useful for reducing data volume.
- Can be biased if data is not randomly distributed
- Less effective when data size increases

Example

Suppose we have a data stream of customer orders for an online store, with 10,000 orders coming in every hour. Using fixed size sampling, we randomly select 1,000 orders from each hour's data stream for analysis, thus reducing the total number of data points to process from 10,000 to 1,000 per hour.

3. Biased Reservoir Sampling

- Used in streams to select a subset of the data in a way that is not uniformly random.
- Can lead to a biased sample that may not be representative of the full dataset.
- The selection of elements is based on a predetermined probability distribution that may be weighted towards certain elements or groups of elements.
- The probability distribution used for biased reservoir sampling may be based on various factors, such as the frequency of occurrence of certain types of data or the importance of certain data points.
- Used when there are constraints on the resources available for sampling, such as limited memory or computational power.
- It is important to carefully consider the potential biases introduced by this sampling technique and adjust the analysis accordingly.

Example

Suppose we have a data stream of product ratings, and we want to select a sample of ratings to estimate the average rating of a product. However, we know that some users tend to give higher ratings than others. Using biased reservoir sampling, we can assign a higher probability of selection to ratings from users who tend to give more accurate ratings. This way, our sample is more likely to represent the true average rating of the product.

4. Concise Sampling

- Goal is to maintain a small reservoir of a fixed size while still achieving representative sampling of the data stream
- Number of samples that can be stored in memory at a given time is limited, which can be a challenge when dealing with large data streams.
- Size of the sample may need to be adjusted based on the amount of memory available to store the data.
- Instead of selecting samples randomly, the sampling algorithm may prioritize choosing samples with unique or representative values of a particular attribute in the data stream

Example

- A bank wants to analyze customer spending habits from a stream of transactions.
- They use concise sampling to choose distinct customer IDs as their attribute.
- The size of the reservoir is limited to 1000 customers.
- They adjust the sample size based on available memory.
- This allows for efficient analysis while maintaining accuracy.

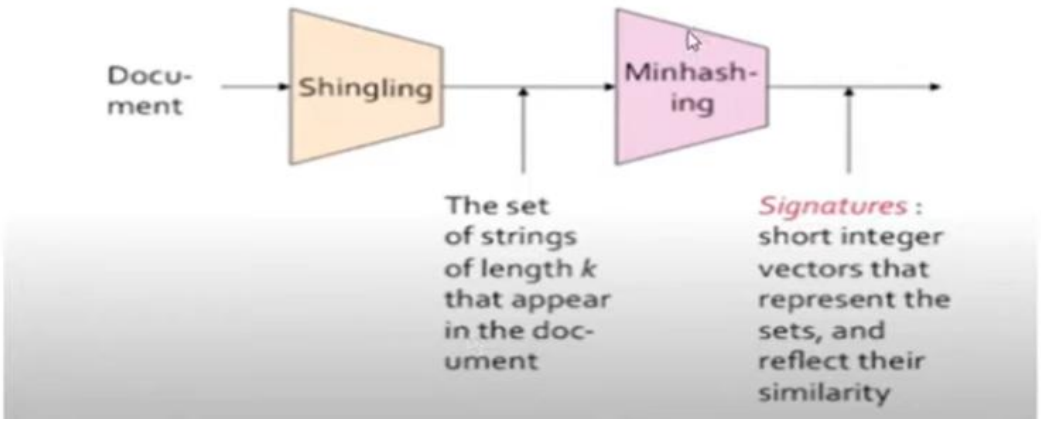
Misra Gries Algorithm

https://beginanalyticsblog.wordpress.com/2017/03/23/your-cheat-sheet-to-the-data-mining-process/#:~:text=In%20our%20example%20above%2C%20m,%2Dvalue%20pairs%20%2F%20memory%20buffers.

Min-Hashing and Jaccard Similarity

Minhashing

- Convert large sets to short signatures while preserving similarity.
- Signatures estimate the Jaccard similarity of sets.



6.1 Collection of Sets as Characteristic Matrix

Element	S_1	S_2	S_3	S_4
a	1	0	0	1
b	0	0	1	0
c	0	1	0	1
d	1	0	1	1
e	0	0	1	0

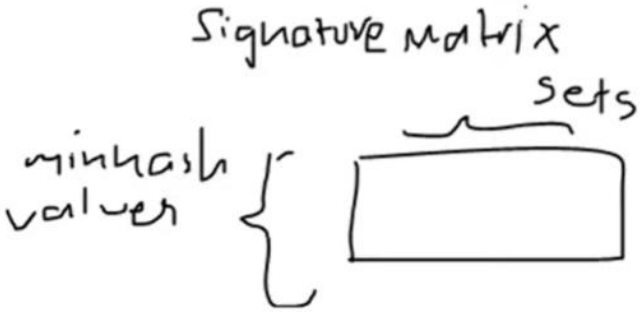
Figure 3.2: A matrix representing four sets

$S_1 = \{a, d\}$ $S_2 = \{c\}$

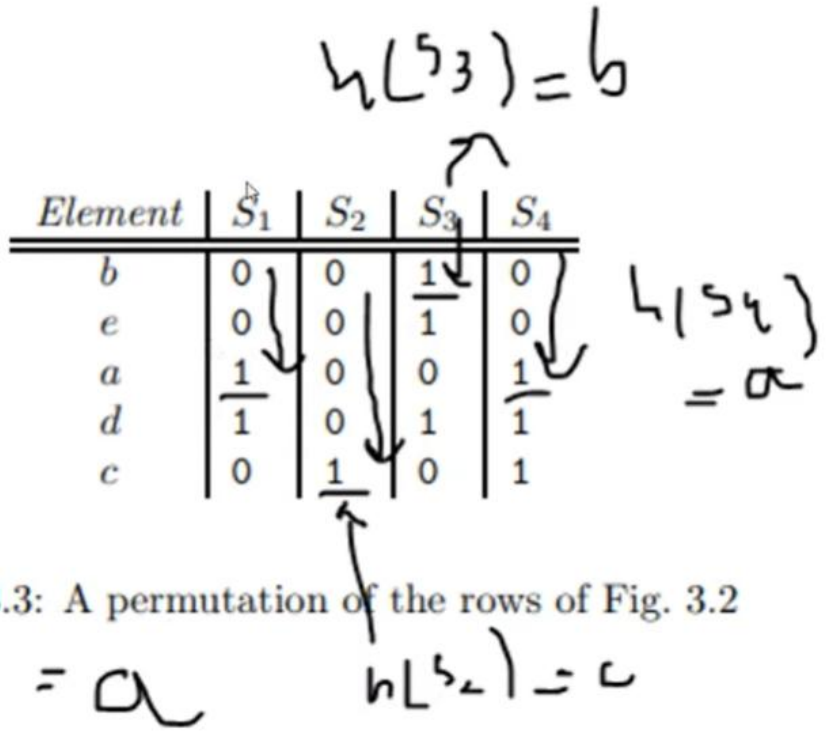
$S_3 = \{b, d, e\}$ $S_4 = \{a, c, d\}$

6.2 Computing Minhash Values

- Rows are permuted randomly.
- Minhashfunc h(C) = the number of the first (in the permuted order) row in which column C has 1.
- Independent hash functions to create signature of each column.



After permuting it randomly.

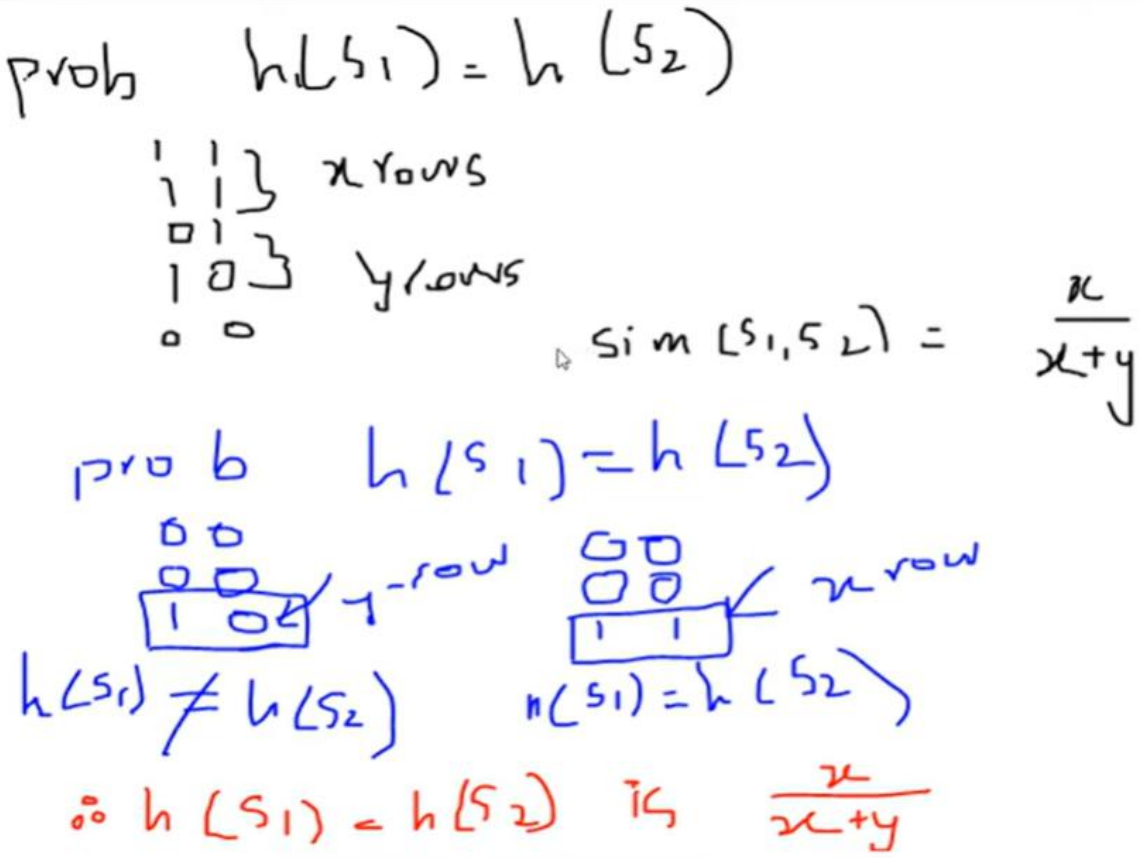


Element	S_1	S_2	S_3	S_4
b	0	0	1	0
e	0	0	1	0
a	1	0	0	1
d	1	0	1	1
c	0	1	0	1

Figure 3.3: A permutation of the rows of Fig. 3.2

6.3 Minhashing and Jaccard Similarity

- The probability that the minhash function for a random permutation of rows produces the same value for two sets equals the Jaccard similarity of those sets.
- X rows (1's in both cols)
- Y rows (1 in one of the columns, 0 in the other)
- Z rows (0 in both cols)



6.4 Algorithm

- Let SIG(i, c) be the element of the signature matrix for the ith hash function and column c. Initially, set SIG(i, c) to ∞ for all i and c

```
For each row r do begin
    For each hash function hi do:
        Compute hi(r);
    For each column c
        If c has 1 in row r
            For each hash function hi do:
                If hi(r) < SIG(i,c) then
                    SIG(i,c) = hi(r)
```

3	1	0	1	0
4	1	0	0	1
7	0	1	0	1
6	0	1	0	1
1	0	1	0	1
2	1	0	1	0
5	1	0	1	0

Signature Matrix:

2	1	2	1

7. Detailed Example

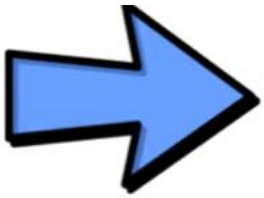
	S_1	S_2	S_3	S_4
h_1	∞	∞	∞	∞
h_2	∞	∞	∞	∞

$h_1(x)$ $h_2(x)$

Row	S_1	S_2	S_3	S_4	$x+1 \mod 5$	$3x+1 \mod 5$
0	1	0	0	1	1	1
1	0	0	1	0	2	4
2	0	1	0	1	3	2
3	1	0	1	1	4	0
4	0	0	1	0	0	3

Figure 3.4: Hash functions computed for the matrix of Fig. 3.2

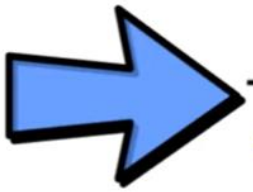
	S_1	S_2	S_3	S_4
h_1	∞	∞	∞	∞
h_2	∞	∞	∞	∞



	S_1	S_2	S_3	S_4
h_1	1	∞	∞	1
h_2	1	∞	∞	1

Row	S_1	S_2	S_3	S_4	$x+1 \mod 5$	$3x+1 \mod 5$
0	1	0	0	1	1	1
1	0	0	1	0	2	4
2	0	1	0	1	3	2
3	1	0	1	1	4	0
4	0	0	1	0	0	3

	S_1	S_2	S_3	S_4
h_1	1	∞	∞	1
h_2	1	∞	∞	1

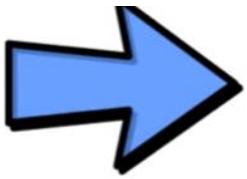


	S_1	S_2	S_3	S_4
h_1	1	∞	2	1
h_2	1	∞	4	1

Row	S_1	S_2	S_3	S_4	$x+1 \mod 5$	$3x+1 \mod 5$
0	1	0	0	1	1	1
1	0	0	1	0	2	4
2	0	1	0	1	3	2
3	1	0	1	1	4	0
4	0	0	1	0	0	3

If the value in the table is less than the value in the signature then only you need to update it in the signature

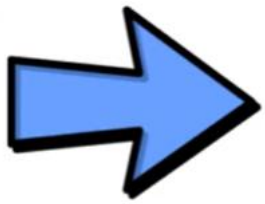
	S_1	S_2	S_3	S_4
h_1	1	∞	2	1
h_2	1	∞	4	1



	S_1	S_2	S_3	S_4
h_1	1	3	2	1
h_2	1	2	4	1

Row	S_1	S_2	S_3	S_4	$x+1 \mod 5$	$3x+1 \mod 5$
0	1	0	0	1	1	1
1	0	0	1	0	2	4
2	0	1	0	1	3	2
3	1	0	1	1	4	0
4	0	0	1	0	0	3

	S_1	S_2	S_3	S_4
h_1	1	3	2	1
h_2	1	2	4	1



	S_1	S_2	S_3	S_4
h_1	1	3	2	1
h_2	0	2	0	0

Row	S_1	S_2	S_3	S_4	$x+1 \mod 5$	$3x+1 \mod 5$
0	1	0	0	1	1	1
1	0	0	1	0	2	4
2	0	1	0	1	3	2
3	1	0	1	1	4	0
4	0	0	1	0	0	3

	S_1	S_2	S_3	S_4
h_1	1	3	2	1
h_2	0	2	0	0

	S_1	S_2	S_3	S_4
h_1	1	3	0	1
h_2	0	2	0	0

Row	S_1	S_2	S_3	S_4	$x+1 \mod 5$	$3x+1 \mod 5$
0	1	0	0	1	1	1
1	0	0	1	0	2	4
2	0	1	0	1	3	2
3	1	0	1	1	4	0
4	0	0	1	0	0	3

	S_1	S_2	S_3	S_4
h_1	1	3	0	1
h_2	0	2	0	0

$sim(S_1, S_2) = 1.0$

	Row	S_1	S_2	S_3	S_4	$x+1 \mod 5$	$3x+1 \mod 5$
*	0	1	0	0	1	1	1
	1	0	0	1	0	2	4
*	2	0	1	0	1	3	2
*	3	1	0	1	1	4	0
	4	0	0	1	0	0	3

Jaccard sim = $\frac{2}{3}$

	S_1	S_2	S_3	S_4
h_1	1	3	0	1
h_2	0	2	0	0

$Sim(S_1, S_2) = \frac{1}{2}$

Row	S_1	S_2	S_3	S_4	$x+1 \mod 5$	$3x+1 \mod 5$
0	1	0	0	1	1	1
1	0	0	1	0	2	4
2	0	1	0	1	3	2
3	1	0	1	1	4	0
4	0	0	1	0	0	3

$Sim(S_1, S_3) = \frac{1}{4}$

	S_1	S_2	S_3	S_4
h_1	1	3	0	1
h_2	0	2	0	0

$Sim(S_1, S_2) = 0$

Row	S_1	S_2	S_3	S_4	$x+1 \mod 5$	$3x+1 \mod 5$
0	1	0	0	1	1	1
1	0	0	1	0	2	4
2	0	1	0	1	3	2
3	1	0	1	1	4	0
4	0	0	1	0	0	3

$Sim(S_1, S_2) = 0$

Scaling with Big Data using Hadoop

https://www.geeksforgeeks.org/introduction-to-hadoop-distributed-file-systemhdfs/

Mapreduce and it’s phases:

https://www.geeksforgeeks.org/mapreduce-understanding-with-real-life-example/

Hadoop Ecosystem Architecture:

https://www.geeksforgeeks.org/hadoop-architecture/

Hive

https://www.geeksforgeeks.org/architecture-and-working-of-hive/
https://www.interviewbit.com/blog/hive-architecture/

Hbase

https://www.geeksforgeeks.org/architecture-of-hbase/
https://www.scaler.com/topics/hadoop/hbase-architecture/
https://www.guru99.com/hbase-architecture-data-flow-usecases.html?gpp&gpp_sid

Pig

https://www.geeksforgeeks.org/introduction-to-apache-pig/
https://data-flair.training/blogs/pig-architecture/

Frequent Item sets and Clustering

Association Rule Mining

- All data used in this technique is categorical.
- There is no dependent variable.
- It uses machine-learning algorithms.
- This technique accepts as input the raw point-of-sale transaction data.
- The output produced is the description of the most frequent affinities among items.
- An example of an association rule would be, "a Customer who bought a laptop computer and virus protection software also bought an extended service plan 70 percent of the time."
- Another example, "A customer who bought bread and milk also bought butter 80% of the time"

Business Applications of Association Rules

- In business environments a pattern or knowledge can be used for many purposes.
- In sales and marketing, it is used for e-commerce site design, online advertising optimization, product pricing, and sales/promotion configurations.
- This analysis can suggest not to put one item on sale at a time, and instead to create a bundle of products promoted as a package to sell other non-selling items.
- In retail environments, it can be used for store design.
- Strongly associated items can be kept close tougher for customer convenience.
- Or they could be placed far from each other so that the customer has to walk the aisles and by doing so is potentially exposed to other items.

Representing Association Rules

- A generic rule is represented between a set X and Y: **X ⇒ Y [S%, C%]**
- **X, Y**: products and/or services
- **X:** Left-hand-side (LHS or Antecedent)
- **Y:** Right-hand-side (RHS or Consequent)
- **S:** Support: how often **X** and **Y** go together in the total transaction set
- **C:** Confidence: how often **Y** goes together with **X**
- There are 100 transactions.
- Out of 100 product X was bought 80 times and product X and Y were bought 60 times.
- Then the association rule is,
- X ⇒ Y [60%, 75%]

Representing Association Rules

$$Rule: \; X \Rightarrow Y$$

$$Support = \frac{frq(X,Y)}{N}$$

$$Confidence = \frac{frq(X,Y)}{frq(X)}$$

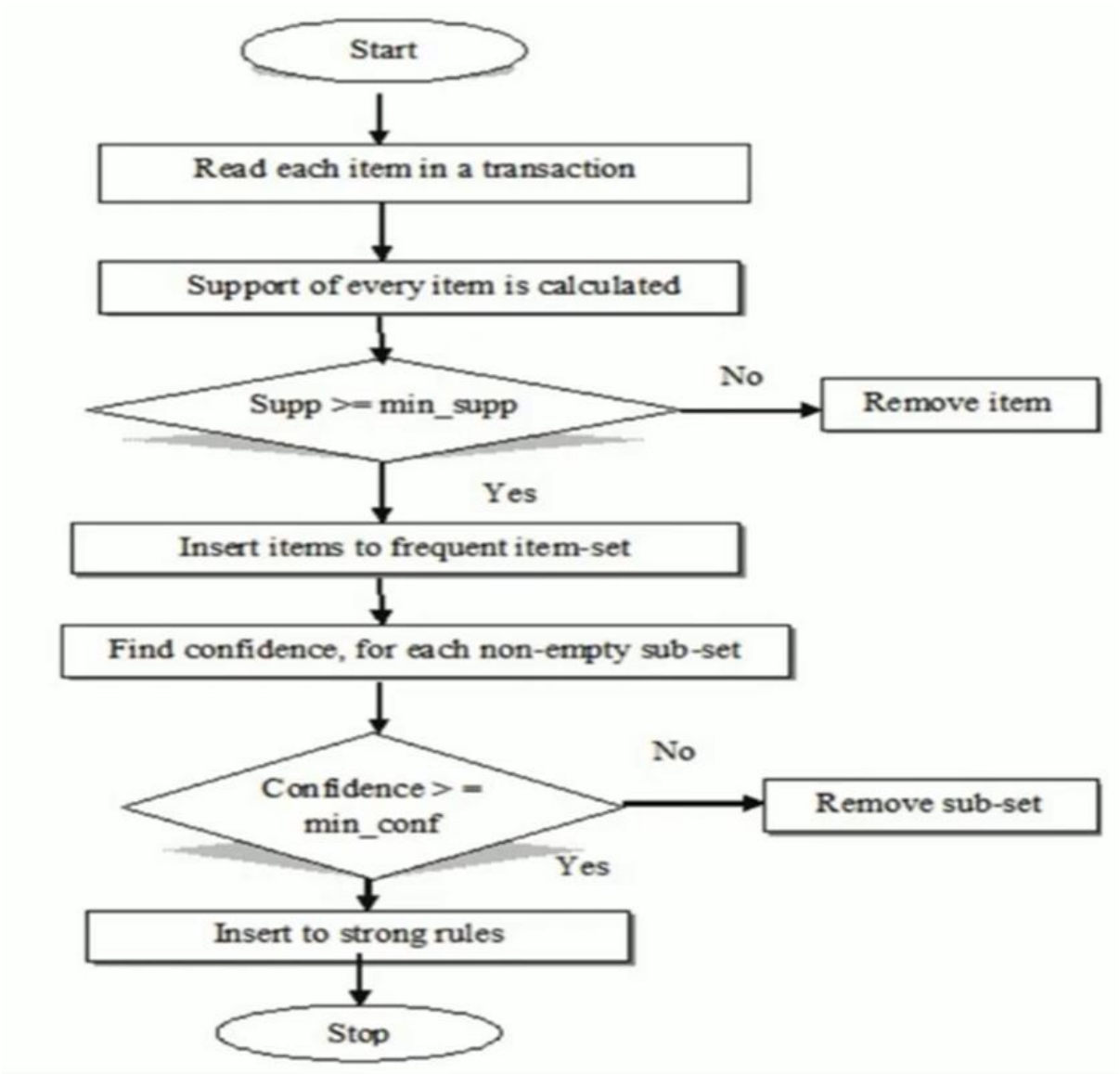
Algorithms for Association Rule

- Not all association rules are interesting and useful, only those that are strong rules and also those that occur frequently.
- In association rule mining, the goal is to find all rules that satisfy the user-specified ***minimum support*** and ***minimum confidence***.
- The resulting sets of rules are all the same irrespective of the algorithm used, that is, given a transaction data set T, a minimum support and a minimum confidence, the set of association rules existing in T is *uniquely determined*.
- The most popular algorithms are Apriori, Eclat, and FP-growth, along with various derivatives and hybrids of the three.

Apriori Algorithm

Apriori Algorithm

- This is the most popular algorithm used for association rule mining.
- A frequent itemset is an itemset whose support is greater than or equal to minimum support threshold.
- The Apriori property is a downward closure property, which means that any subsets of a frequent itemset are also frequent itemsets.
- Thus, if (A,B,C,D) is a frequent itemset, then any subset such as (A,B,C) or (B,D) are also frequent itemsets.
- This uses a bottom-up approach; and the size of frequent subsets is gradually increased, from one-item subsets to two-item subsets, then three-item subsets, and so on.
- Groups of candidates at each level are tested against the data for minimum support.



Apriori algorithm

Association Rules Exercise

Transactions List

1	Milk	Egg	Bread	Butter
2	Milk	Butter	Egg	Ketchup
3	Bread	Butter	Ketchup	
4	Milk	Bread	Butter	
5	Bread	Butter	Cookies	
6	Milk	Bread	Butter	Cookies
7	Milk	Cookies		
8	Milk	Bread	Butter	
9	Bread	Butter	Egg	Cookies
10	Milk	Butter	Bread	
11	Milk	Bread	Butter	
12	Milk	Bread	Cookies	Ketchup

Association Rules Exercise

- Here are a dozen sales transactions.
- The objective is to use this transaction data to find affinities between products, that is, which products sell together often.
- The support level will be set at 33 percent; the confidence level will be set at 50 percent.

Association Rules Exercise

$$Rule: \; X \Rightarrow Y$$

$$Support = \frac{frq(X,Y)}{N}$$

$$Confidence = \frac{frq(X,Y)}{frq(X)}$$

N is total number of transactions

Transactions List

1	Milk	Egg	Bread	Butter
2	Milk	Butter	Egg	Ketchup
3	Bread	Butter	Ketchup	
4	Milk	Bread	Butter	
5	Bread	Butter	Cookies	
6	Milk	Bread	Butter	Cookies
7	Milk	Cookies		
8	Milk	Bread	Butter	
9	Bread	Butter	Egg	Cookies
10	Milk	Butter	Bread	
11	Milk	Bread	Butter	
12	Milk	Bread	Cookies	Ketchup

1-item Sets	Frequency
Milk	9
Bread	10
Butter	10
Egg	3
Ketchup	3
Cookies	5

Frequent 1-item Sets	Frequency
Milk	9
Bread	10
Butter	10
Cookies	5

Frequent 1 – items sets are those items which are bought atleast 33% of times because that's the minimum support level

Transactions List

1	Milk	Egg	Bread	Butter
2	Milk	Butter	Egg	Ketchup
3	Bread	Butter	Ketchup	
4	Milk	Bread	Butter	
5	Bread	Butter	Cookies	
6	Milk	Bread	Butter	Cookies
7	Milk	Cookies		
8	Milk	Bread	Butter	
9	Bread	Butter	Egg	Cookies
10	Milk	Butter	Bread	
11	Milk	Bread	Butter	
12	Milk	Bread	Cookies	Ketchup

2-item Sets	Frequency
Milk, Bread	7
Milk, Butter	7
Milk, Cookies	3
Bread, Butter	9
Butter, Cookies	3
Bread, Cookies	4

Frequent 2-item Sets	Frequency
Milk, Bread	7
Milk, Butter	7
Bread, Butter	9
Bread, Cookies	4

Transactions List

1	Milk	Egg	Bread	Butter
2	Milk	Butter	Egg	Ketchup
3	Bread	Butter	Ketchup	
4	Milk	Bread	Butter	
5	Bread	Butter	Cookies	
6	Milk	Bread	Butter	Cookies
7	Milk	Cookies		
8	Milk	Bread	Butter	
9	Bread	Butter	Egg	Cookies
10	Milk	Butter	Bread	
11	Milk	Bread	Butter	
12	Milk	Bread	Cookies	Ketchup

Milk, Bread, Butter, Cookies

3-item Sets	Frequency
Milk, Bread, Butter	6
Milk, Bread, Cookies	1
Bread, Butter, Cookies	3
Milk, Butter, Cookies	2

Frequent 3-item Sets	Frequency
Milk, Bread, Butter	6

Association Rule Mining - Subset Creation

- Frequent 3-Item Set = I => {Milk, Bread, Butter}

- Non-Empty subset are
  - {{Milk}, {Bread}, {Butter}, {Milk, Bread}, {Milk, Butter}, {Bread, Butter}}

- How to form Association Rule…?
  - For every non-empty subset S of I, the association rule is,
    - **S → (I-S)**
    - **If support(I) / support(S) >= min_confidence**

Association Rule Mining - Subset Creation

- Non-Empty subset are
  - {{Milk}, {Bread}, {Butter}, {Milk, Bread}, {Milk, Butter}, {Bread, Butter}}
  - Min_Support = 30% and Min_Confidence = 60%

- Rule 1: {Milk} → {Bread, Butter} {S=50%, C=66.67%}
  - Support = 6/12 = 50%
  - Confidence = Support (Milk, Bread, Butter)/Support(Milk) = $\frac{6/12}{9/12}$ = 6/9 = 66.67% > 60%
  - Valid

- Rule 2: {Bread} → {Milk, Butter} {S=50%, C=60%}
  - Support = 6/12 = 50%
  - Confidence = Support (Milk, Bread, Butter)/Support(Bread) = 6/10 = 60% >= 60%
  - Valid

- Non-Empty subset are
  - {{Milk}, {Bread}, {Butter}, {Milk, Bread}, {Milk, Butter}, {Bread, Butter}}
  - Min_Support = 30% and Min_Confidence = 60%

- Rule 3: {Butter} → {Milk, Bread} {S=50%, C=60%}
  - Support = 6/12 = 50%
  - Confidence = Support (Milk, Bread, Butter)/Support(Butter) = 6/10 = 60%>=60
  - Valid

- Rule 4: {Milk, Bread} → {Butter} {S=50%, C=85.7%}
  - Support = 6/12 = 50%
  - Confidence = Support (Milk, Bread, Butter)/Support(Milk, Bread) = 6/7 = 85.7% > 60%
  - Valid

- Non-Empty subset are
  - {{Milk}, {Bread}, {Butter}, {Milk, Bread}, {Milk, Butter}, {Bread, Butter}}
  - Min_Support = 30% and Min_Confidence = 60%

- Rule 5: {Milk, Butter} → {Bread} {S=50%, C=85.7%}
  - Support = 6/12 = 50%
  - Confidence = Support (Milk, Bread, Butter)/Support(Milk, Butter) = 6/7 = 85.7% >= 60%
  - Valid

- Rule 6: {Bread, Butter} → {Milk} {S=50%, C=66.67%}
  - Support = 6/12 = 50%
  - Confidence = Support (Milk, Bread, Butter)/Support(Bread, Butter) = 6/9 = 66.67%>=60
  - Valid

Consider the following transactions.
Apply the association rule mining to get the association rules with min support of 2 and confidence of 50%

TID	List of Items IDs
T100	I1, I2, I5
T200	I2, I4
T300	I2, I3
T400	I1, I2, I4
T500	I1, I3
T600	I2, I3
T700	I1, I3
T800	I1, I2, I3, I5
T900	I1, I2, I3

Transactions List

TID	List of Items IDs
T100	I1, I2, I5
T200	I2, I4
T300	I2, I3
T400	I1, I2, I4
T500	I1, I3
T600	I2, I3
T700	I1, I3
T800	I1, I2, I3, I5
T900	I1, I2, I3

1-item Sets	Frequency
I1	6
I2	7
I3	5
I4	4
I5	2

Frequent 1-item Sets	Frequency
I1	6
I2	7
I3	5
I4	4

Transactions List

TID	List of Items IDs
T100	I1, I2, I5
T200	I2, I4
T300	I2, I3
T400	I1, I2, I4
T500	I1, I3
T600	I2, I3
T700	I1, I3
T800	I1, I2, I3, I5
T900	I1, I2, I3

2-item Sets	Frequency
I1, I2	4
I1, I3	4
I1, I4	1
I1, I5	2
I2, I3	3
I2, I4	2
I2, I5	2
I3, I4	0
I3, I5	1
I4, I5	0

Frequent 2-item Sets	Frequency
I1, I2	4
I1, I3	4
I1, I5	2
I2, I3	3
I2, I4	2
I2, I5	2

Transactions List

TID	List of Items IDs
T100	I1, I2, I5
T200	I2, I4
T300	I2, I3
T400	I1, I2, I4
T500	I1, I3
T600	I2, I3
T700	I1, I3
T800	I1, I2, I3, I5
T900	I1, I2, I3

I1, I2, I3, I4, I5

3-item Sets	Frequency
I1, I2, I3	2
I1, I2, I4	0
I1, I2, I5	2
I1, I3, I4	1
I1, I3, I5	1
I1, I4, I5	0
I2, I3, I4	0
I2, I3, I5	1
I2, I4, I5	0
I3, I4, I5	0

Frequent 3-item Sets	Frequency
I1, I2, I3	2
I1, I2, I5	2

Transactions List

TID	List of Items IDs
T100	I1, I2, I5
T200	I2, I4
T300	I2, I3
T400	I1, I2, I4
T500	I1, I3
T600	I2, I3
T700	I1, I3
T800	I1, I2, I3, I5
T900	I1, I2, I3

I1, I2, I3, I5

4-item Sets	Frequency
I1, I2, I3, I5	1

Frequent 4-item Sets	Frequency
Not Possible	

- Frequent 3-Item Set = I => {1, 2, 3} and {1, 2, 5}
- Min_Support = 2 = 2/9 = 22.22% and Min_Confidence = 50%

- Non-Empty subset are
  - {{1}, {2}, {3}, {1, 2}, {1, 3}, {2, 3}}
  - {{1}, {2}, {5}, {1, 2}, {1, 5}, {2, 5}}

- How to form Association Rule…?
  - For every non-empty subset S of I, the association rule is,
    - **S → (I-S)**
    - **If support(I) / support(S) >= min_confidence**

- Frequent 3-Item Set = I => {1, 2, 3}
- Non-Empty subset are
  - {{1}, {2}, {3}, {1, 2}, {1, 3}, {2, 3}}

- Rule 1: {1} → {2, 3} {S= 22.22 %, C=33.34%}
  - Support = 2/9 = 22.22%
  - Confidence = Support (1, 2, 3)/Support(1) = $\frac{2/9}{6/9}$ = 2/6 = 33.34% < 50%
  - Invalid Rule

- Rule 2: {2} → {1, 3} {S= 22.22 %, C=28.57%}
  - Support = 2/9 = 22.22 %
  - Confidence = Support (1, 2, 3)/Support(2) = 2/7 = 28.57% < 50%
  - Invalid Rule

- Frequent 3-Item Set = I => {1, 2, 3}
- Non-Empty subset are
  - {{1}, {2}, {3}, {1, 2}, {1, 3}, {2, 3}}

- Rule 3: {3} → {1, 2} {S= 22.22 %, C=40%}
  - Support = 2/9 = 22.22 %
  - Confidence = Support (1, 2, 3)/Support(3) = 2/5 = 40% < 50%
  - Invalid Rule

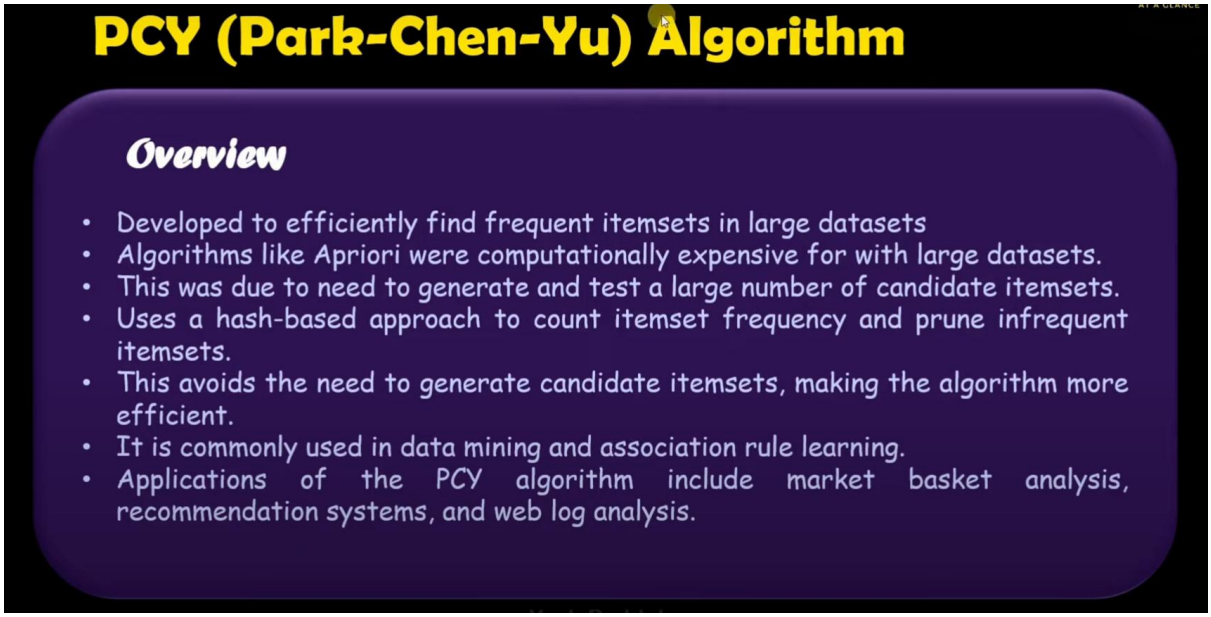
- Rule 4: {1, 2} → {3} {S= 22.22 %, C=50%}
  - Support = 2/9 = 22.22 %
  - Confidence = Support (1, 2, 3)/Support(1, 2) = 2/4 = 50% >= 50%
  - Valid Rule

- Frequent 3-Item Set = I => {1, 2, 3}
- Non-Empty subset are
  - {{1}, {2}, {3}, {1, 2}, {1, 3}, {2, 3}}

- Rule 5: {1, 3} → {2} {S= 22.22 %, C=50%}
  - Support = 2/9 = 22.22 %
  - Confidence = Support (1, 2, 3)/Support(1, 3) = 2/4 = 50% >= 50%
  - Valid Rule

- Rule 6: {2, 3} → {1} {S= 22.22 %, C=66.67%}
  - Support = 2/9 = 22.22 %
  - Confidence = Support (1, 2, 3)/Support(2, 3) = 2/3 = 66.67% >= 50%
  - Valid Rule

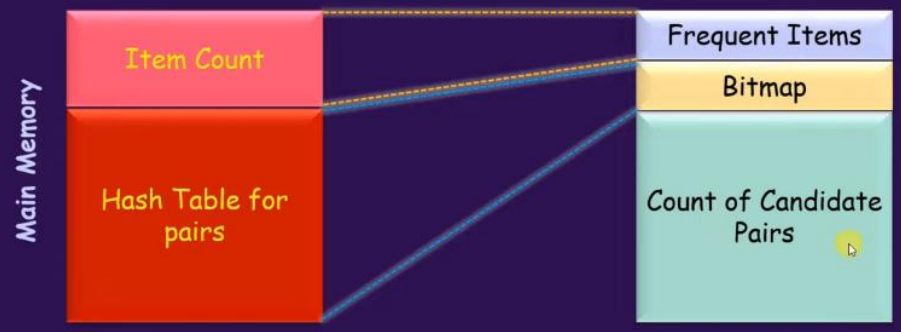
PCY ALGORITHM



PCY (Park-Chen-Yu) Algorithm

AT A GLANCE

Main Memory: Picture of PCY



Yash Paddalwar

PCY (Park-Chen-Yu) Algorithm

Algorithm

Pass 1:

```
FOR (each basket):
    FOR (each item in the basket):
        add 1 to item's count
    FOR (each pair of items):
        hash the pair to a bucket
        add 1 to the count for that bucket
```

Pass 2:

Count all pairs {i, j} that meet the conditions for being a candidate pair:

1. Both i and j are frequent items
2. The pair {i, j} hashes to a bucket whose bit in the bit vector is 1

PCY (Park-Chen-Yu) Algorithm

Example

Find frequent item pairs using PCY algorithm considering threshold as 2

Transactions	Items
T1	1, 2, 3
T2	4, 5
T3	1, 4, 5
T4	1, 2, 4
T5	3, 4, 5
T6	2, 4, 5

PCY (Park-Chen-Yu) Algorithm

Example

Step 1: Find length of each item and eliminate items whose count is less than 1

Transactions	Items
T1	1, 2, 3
T2	4, 5
T3	1, 4, 5
T4	1, 2, 4
T5	3, 4, 5
T6	2, 4, 5

1	3
2	3
3	2
4	5
5	4

Eliminate those items whose count is less than 1, but in our case no item count is less than 1.

Candidate Item-set:
{1, 2, 3, 4, 5}

PCY (Park-Chen-Yu) Algorithm

Example

Step 2: Form key-value pair where key is the item-pair and value is their occurrence

Transactions	Items	Pairs with their counts
T1	1, 2, 3	{(1,2) : 2}, {(2,3) : 1}, {(1,3) : 1}
T2	4, 5	{(4,5) : 4}
T3	1, 4, 5	{(1,4) : 2}, {(1,5) : 1}
T4	1, 2, 4	{(2,4) : 2}
T5	3, 4, 5	{(3,4) : 1}, {(3,5) : 1}
T6	2, 4, 5	{(2,5) : 1}

PCY (Park-Chen-Yu) Algorithm

Example

Step 3: Eliminate pairs whose count of occurrence does not satisfy threshold value

Transactions	Pairs
T1	{(1,2) : 2}, {(2,3) : 1}, {(1,3) : 1}
T2	{(4,5) : 4}
T3	{(1,4) : 2}, {(1,5) : 1}
T4	{(2,4) : 2}
T5	{(3,4) : 1}, {(3,5) : 1}
T6	{(2,5) : 1}

Since, threshold is 2, hence eliminate those pairs whose count is less than 2

Final Pairs:
{(1,2), (4,5), (1,4), (2,4)}

PCY (Park-Chen-Yu) Algorithm

Example

Step 4: Find bucket numbers

Final Pairs: {(1,2), (4,5), (1,4), (2,4)}

Now, Apply hash function to find bucket no. :

For pair (i,j), hash function will be (i * j) mod 10

Pairs	Bucket No.
(1,2)	(1 * 2) mod 10 = 2
(4,5)	(4 * 5) mod 10 = 0
(1,4)	(1 * 4) mod 10 = 4
(2,4)	(2 * 4) mod 10 = 8

PCY (Park-Chen-Yu) Algorithm

Example

Step 5: Create Candidate set table

Bit Vector	Bucket No	Count	Pairs	Candidate Set
1	2	2	(1,2)	(1,2)
1	0	4	(4,5)	(4,5)
1	4	2	(1,4)	(1,4)
1	8	2	(2,4)	(2,4)

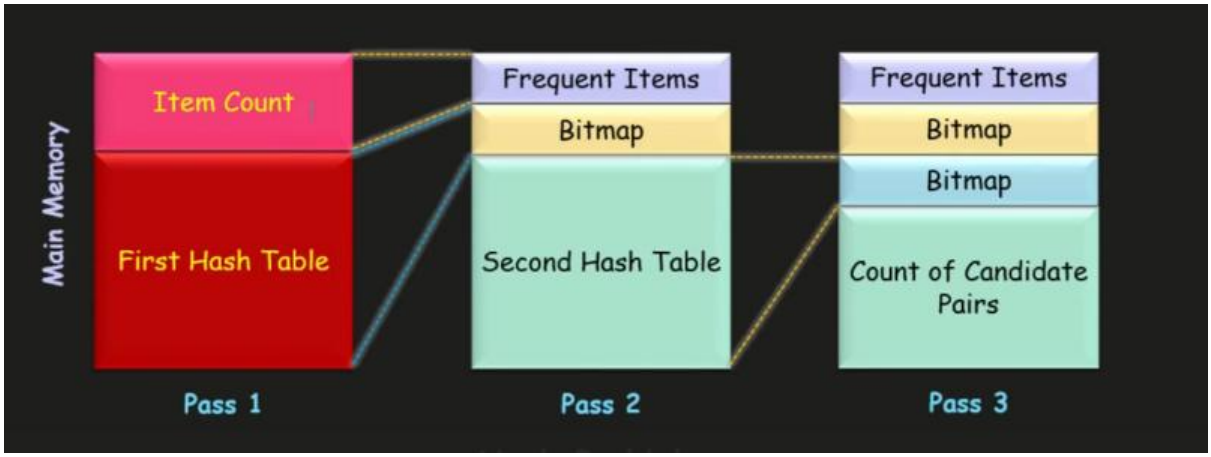
Therefore, Candidate pair set: {(1,2), (4,5), (1,4), (2,4)}

Multistage PCY (Park-Chen-Yu)

Multistage PCY - Refinement of PCY algorithm

Why use Multistage PCY?

1. More Hash Tables: Multistage uses several successive hash tables to further reduce the number of candidate pairs.
2. Additional Passes: Unlike PCY, which takes two passes to find frequent pairs, Multistage takes more than two passes. This allows it to refine the candidate pairs more effectively.
3. Memory Utilization: In Multistage, bit-vectors eventually consume all of the main memory. This is a trade-off for the increased accuracy in candidate pair selection.



Algorithm

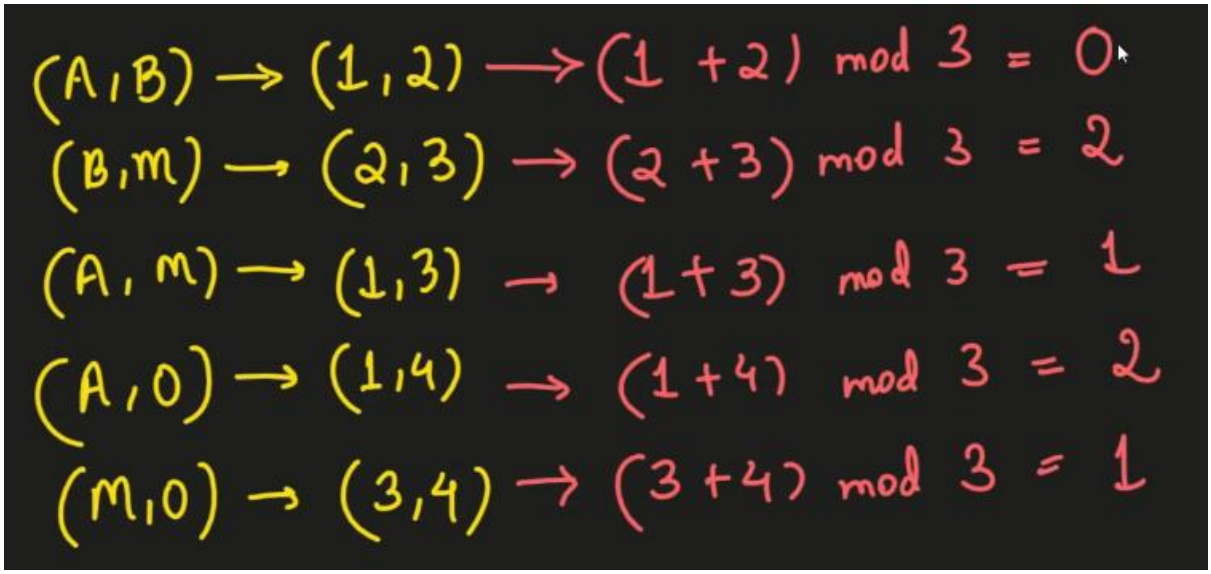
1. Pass 1: Count items and Hash pairs {i, j}
2. Pass 2:
   if (i, j are frequent) and ({i, j} hashes to frequent bucket in B1):
   Hash pairs {i, j} into Hash2
3. Pass 3:
   if (i,j are frequent) and ({i,j} hashes to freq. bucket in B1) and ({i,j} hashes to freq. bucket in B2):
   Count pairs {i,j}

Example

Basket 1: {apple, banana, mango} -> {A, B, M}
Basket 2: {apple, orange} -> {A, O}
Basket 3: {banana, mango} -> {B, M}
Basket 4: {apple, banana} -> {A, B}
Basket 5: {mango, orange} -> {M, O}
Minimum support count to be 2.

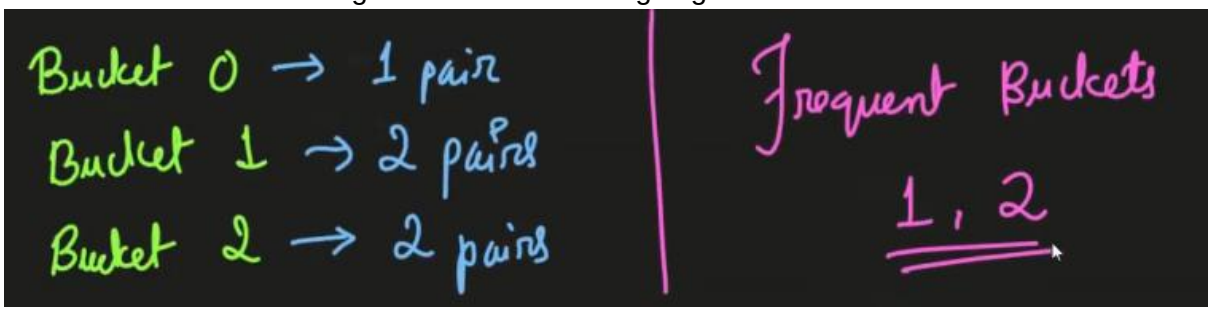
1 ← A → 3
2 ← B → 3
3 ← M → 3
4 ← O → 2

Hash 1 = (i + j) mod 3



The numbering given to A,B,M,O are used here for computation of hash1 and not the frequency.

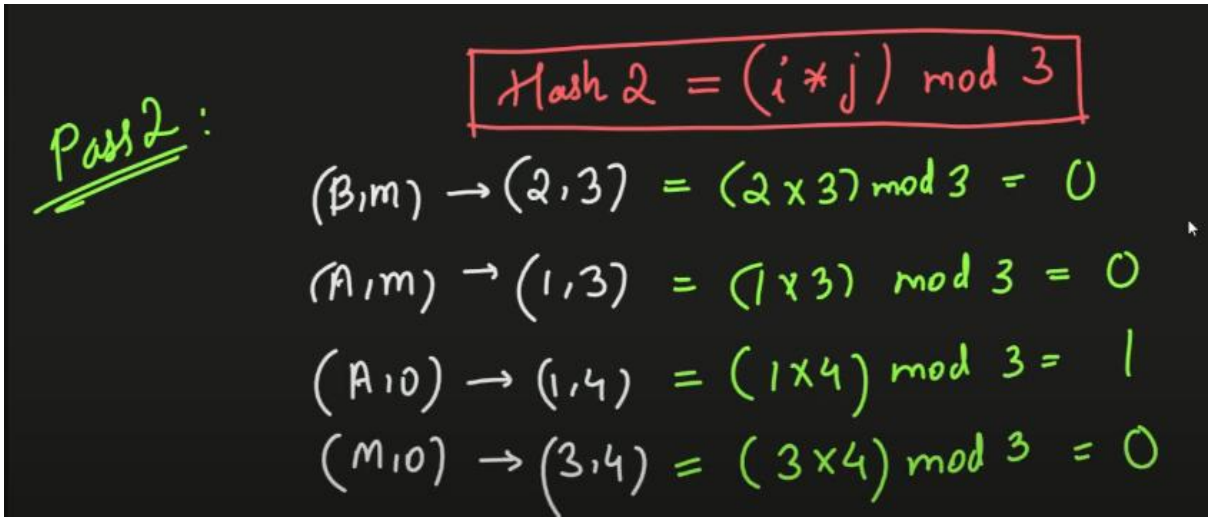
As the values of hash1 range from 0-2 there are going to be a total of 3 buckets.

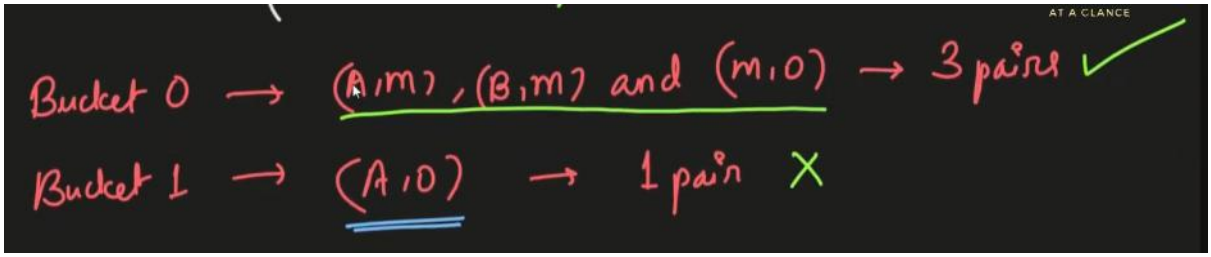


A bucket is said to be frequent if it contains a minimum support number of pairs.

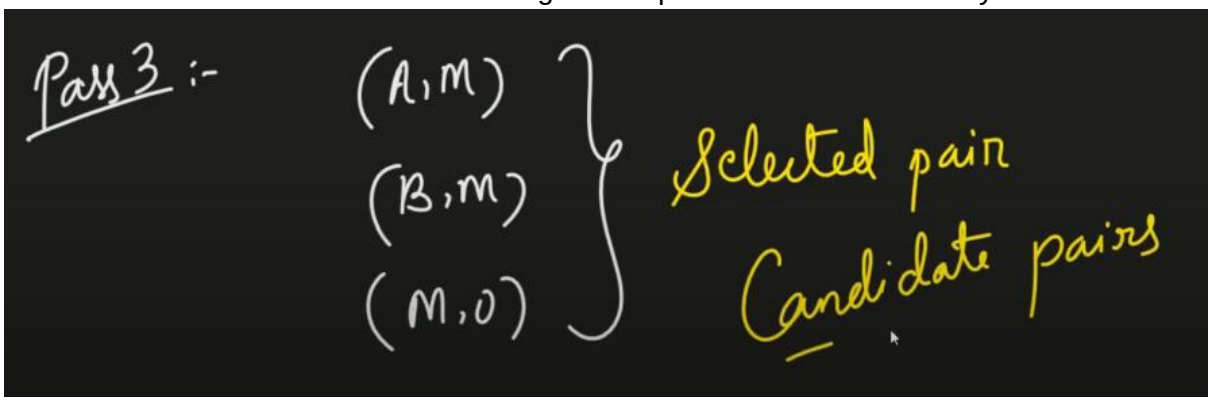
So this was pass 1.

We are going to consider pairs from bucket 1 and 2 for the 2nd pass because those are the frequent buckets.



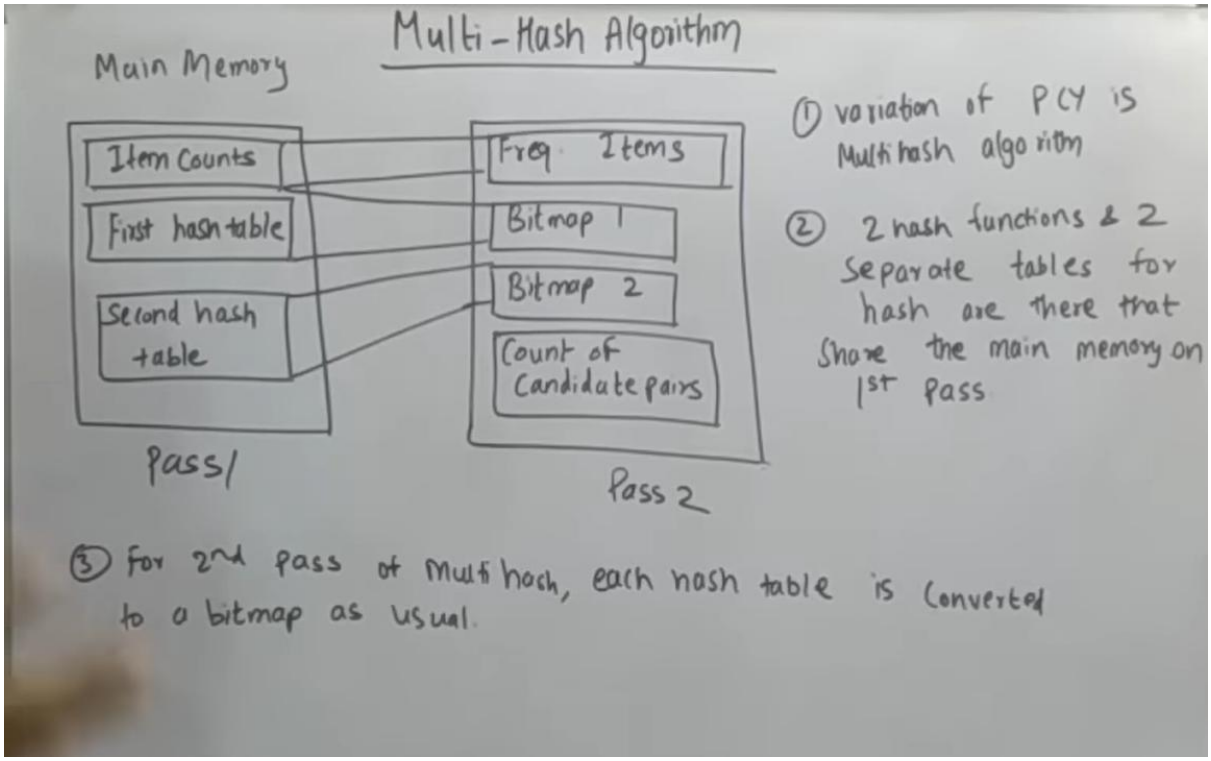


So Pass 2 is done. Check the conditions given for pass 3 above and classify them.



Note: The hash function you take in pass 1 should be independent from the one you take in pass 2 else the result might not be correct or you might get the same pairs in the same bucket.

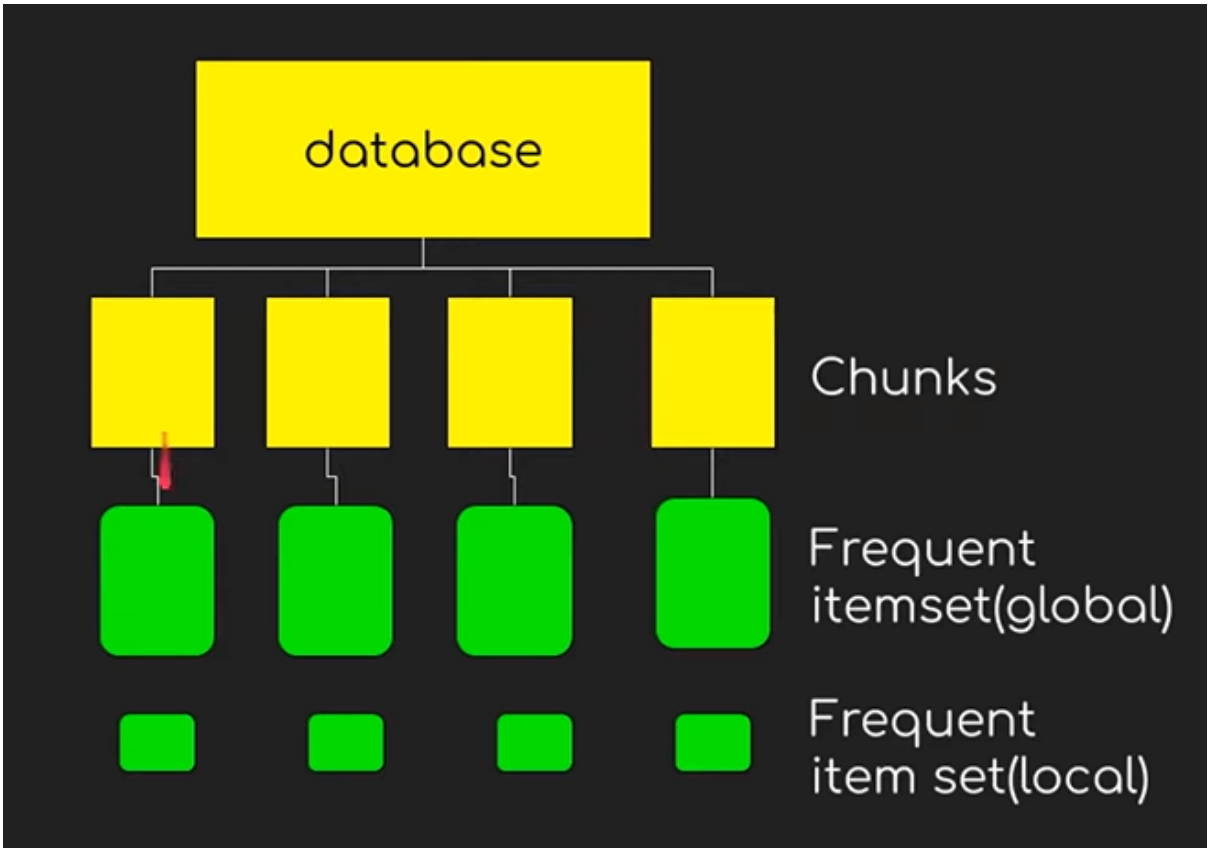
Multihash Algorithm:



SON Algorithm (Savasere, Omiecinski, and Navathe)

Basic knowledge about SON

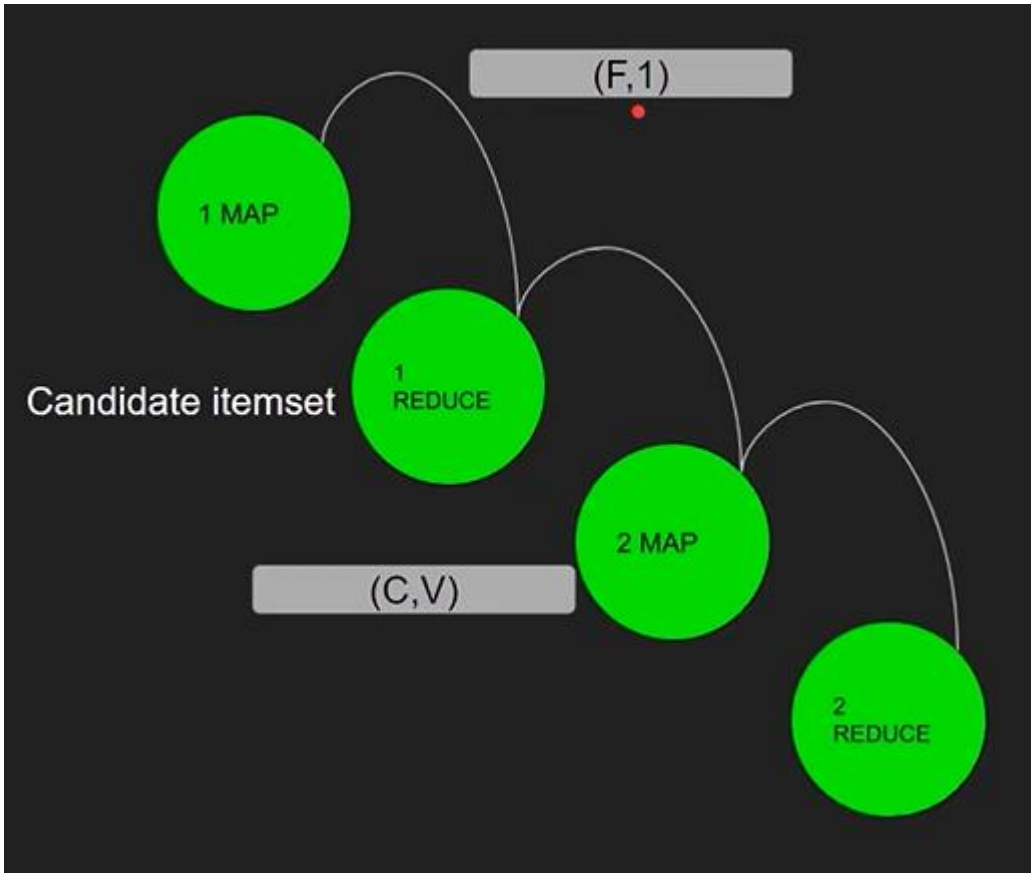
- Also known as “partition algorithm”
- It basically makes partition of database and does testing on each of them later on combines extracted results (this is how it follows MapReduce paradigm)
- It basically does parallel computing which saves time and memory
- Parallel computing and mapreduce makes SON good for finding frequent itemsets in Big data



A	5
B	4
A	2
E	1
….	….

Output of first map reduce

A	1
B	1
C	1
E	1
….	….



A	1
B	1
C	1
E	1
....	....

Output of Second map reduce

A	5
B	4
D	2
E	1
....	....

SON does not generate any false negative

- During the second map-reduce phase . All candidate itemsets are considered
- The result is the total support for each of the itemsets that the Reduce task was assigned to handle. Those itemsets whose sum of values is at least s are frequent in the whole dataset, so the Reduce task outputs these itemsets with their counts. Itemsets that do not have total support at least s are not transmitted to the output of the Reduce task.
- Basically "*if an itemset I is infrequent in every chunk, I is infrequent in whole dataset* "

SON Algorithm and MapReduce

Overview

- SON stands for Savasere, Omiecinski, and Navathe.
- Improvement of PCY algorithm and it's versions.
- Used to find frequent itemsets in large datasets.
- Uses two MapReduce passes.
- Can be parallelized for faster processing.
- Used in data mining and market basket analysis.

Principle

An itemset is said to be frequent only if it is frequent in subsets

Ideology

Pass 1: Finds candidate itemsets

Pass 2: Finds correct frequent itemsets

Algorithm

Pass 1: First Map task

1. Divide data into chunks
2. For each chunk the support will be:
   support = s / number of chunks
3. Using random algorithm find frequent itemset in that chunk and output them as (F, 1) where F is frequent itemset and 1 is irrelevant number

Pass 1: First Reduce task

1. The value is ignored from (F, 1) and key part F is assigned to each reduce task
2. Produces keys that appear one or more times
3. These produced keys are candidate itemsets

Pass 2: Second Map task

1. Each Map task takes the output from first reduce task and a portion of the input data file
2. Every map task will have all the candidate itemsets
3. It counts the occurrence of each candidate itemset among the baskets in the portion of the dataset
4. Output of the map task is (C,v)
   Where, C is candidate itemset
   v is the support of that itemset among the baskets that were input to this map task

Pass 2: Second Reduce task

1. Each reduce task takes itemsets (keys from map phase)
2. Calculates sum of associated values (support) for each itemset
3. Checks the below condition:

```
if support of itemset >= s:
    emit(itemset)
```

4. Itemset that does not satisfies the support criteria are not transmitted as output

Advantage

False Negative

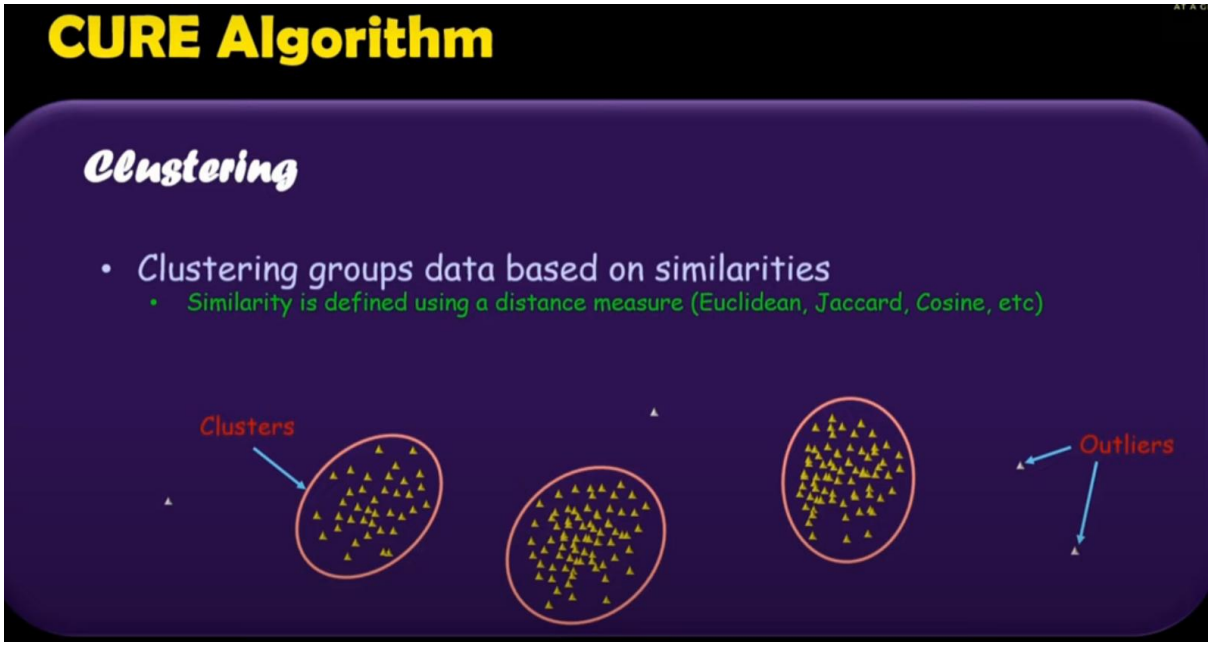
In reality the itemset is frequent

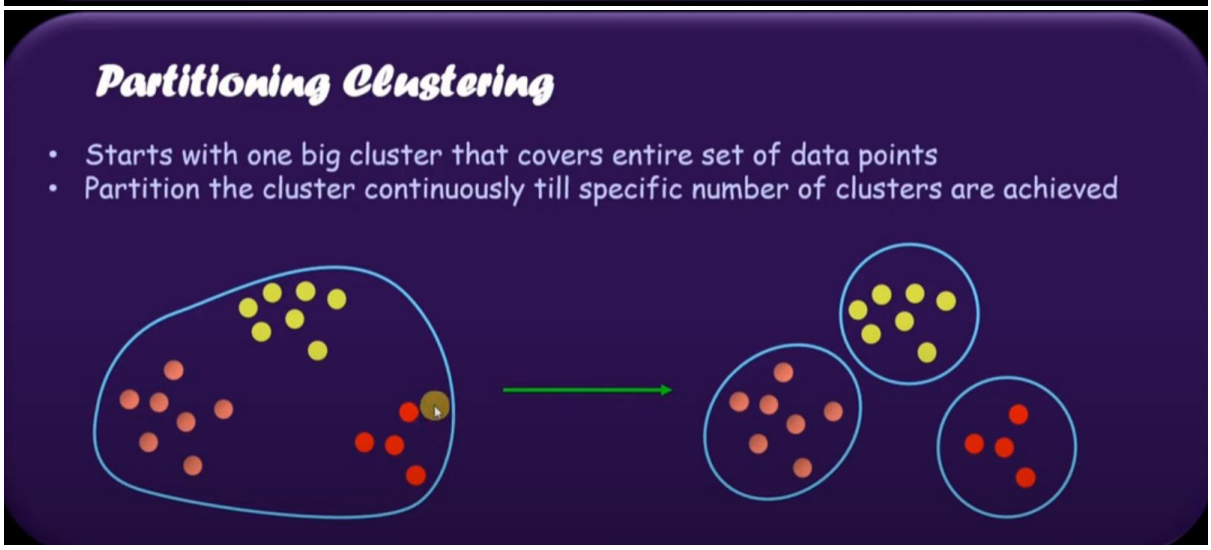
As per the results the itemset is not frequent

False Negatives will not be entertained

CURE Algorithm

[Cure Algorithm in Hindi | Big data analytics Tutorials - YouTube]





Hierarchical Clustering

- Starts with single data point level cluster
- Merge clusters step by step until desired number of clusters are achieved

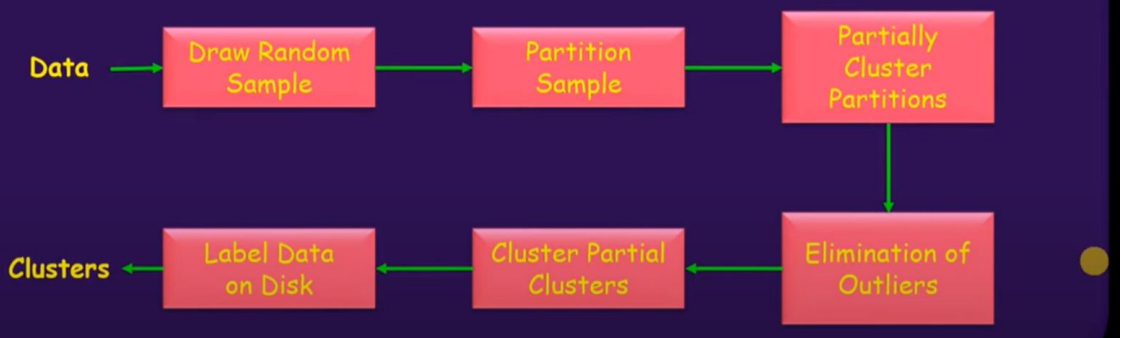
CURE Algorithm

Overview

- CURE stands for Clustering Using REpresentatives
- Specially designed to work efficiently on larger datasets
- Uses a collection of representative points to represent clusters
- Adopts a middle ground between centroid based and all-point extremes
- Capable of detecting clusters of any shape
- Detect outliers and remove it

CURE Algorithm

Architecture



Algorithm

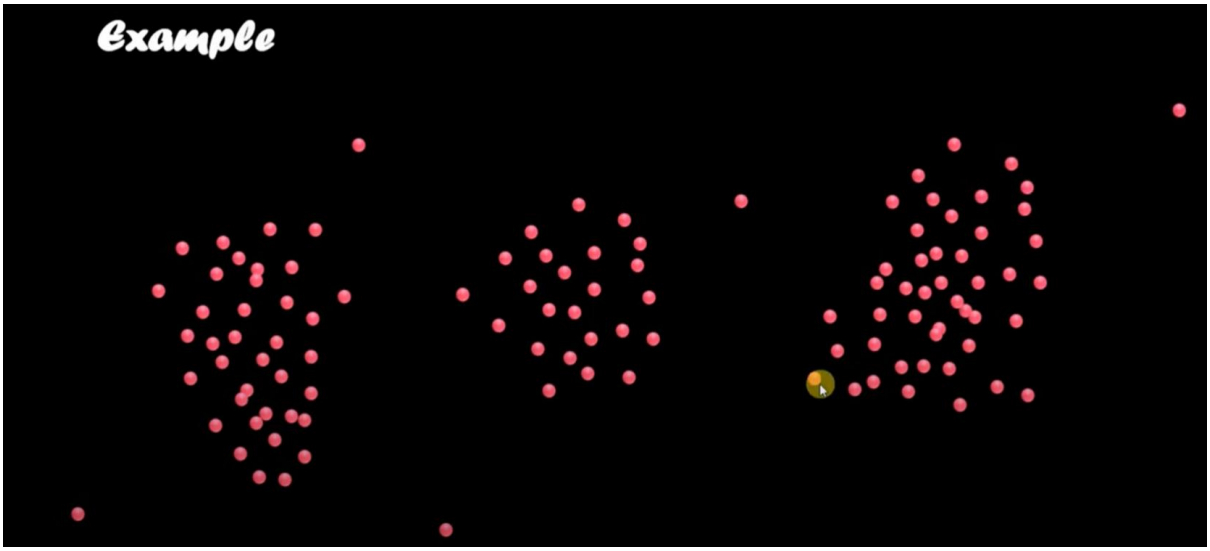
Pass 1:

1. Pick random samples that fit in main memory and cluster them
2. Choose c scattered points in each cluster. (Let's take c = 4)
3. These scattered points are shrunk towards centroid in a fraction of α where 0 < α < 1
4. Use dmin cluster merging approach considering these scattered points as representatives of clusters

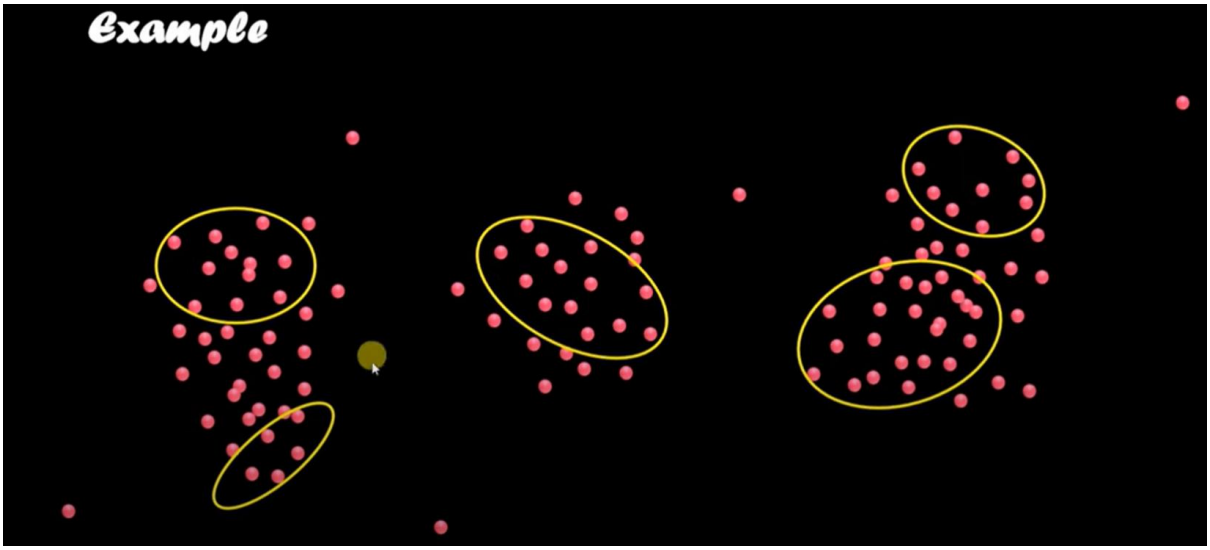
Pass 2:

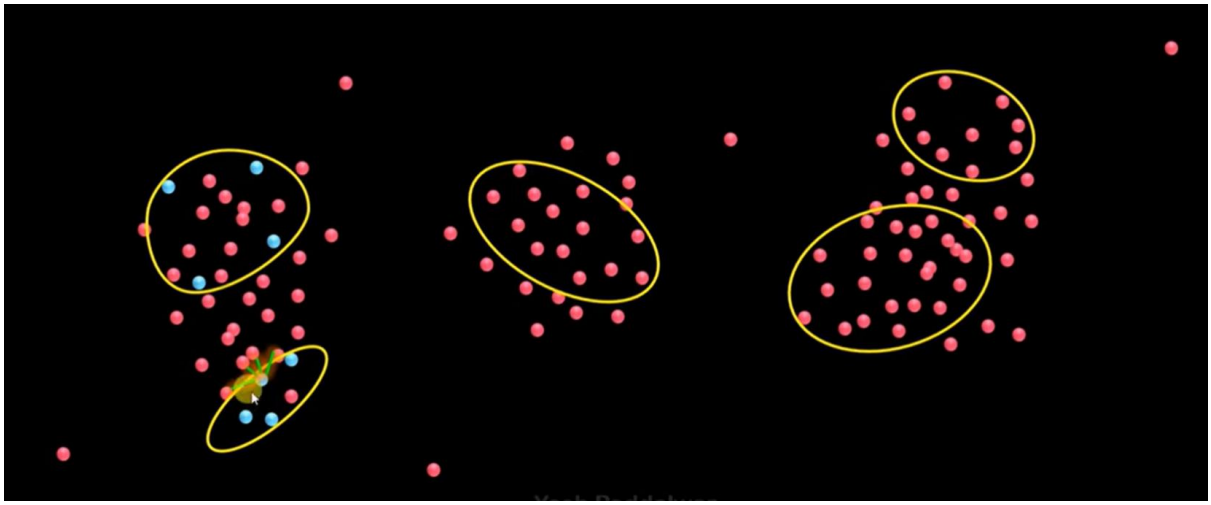
1. After every merge, new points merged are considered as representative for new cluster
2. Finally, cluster merging will stop when target k (number of clusters) is achieved

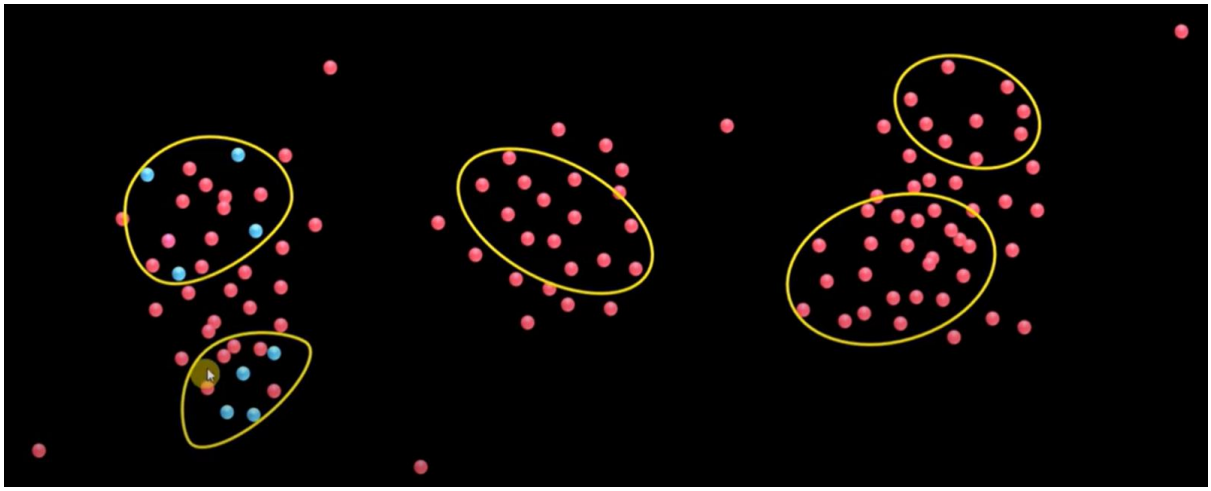
Example

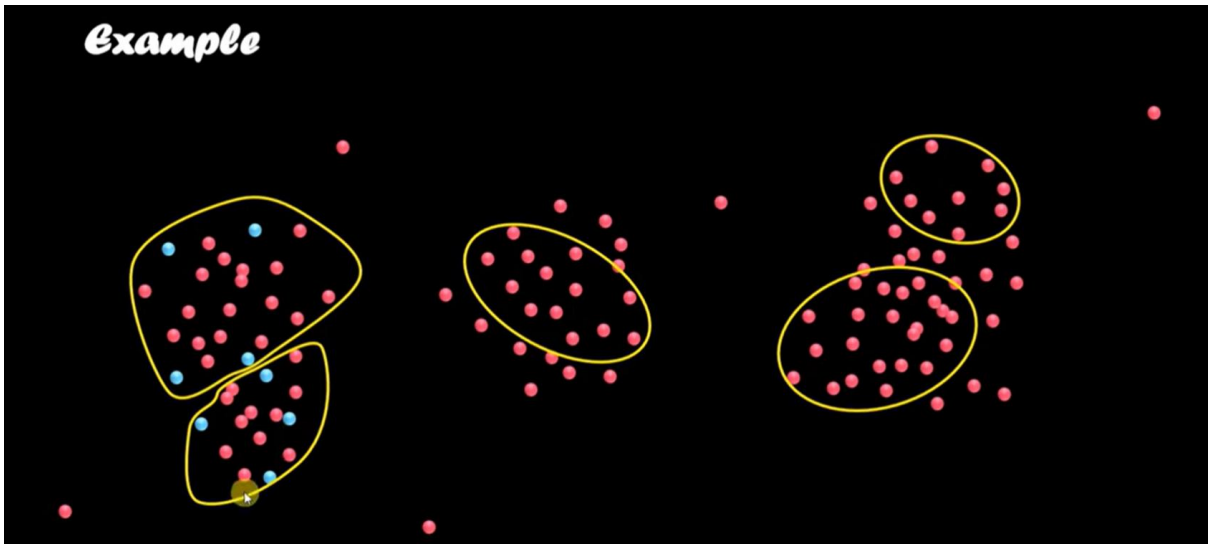


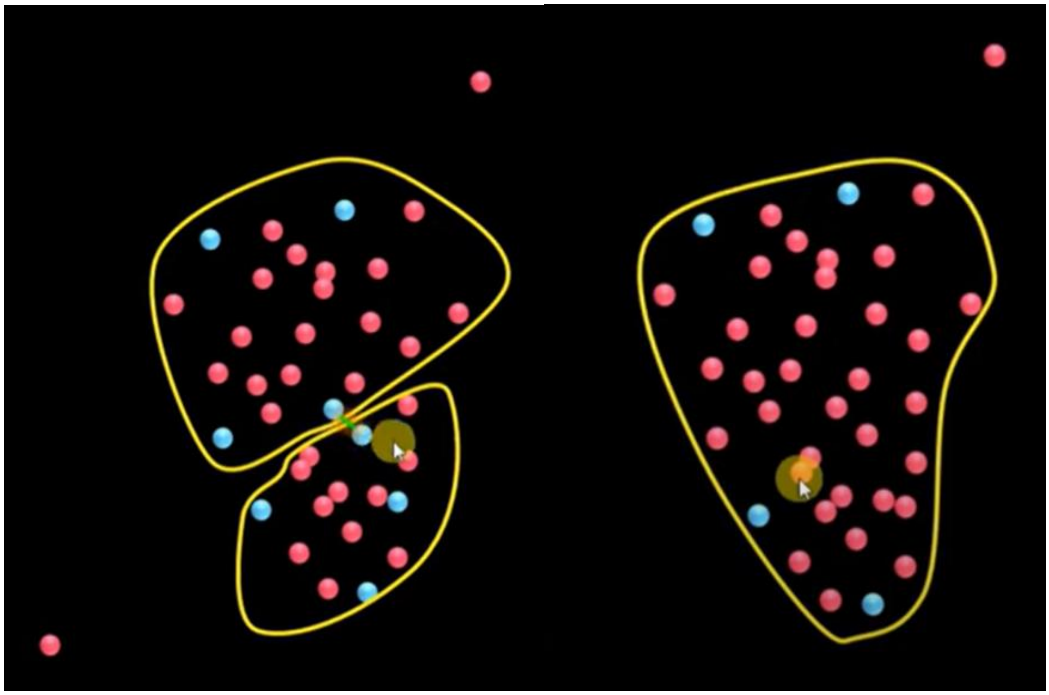
Example

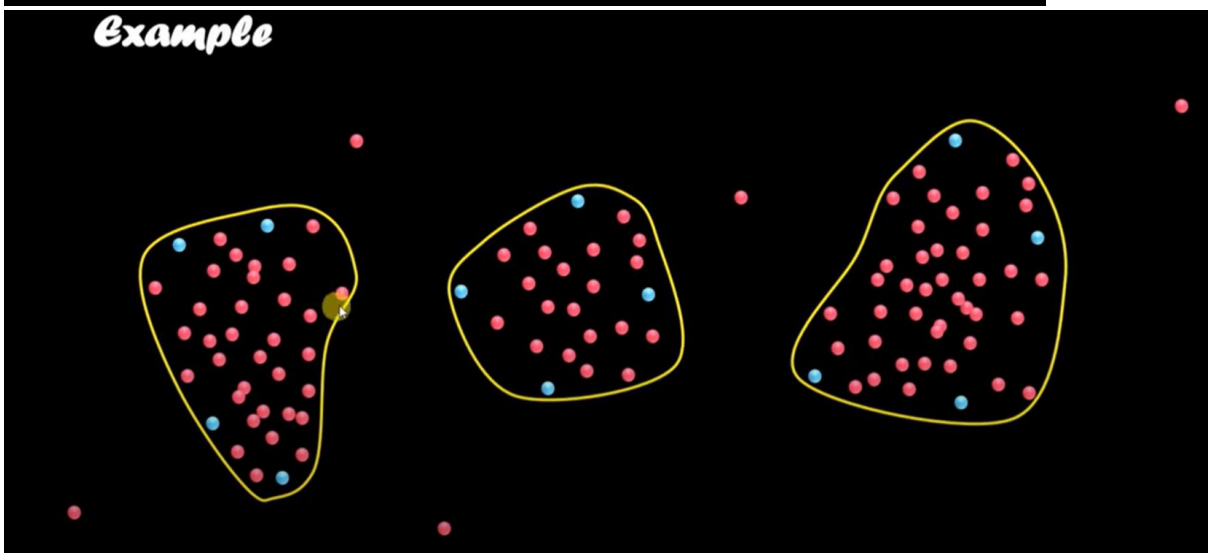


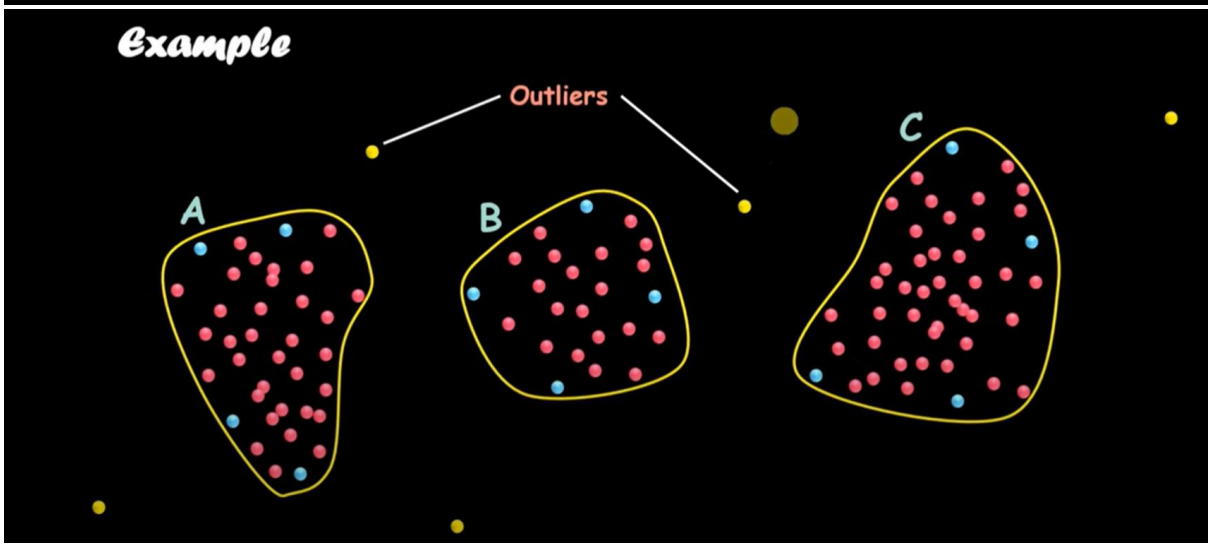


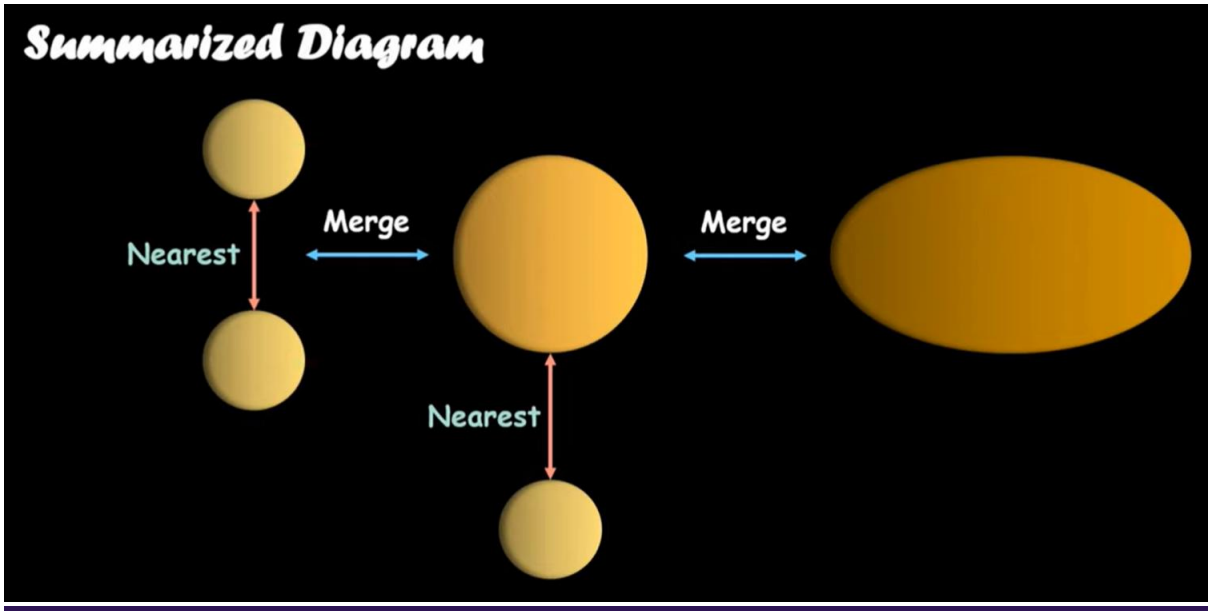



Example




Example

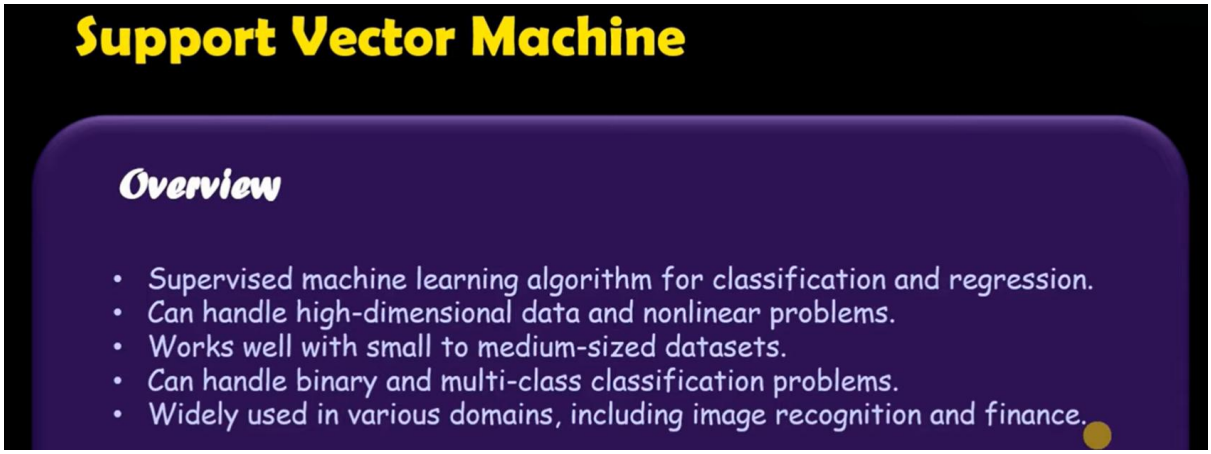

Example
Outliers
A
B
C

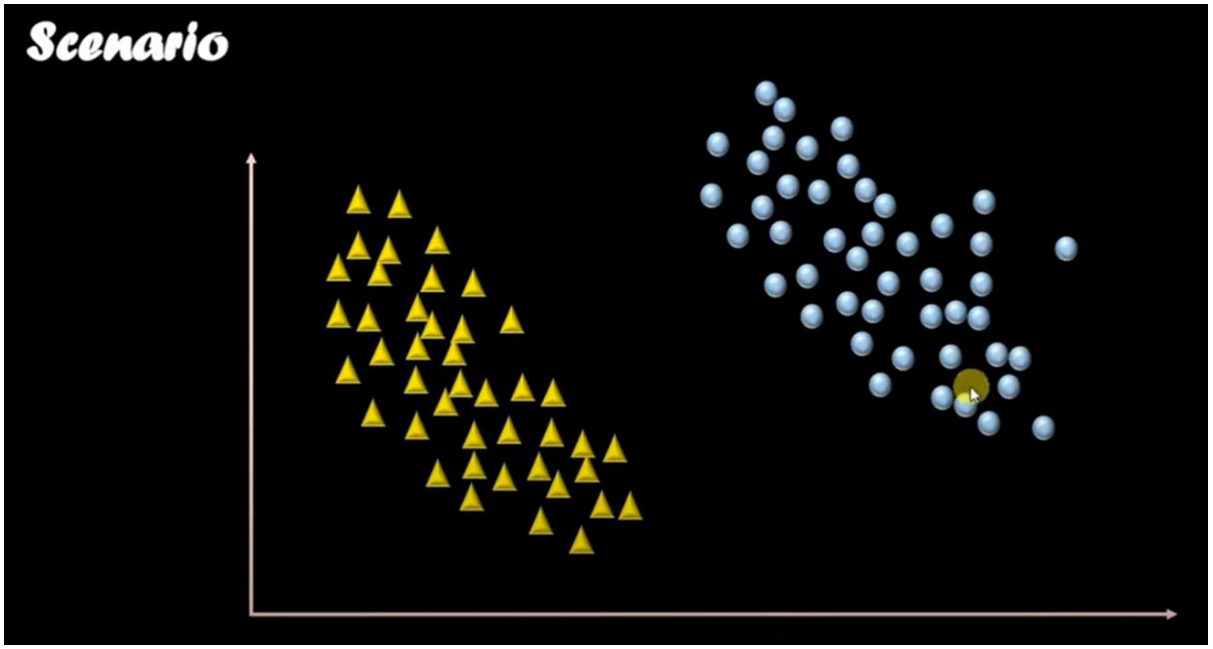


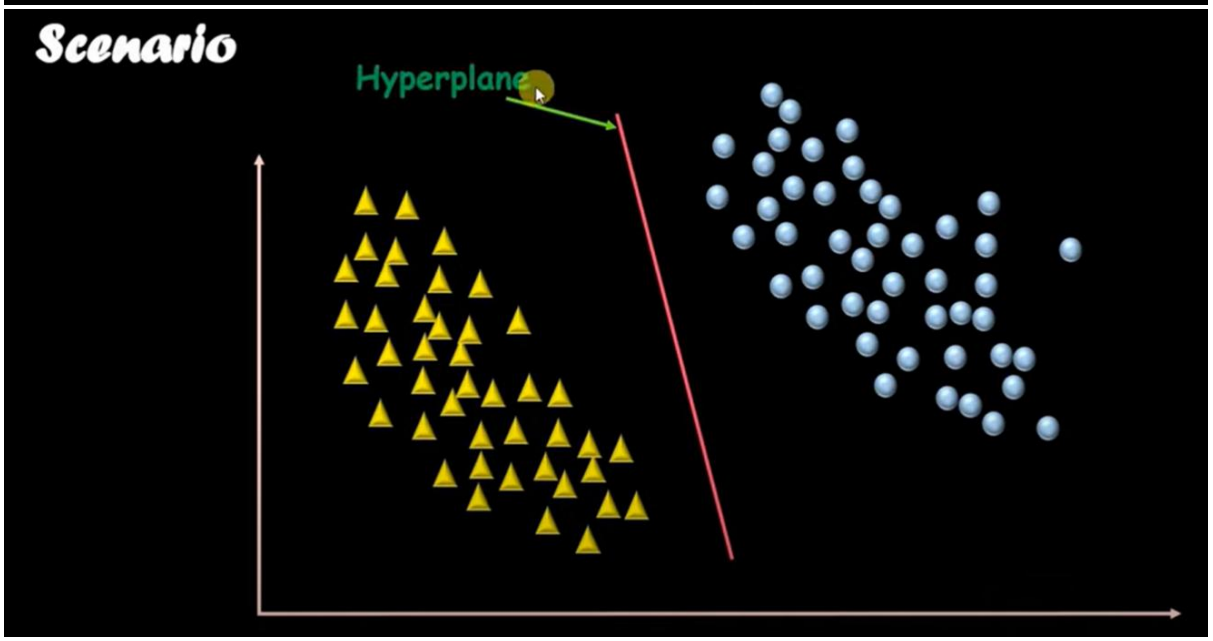
Advantages

- Accurate results
- Adjusts perfectly to non-spherical cluster shapes
- Efficient for large datasets
- Less sensitive to outliers
- Time complexity: $O(n^2 \log n)$ [$O(n^2)$ for small dimensionality of data]
- Space complexity: $O(n)$

Support Vector Machine | High Margin




Scenario


Scenario
Hyperplane

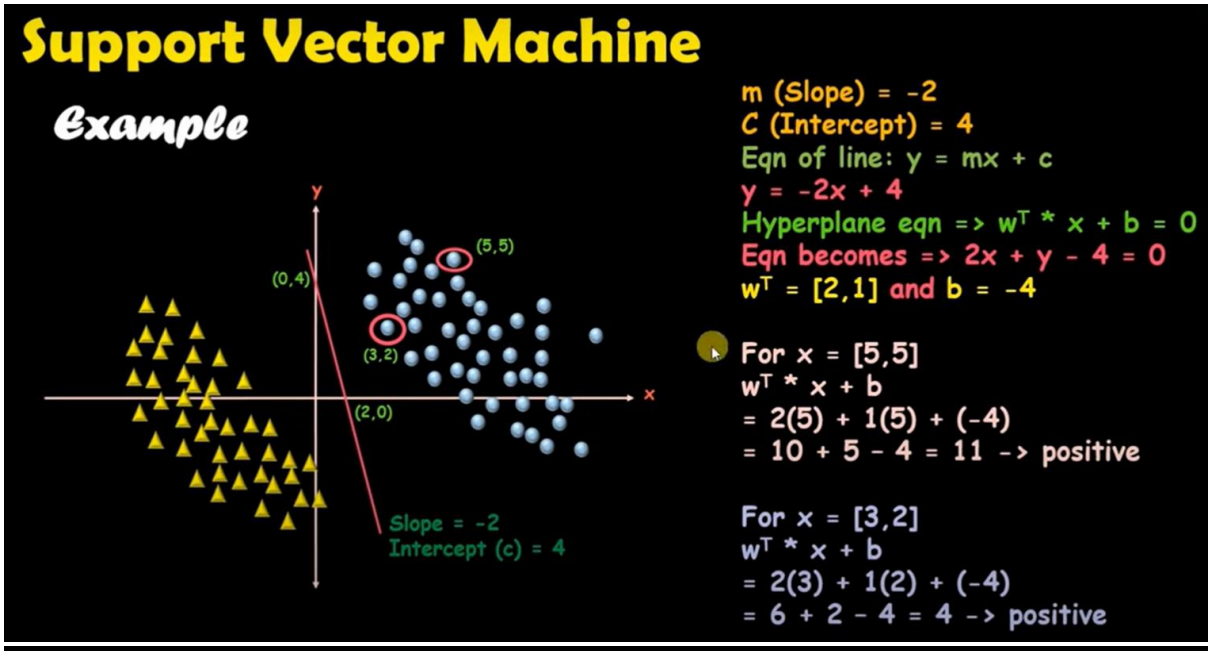
Equation of hyperplane:

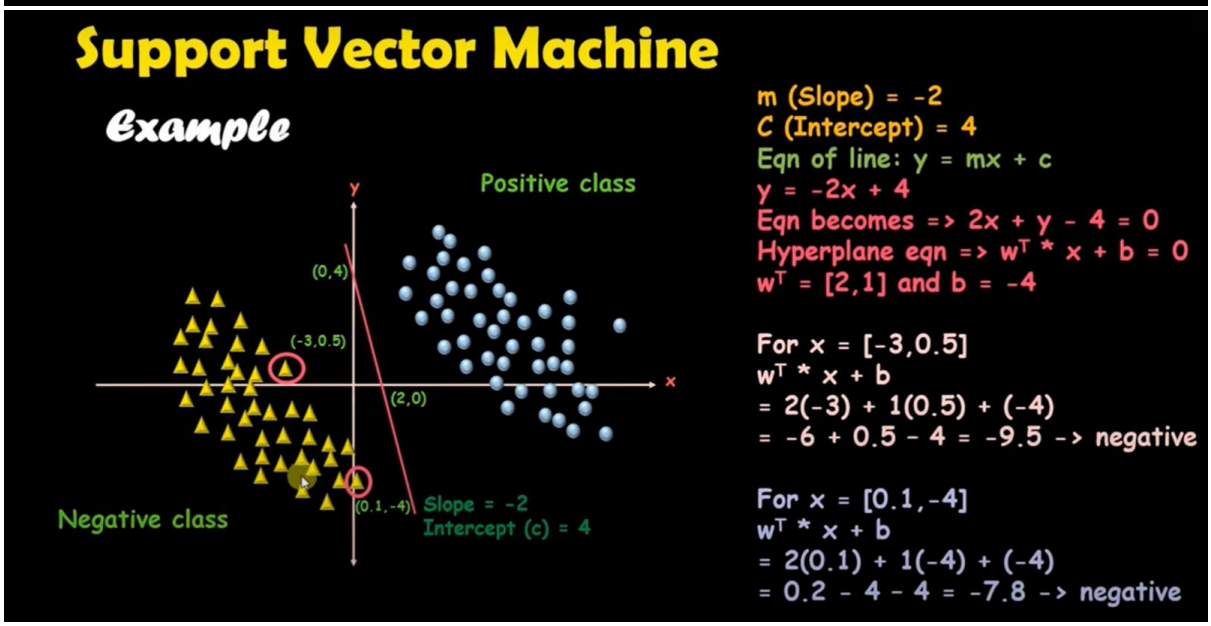
$$w^T * x + b = 0$$

w vector is perpendicular to the hyperplane and defines the orientation and direction of the decision boundary
They are the coefficients

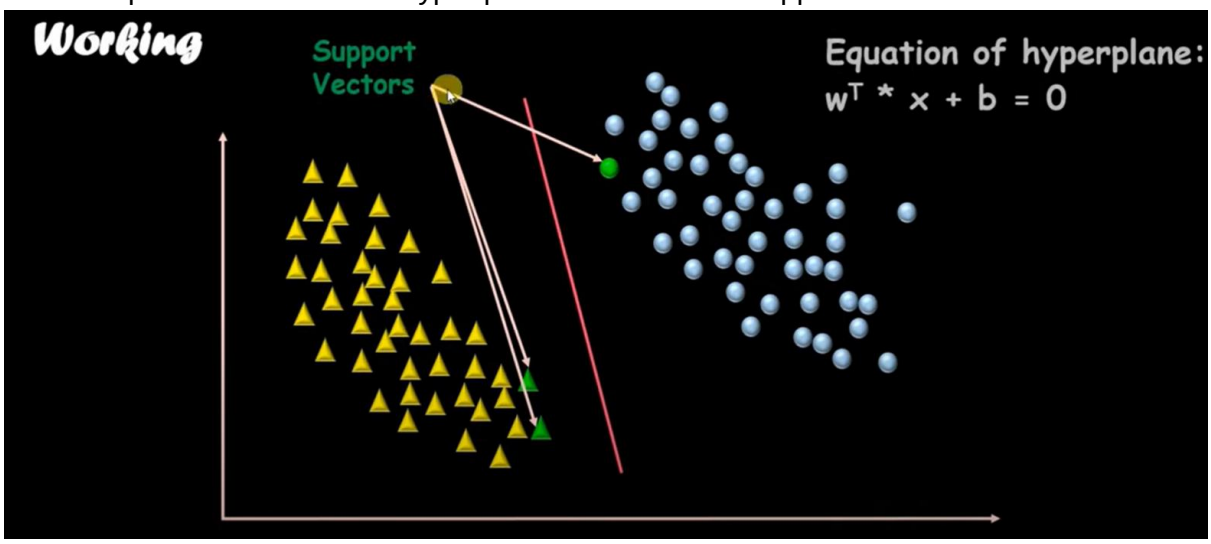
"x" in SVM refers to an input feature vector or data point containing the values of individual features that describe a specific data instance.

"b" represents the bias term or intercept, which helps in positioning the decision boundary accurately.





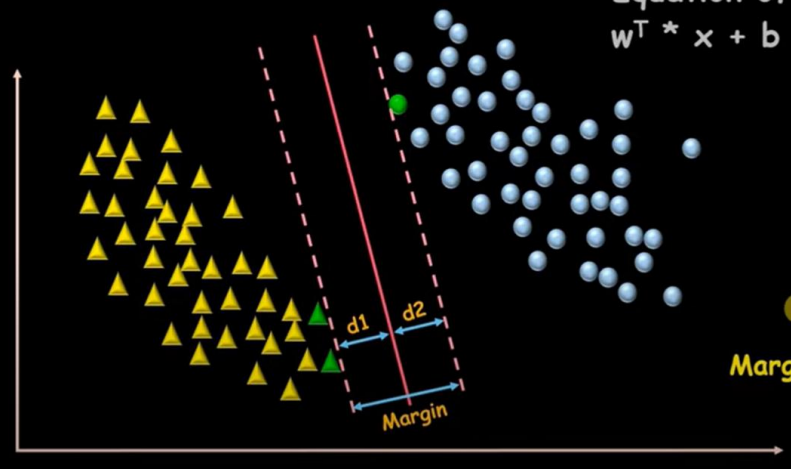
The data points nearest to the hyper plane are said to be support vectors



Support Vector Machine

Working

Equation of hyperplane:
$w^T * x + b = 0$

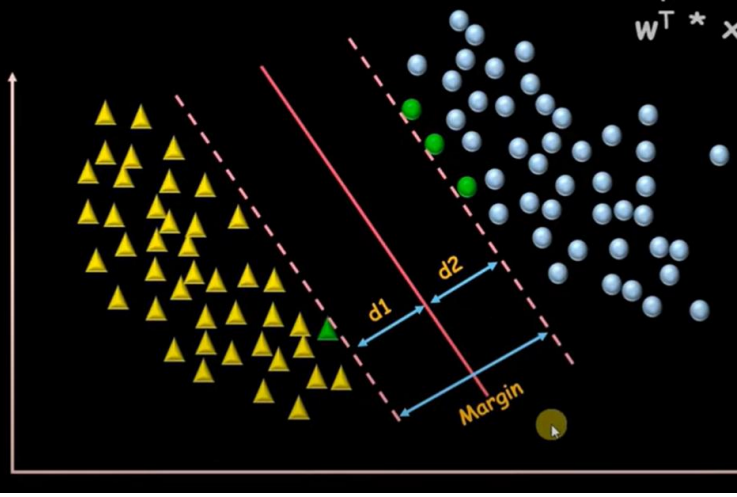


Margin = d1 + d2

Support Vector Machine

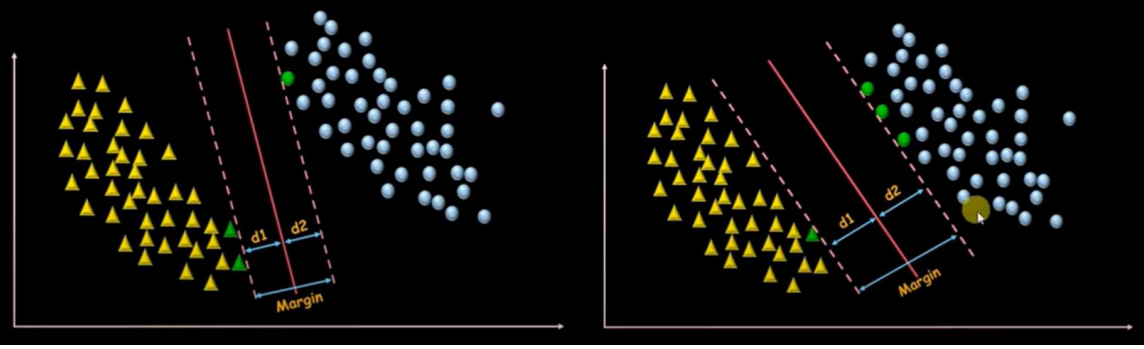
Working

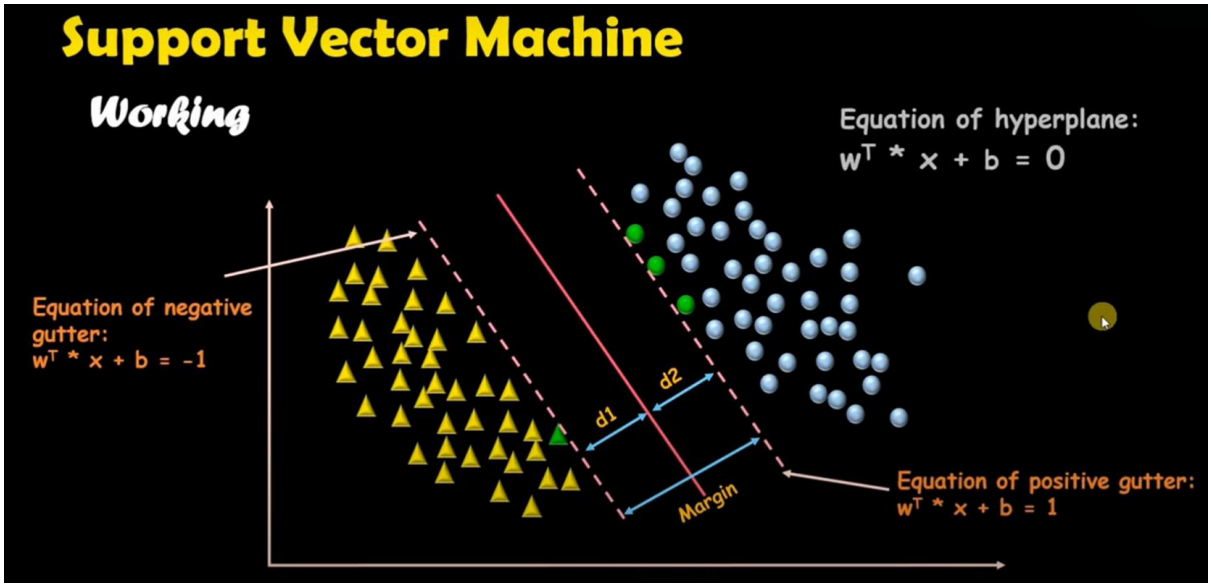
Equation of hyperplane:
$w^T * x + b = 0$



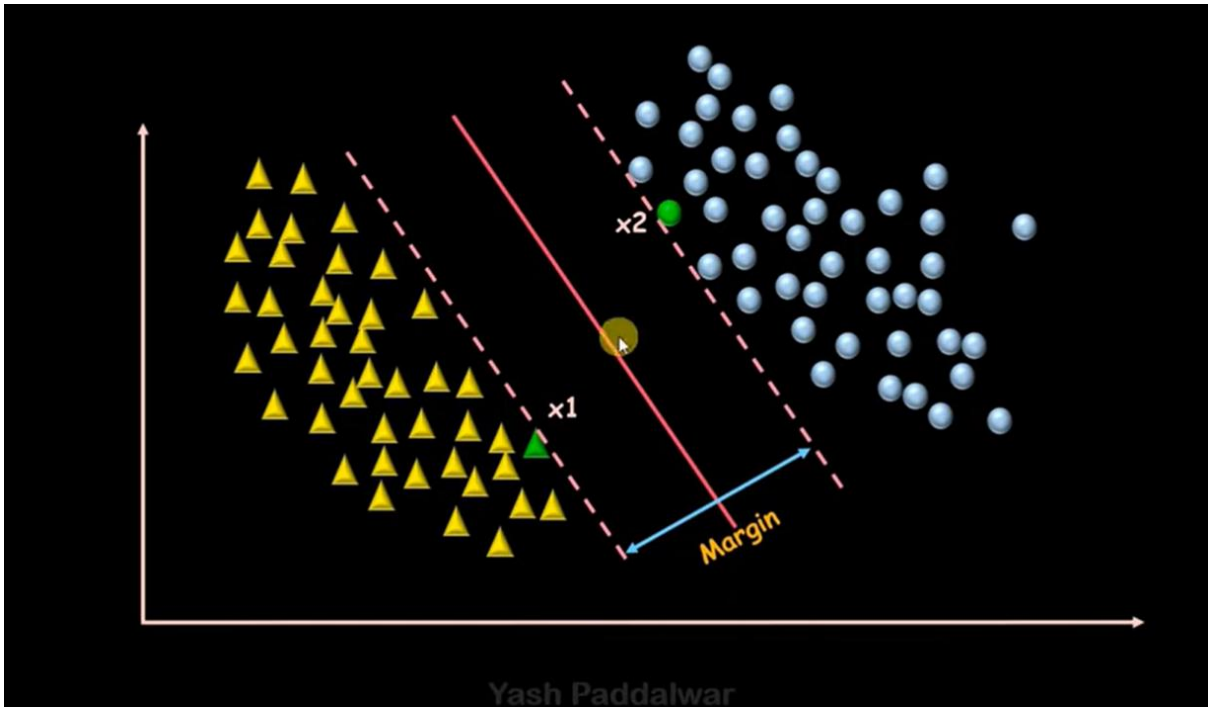
Support Vector Machine

Margin1 << Margin2

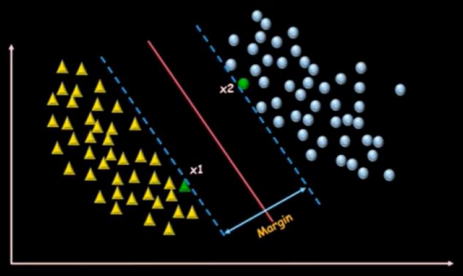




To have less complexity in calculations we have taken 1 & -1



Support Vector Machine



Rule: $y_i = \begin{cases} w^T * x + b >= 1 \text{ , if positive} \\ w^T * x + b <= -1 \text{ , if negative} \end{cases}$

yi for positive class will be +1
yi for negative class will be -1

Multiplying yi with both the cases
Condition boils down to:

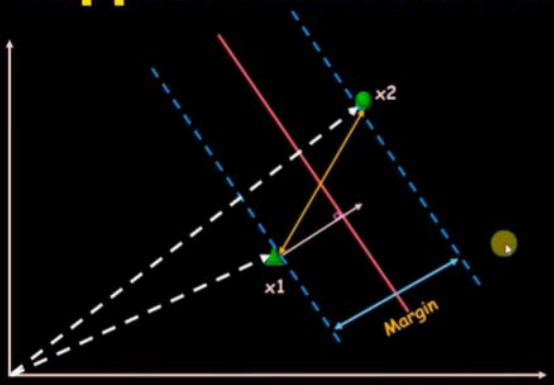
For positive: $y_i (w^T * x_i + b) >= 1$
$= 1(w^T * x_i + b) >= 1$
$= w^T * x_i + b >= 1$

For negative: $y_i (w^T * x_i + b) <= -1$
$= -1(w^T * x_i + b) <= -1$
$= w^T * x_i + b >= 1$

Hence, single condition for both:
$y_i (w^T * x_i + b) >= 1$

Equation of positive gutter: $w^T * x + b = 1$
Equation of negative gutter: $w^T * x + b = -1$

Support Vector Machine



Condition -> $y_i (w^T * x_i + b) >= 1$
Goal is to find Margin distance

Consider vector $\overline{x1}$ and $\overline{x2}$
Distance between x1 and x2 is: $\overline{x2} - \overline{x1}$
We already have the perpendicular vector w^T
To make w^T as unit vector divide it by its norm

$$\text{Margin} = (\overline{x2} - \overline{x1}) \cdot \frac{w^T}{||w||}$$

$$\text{Margin} = \frac{(\overline{x2} \cdot w^T - \overline{x1} \cdot w^T)}{||w||}$$

Margin = (1 - b) - (-1 - b) / ||w||
Margin = (1 - b + 1 + b) / ||w||

Margin = 2 / ||w||

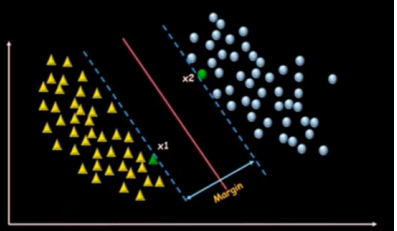
For Support vectors: $y_i (w^T * x_i + b) = 1$
For x2: $1(w^T * x2 + b) = 1$ -> $w^T * x2 = 1 - b$
For x1: $-1(w^T * x1 + b) = 1$ -> $w^T * x1 = -1 - b$

Support Vector Machine

Margin = 2 / ||w||

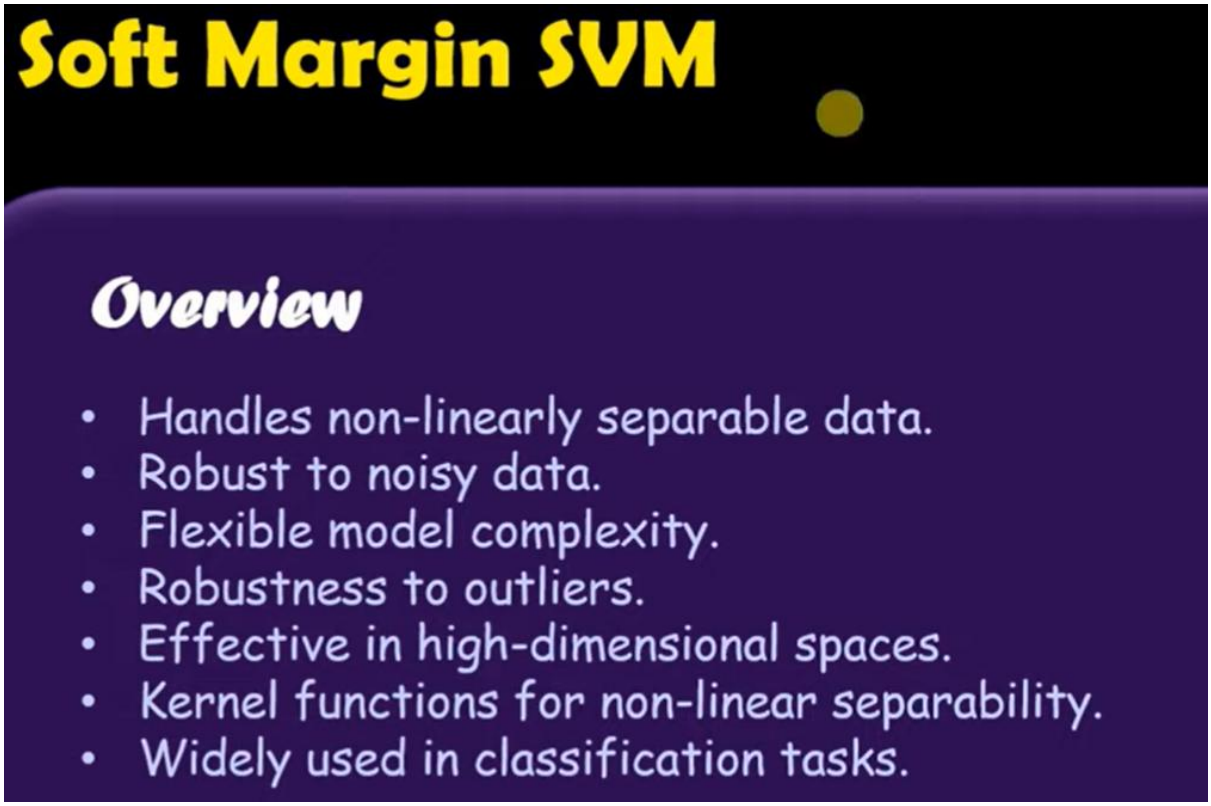
$$\text{argmax}(w^*, b^*) \frac{2}{||w||}$$

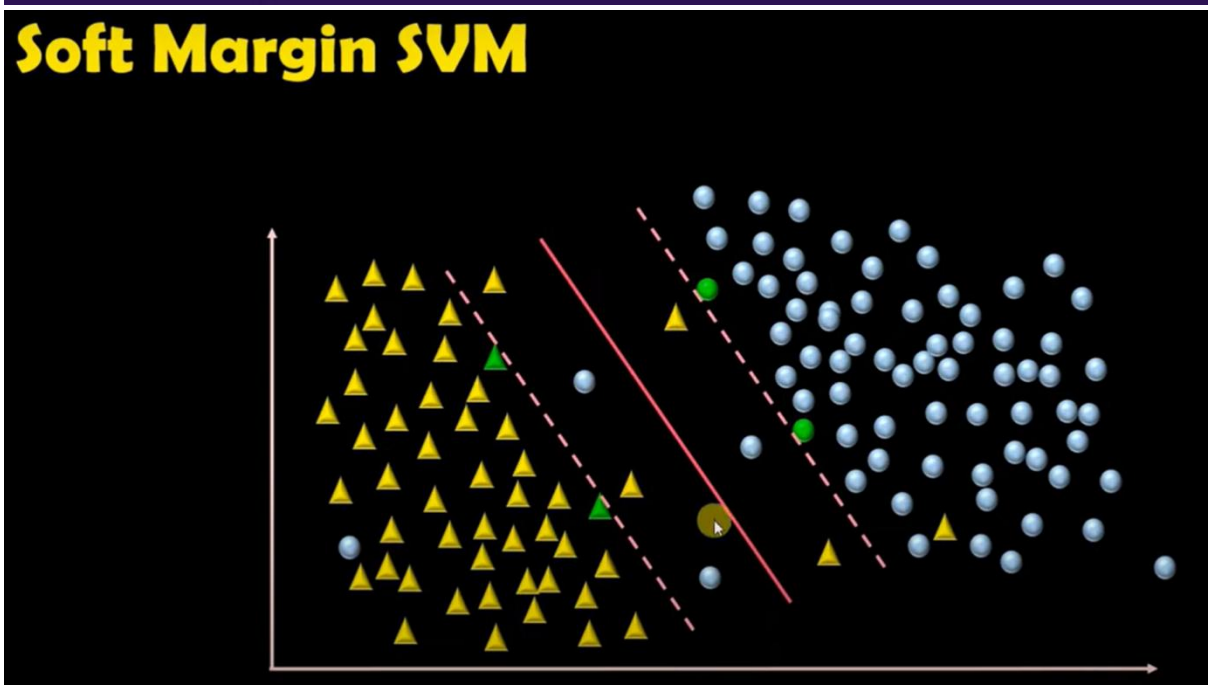
on a condition, $y_i (w^T * x_i + b) >= 1$



This is for hard margin SVM, which means it is applicable for only data where hyperplane can perfectly separate both classes

Support Vector Machine | Soft Margin



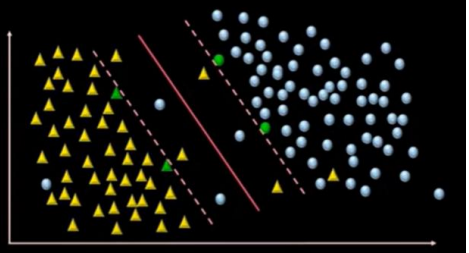


Soft Margin SVM

Margin = 2 / ||w||

$$\text{argmax}(w^*, b^*) \ \frac{2}{||w||}$$

on a condition, $y_i \ (w^T * x_i + b) >= 1$



Soft Margin SVM

$$\text{argmax}(w^*, b^*) \ \frac{2}{||w||}$$

$$\text{argmin}(w^*, b^*) \ \left(\frac{2}{||w||}\right)^{-1}$$

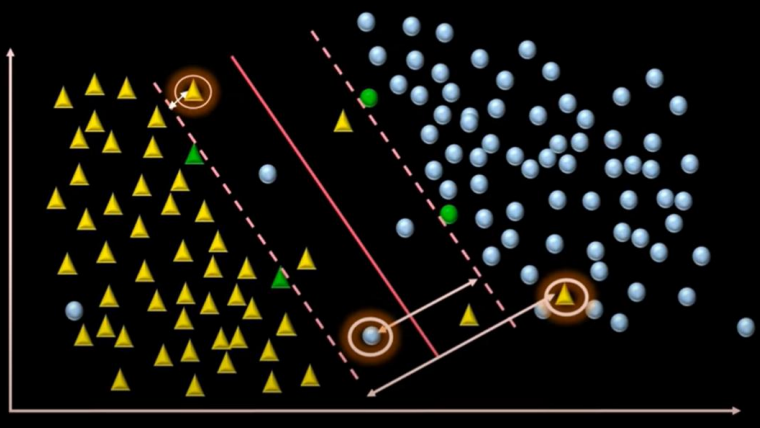
$$\text{argmin}(w^*, b^*) \ \frac{||w||}{2}$$

Soft Margin SVM

ξ (Xi)
Distance of a misclassified point from the gutter of the class in which it should ideally fall

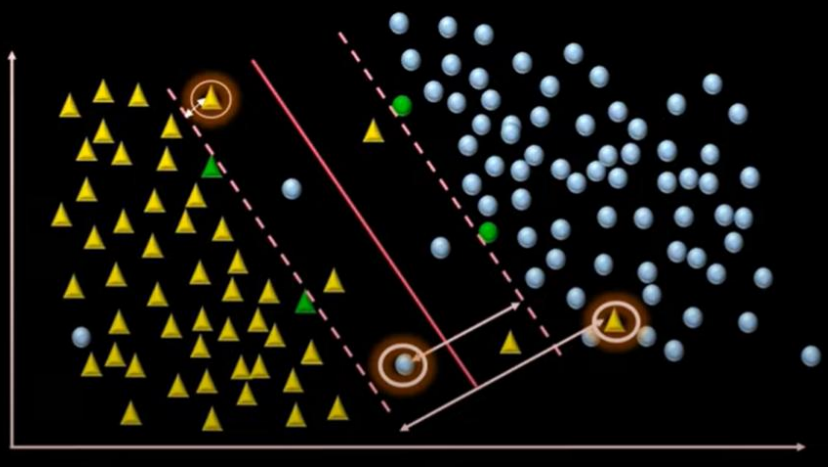
For Correctly classified data points: ξ = 0

Data point that falls in the same class but lies between the margin (gutter) and the hyperplane will have: ξ > 0



Soft Margin SVM

$$\text{argmin}(w^*, b^*) \frac{||w||}{2} + C \sum_{i=1}^{n} \xi_i$$



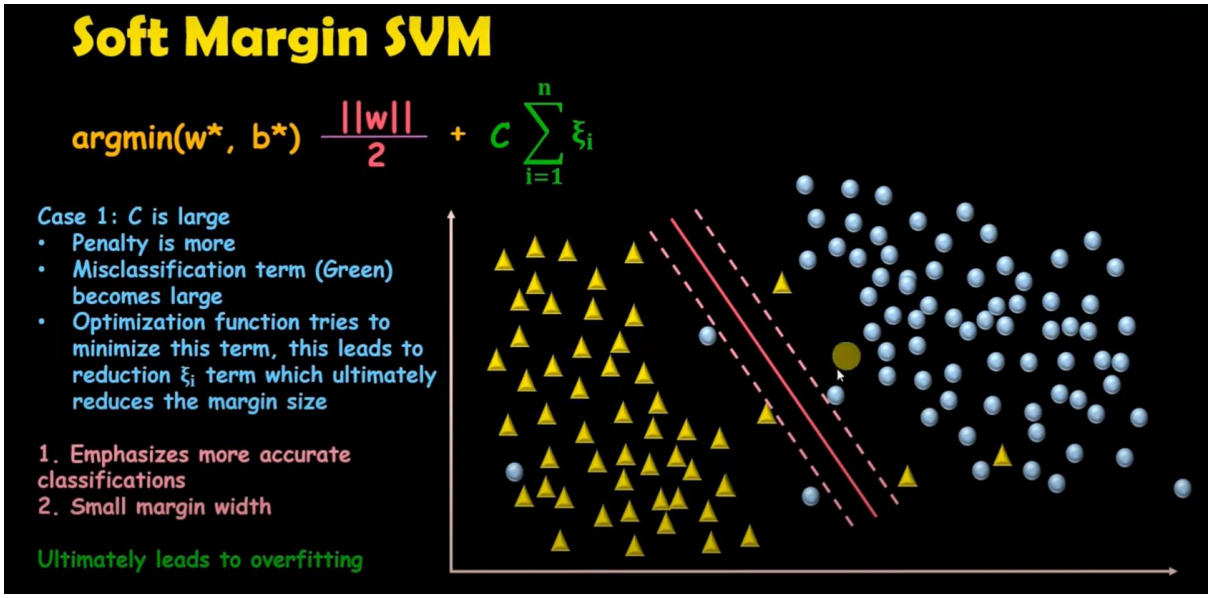
A hyper-parameter that determines the penalty for misclassifications

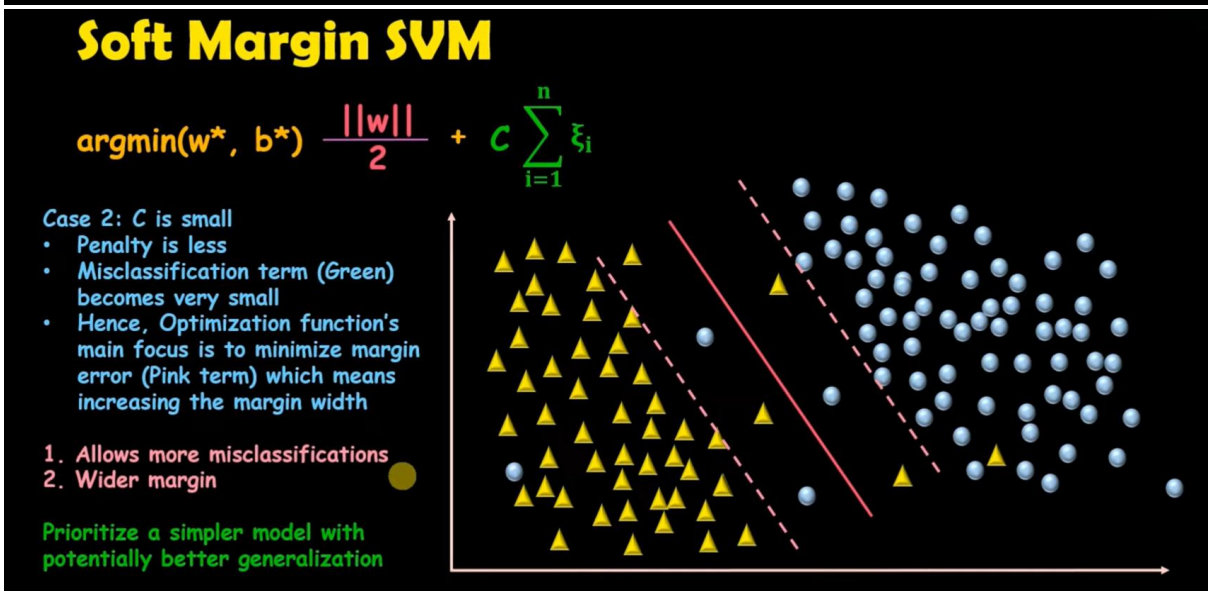
Soft Margin SVM

$$\text{argmin}(w^*, b^*) \frac{||w||}{2} + C \sum_{i=1}^{n} \xi_i$$

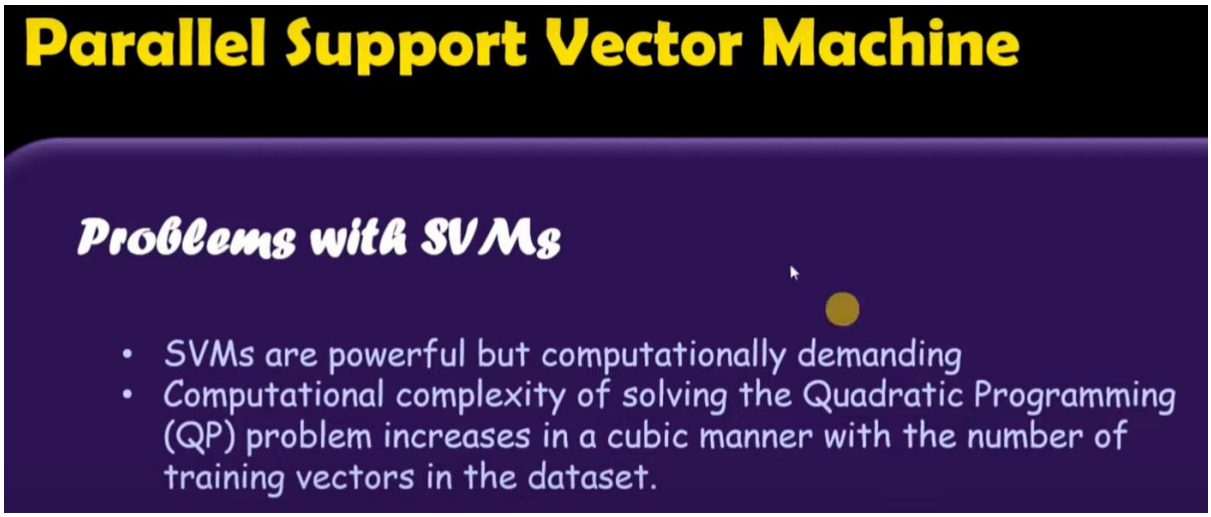
Margin Error

Misclassification term





Parallel SVM

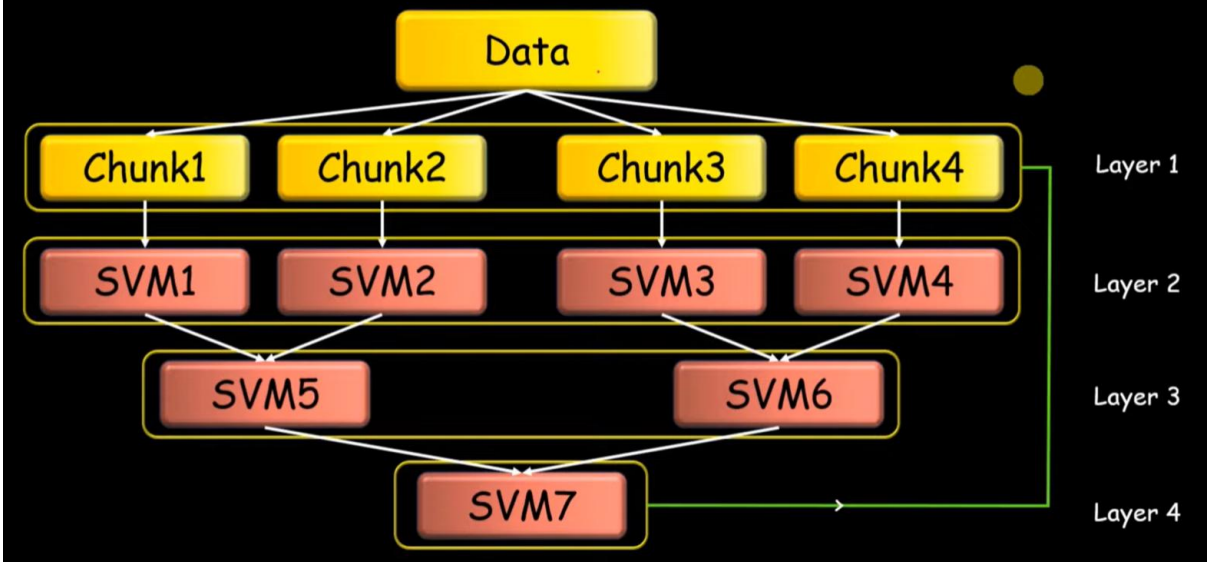


Parallel Support Vector Machine

Overview

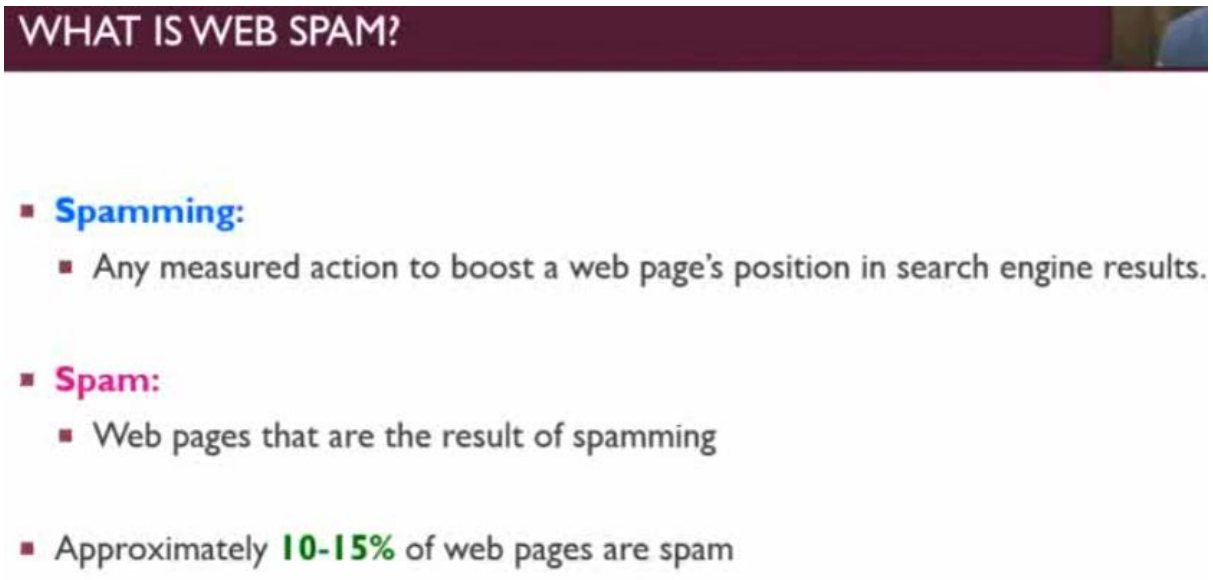
- Sequential Minimal Optimization reduces chunk size to enhance efficiency
- This idea results in eliminating non-support vectors saves computation
- Moreover, shrinking and caching reduce computation requirements
- Digesting optimizes subsets before adding new data
- Guaranteed convergence to the globally optimal solution

Parallel Support Vector Machine




Advantages
Scalable
Faster Training
Efficient Resource Use
Big Data Handling
Improved Convergence
Real-time Processing

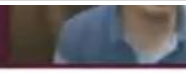
Link Spam


WHAT IS WEB SPAM?
Spamming:
Any measured action to boost a web page's position in search engine results.
Spam:
Web pages that are the result of spamming
Approximately 10-15% of web pages are spam

WEB SPAM TAXONOMY

- Boosting techniques
  - Techniques for achieving high relevance/importance for a web page
- Hiding techniques
  - Techniques to hide the use of boosting
    - **From humans and web crawlers**

BOOSTING TECHNIQUES



- Term spamming
  - Manipulating the text of web pages in order to appear relevant to queries.
- Link spamming
  - Creating link structures that boost page rank or hubs and authorities scores

TERM SPAMMING

- Repetition
  - of one or a few specific terms e.g., free, cheap
  - Goal is to subvert TF.IDF ranking schemes
- Dumping
  - of a large number of unrelated terms
  - e.g., copy entire dictionaries
- Weaving
  - Copy legitimate pages and insert spam terms at random positions
- Phrase Stitching
  - Glue together sentences and phrases from different sources

TERM SPAM TARGETS

- Body of web page
- Title
- URL
- HTML meta tags
- Anchor text

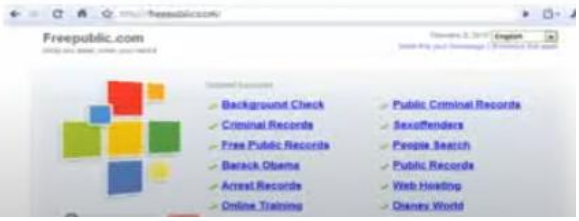
GOOGLE VS SPAMMERS

Once Google became the dominant search engine, spammers began to work out ways to fool Google

Spam farms were developed to concentrate PageRank on a single page

Link spam:

- Creating link structures that boost PageRank of a particular page

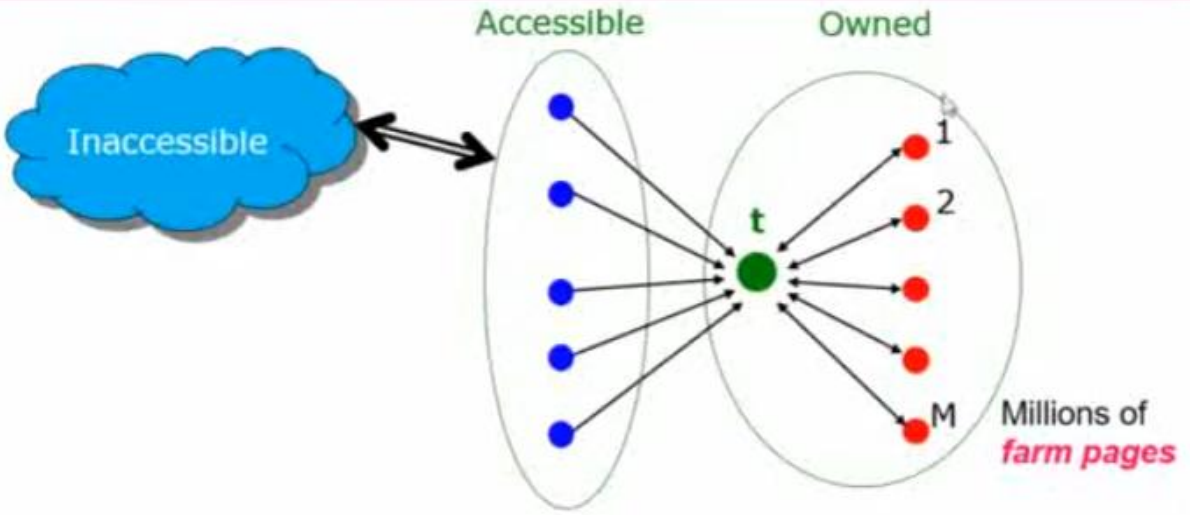


LINK SPAMMING

- **Three kinds of web pages from a spammer's point of view**
  - **Inaccessible pages**
  - **Accessible pages**
    - e.g., blog comments pages
    - spammer can post links to his pages
  - **Owned pages**
    - Completely controlled by spammer
    - May span multiple domain names

LINK FARMS

- **Spammer's goal:**
  - Maximize the Page Rank of target page **t**
- **Technique:**
  - Get as many links from accessible pages as possible to target page **t**
  - Construct "link farm" to get PageRank multiplier effect



One of the most common and effective organizations for a link farm

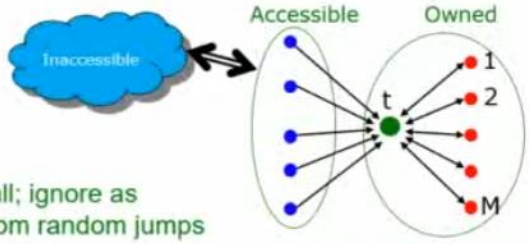
- **x**: PageRank contributed by accessible pages
- **y**: PageRank of target page **t**
- Rank of each "farm" page $= \frac{\beta y}{M} + \frac{1-\beta}{N}$
- $y = x + \beta M\left[\frac{\beta y}{M} + \frac{1-\beta}{N}\right] + \frac{1-\beta}{N}$

$$= x + \beta^2 y + \frac{\beta(1-\beta)M}{N} + \boxed{\frac{1-\beta}{N}}$$

Very small; ignore as comes from random jumps to N pages

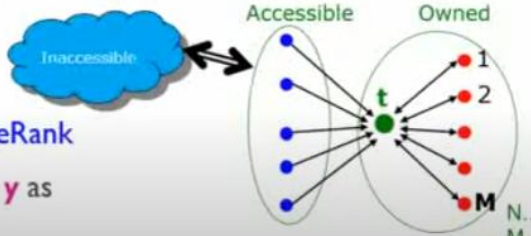
- $y = \frac{x}{1-\beta^2} + c\frac{M}{N}$ where $c = \frac{\beta}{1+\beta}$

Now we solve for **y**



N…# pages on the web
M…# of pages spammer owns
T gives beta fraction of it's page rank to M link farm pages

- $y = \frac{x}{1-\beta^2} + c\frac{M}{N}$ where $c = \frac{\beta}{1+\beta}$ more the M, more the y will be.
- For β = 0.85, $1/(1-\beta^2) = 3.6$



- Multiplier effect for acquired PageRank
- By making **M** large, we can make **y** as large as we want

N…# pages on the web
M…# of pages spammer owns

COMBATING SPAM

- **Combating term spam**
  - Analyze text using statistical methods
  - Similar to email spam filtering
  - Also useful: Detecting approximate duplicate pages
- **Combating link spam**
  - **Detection and blacklisting of structures that look like spam farms**
    - Leads to another war – hiding and detecting spam farms
  - **TrustRank** = topic-specific PageRank with a teleport set of **trusted pages**
    - Example: .edu domains, similar domains for non-US schools

TRUST RANK: IDEA

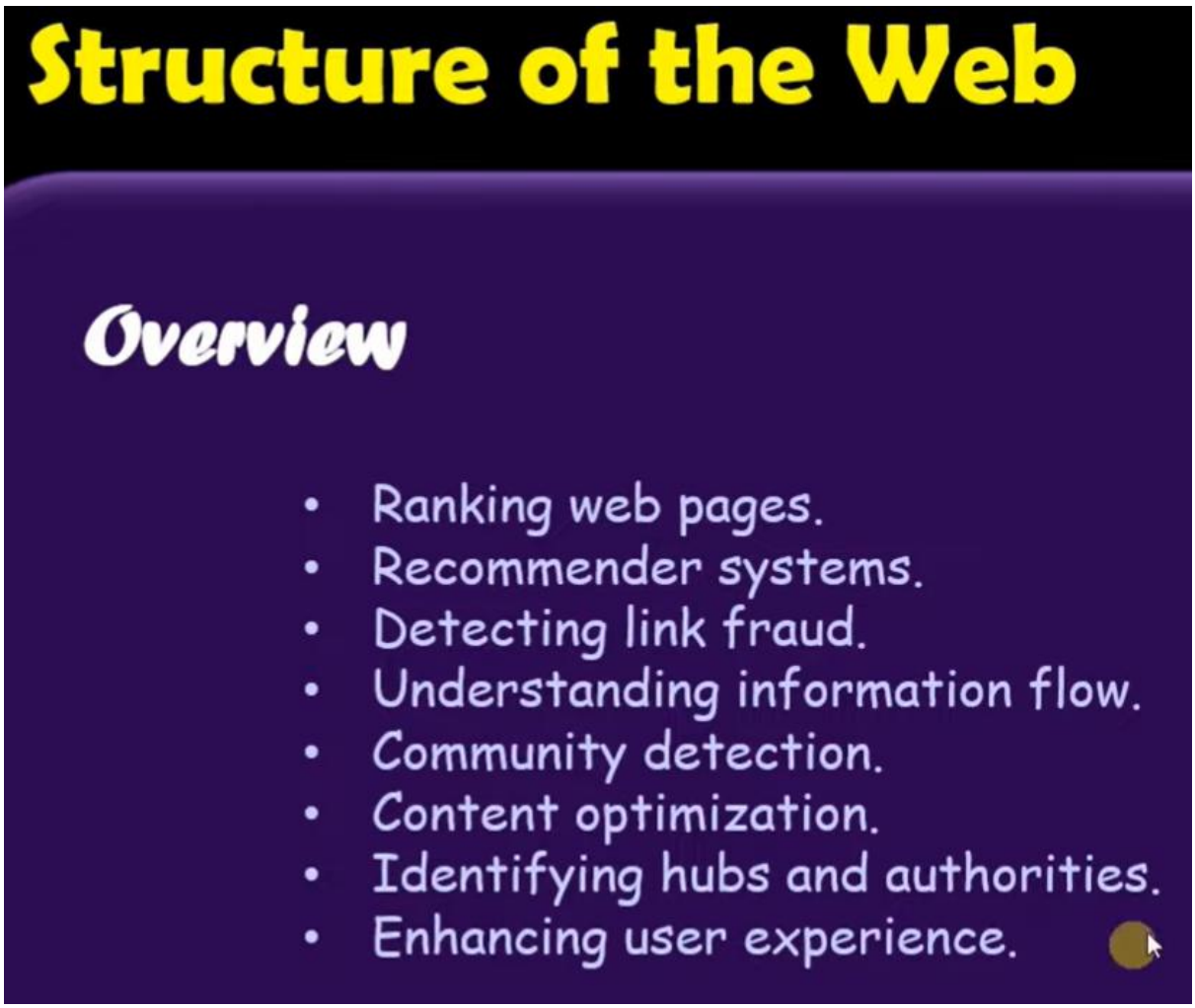
- **Basic principle: Approximate isolation**
  - It is rare for a "good" page to point to a "bad" (spam) page
- Sample a set of **seed pages** from the web
- Have an **human** to identify the good pages and the spam pages in the seed set
  - **Expensive task**, so we must make seed set as small as possible

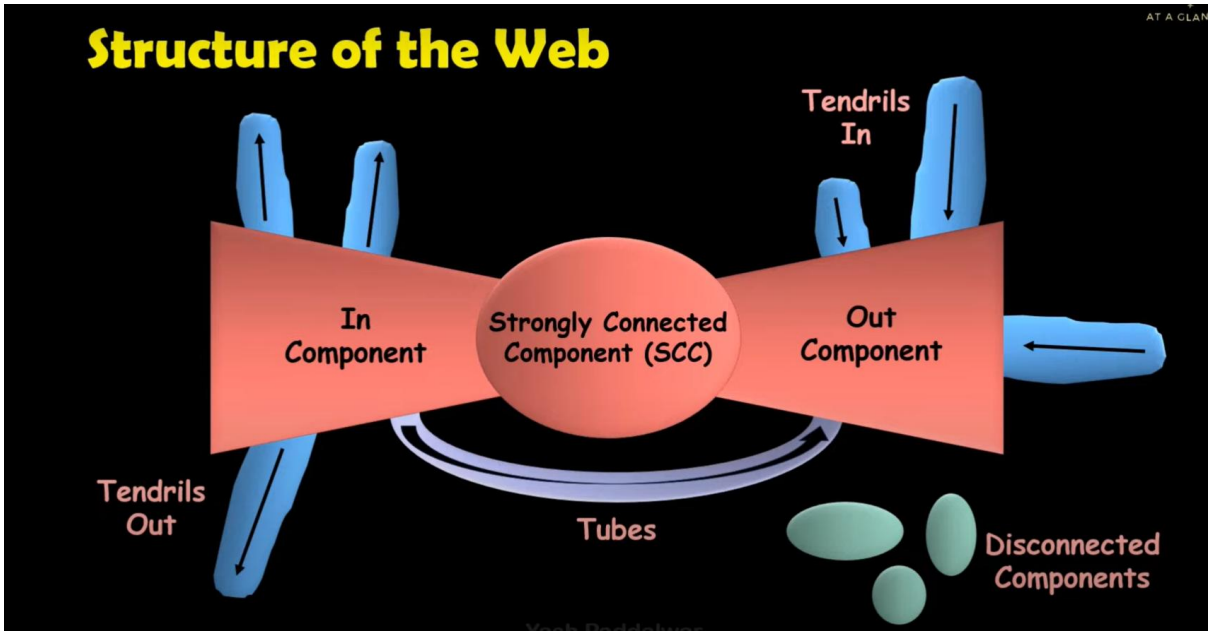
TRUST PROPAGATION

- Call the subset of seed pages that are identified as **good** the **trusted pages**
- Perform a topic-sensitive PageRank with **teleport set = trusted pages**
  - **Propagate trust through links:**
    - Each page gets a trust value between **0** and **1**
- **Solution 1: Use a threshold value and mark all pages below the trust threshold as spam.**

- **Trust attenuation:**
  - The degree of trust conferred by a trusted page decreases with the distance in the graph
- **Trust splitting:**
  - The larger the number of out-links from a page, the less scrutiny the page author gives each out-link
  - Trust is **split** across out-links

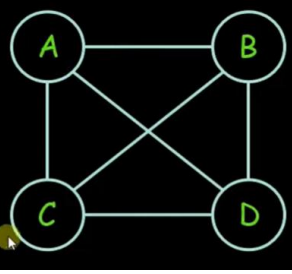
Structure of the Web | Real Life Examples | Link Analysis





Structure of the Web

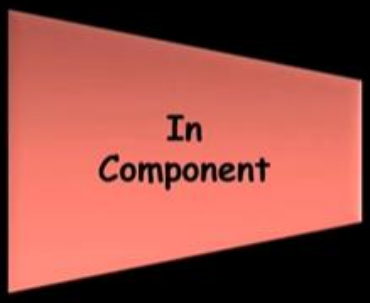
- Groups of web nodes.
- Each node connects to all others.
- Tight community clusters.
- Characterize web connectivity patterns.
- Useful in network analysis.



Structure of the Web

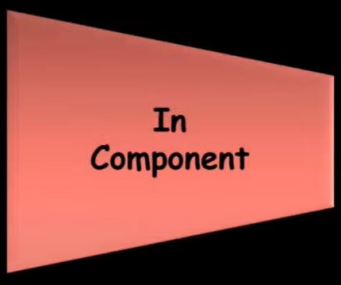
Example

Consider SCC, representing an online store, where all web pages related to products, categories, and checkout are tightly interconnected. In this component, visitors can seamlessly browse and shop for items, creating a self-contained shopping experience within the website.



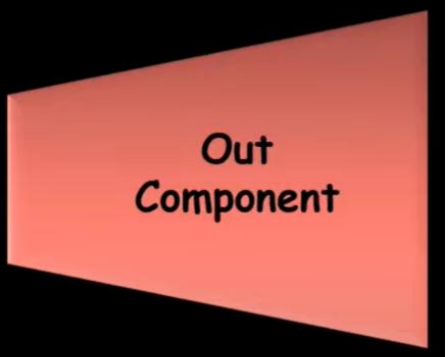
- Consists of webpages that could reach the SCC by following links
- SCC cannot reach the in-component

Example

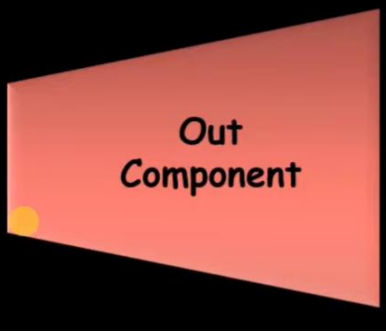


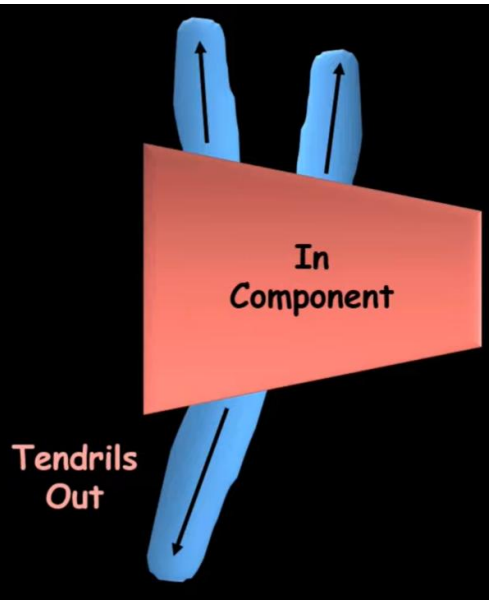
In Component: Your portfolio website
SCC: GitHub Repository

- Consists of webpages that are reachable from the SCC
- They could not reach SCC



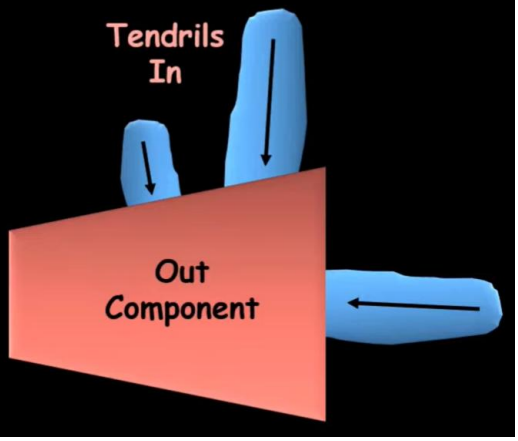
SCC: GitHub Repository
Out Component: Official Documentation





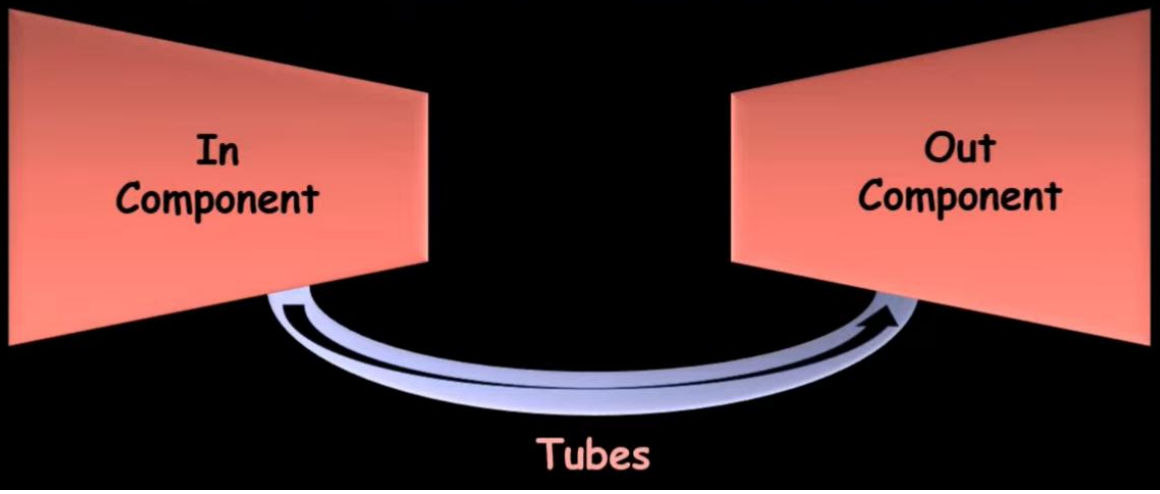
- These are reachable from the in-component but not able to reach the in-component
- Ex: From portfolio website to project page or social media sites

Structure of the Web



- These can reach the out-component but are not reachable from the out-component
- Ex: Some official documentation page cannot point to any third party websites

- Pages that are reachable from in-component and are able to reach the out-component
- They don't reach SCC and are not reachable from SCC



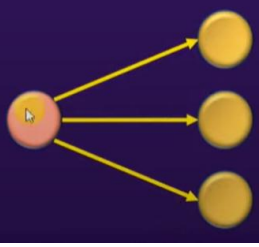
- They cannot reach SCC, in-component and out-component
- The SCC, in-component and out-component also cannot reach them
- Example: A static isolated website

Hubs and Authorities | HITS Algorithm

Hubs and Authorities

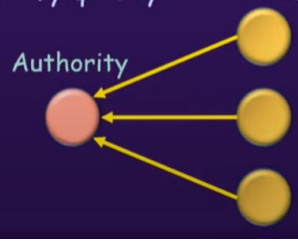
Hubs

- Hubs are nodes with many outgoing links.
- The hub score measures a node's importance as a hub.
- High hub score nodes are good sources of information and connect other nodes.
- Hubs can be identified using algorithms like HITS.
- Hubs are important in search engines, social networks, & recommendation systems.



Authorities

- Authorities are nodes with many incoming links.
- Authorities are nodes that are highly relevant or informative.
- The authority score measures a node's importance as an authority.
- High authority nodes provide valuable content and are linked to by other nodes.
- Authorities are important in search engines, social networks, and recommendation systems, as they help identify quality sources of information.



HITS Algorithm

Overview

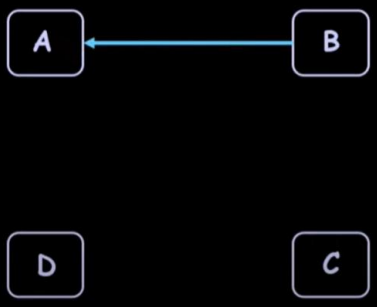
- HITS (Hyperlink-Induced Topic Search) is an algorithm for ranking web pages.
- It measures a page's relevance and importance based on its authority and hub scores.
- HITS algorithm is widely used in search engines to improve the accuracy and relevance of search results.

HITS Algorithm

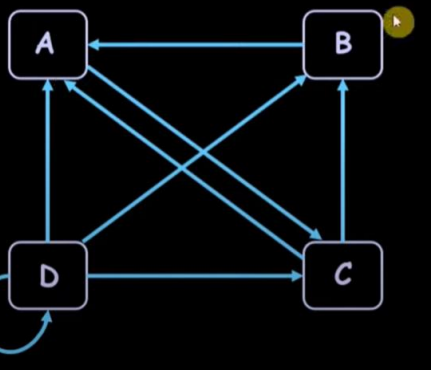
Example

Compute Hub and Authority score for the matrix using the HITS algorithm with k=6, and identify the best authority and hub node:

	A	B	C	D
A	0	1	1	1
B	0	0	1	1
C	1	0	0	1
D	0	0	0	1



	A	B	C	D
A	0	1	1	1
B	0	0	1	1
C	1	0	0	1
D	0	0	0	1

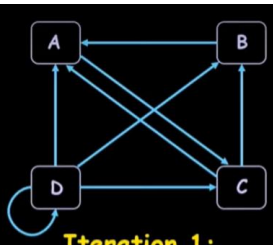


Initially all hubs score will be 1

$Ha = Hb = Hc = Hd = 1$

HITS Algorithm

Example



AT A GLANCE

Iteration 1:

Calculate Authorities Score (Incomings)

Aa = Hb + Hc + Hd = 1 + 1 + 1 = 3
Ab = Hc + Hd = 1 + 1 = 2
Ac = Ha + Hd = 1 + 1 = 2
Ad = Hd = 1

Calculate Hubs Score (Outgoings)

Ha = Ac = 0.25
Hb = Aa = 0.3750
Hc = Aa + Ab = 0.3750 + 0.25 = 0.6250
Hd = Aa + Ab + Ac + Ad = 1

Calculate Normalized Authorities Score

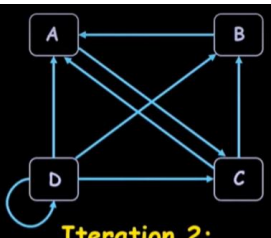
Aa = 3 / 8 = 0.3750
Ab = 2 / 8 = 0.2500
Ac = 2 / 8 = 0.2500
Ad = 1 / 8 = 0.1250

Calculate Normalized Hubs Score

Ha = 0.25 / 2.25 = 0.1111
Hb = 0.3750 / 2.25 = 0.1667
Hc = 0.6250 / 2.25 = 0.2778
Hd = 1 / 2.25 = 0.4444

HITS Algorithm

Example



AT A GLANCE

Previous Iteration Results

Calculate Normalized Hubs Score
Ha = 0.25 / 2.25 = 0.1111
Hb = 0.3750 / 2.25 = 0.1667
Hc = 0.6250 / 2.25 = 0.2778
Hd = 1 / 2.25 = 0.4444

Iteration 2:

Calculate Authorities Score (Incomings)

Aa = Hb + Hc + Hd = 0.8889
Ab = Hc + Hd = 0.7222
Ac = Ha + Hd = 0.5555
Ad = Hd = 0.4444

Calculate Hubs Score (Outgoings)

Ha = Ac = 0.2128
Hb = Aa = 0.3404
Hc = Aa + Ab = 0.3404 + 0.2766 = 0.6170
Hd = Aa + Ab + Ac + Ad = 1

Calculate Normalized Authorities Score

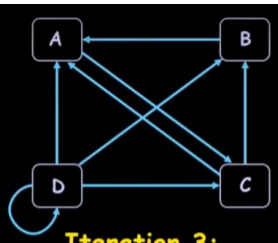
Aa = 0.8889 / 2.611 = 0.3404
Ab = 0.7222 / 2.611 = 0.2766
Ac = 0.5555 / 2.611 = 0.2128
Ad = 0.4444 / 2.611 = 0.1702

Calculate Normalized Hubs Score

Ha = 0.2128 / 2.1702 = 0.0981
Hb = 0.3404 / 2.1702 = 0.1569
Hc = 0.6170 / 2.1702 = 0.2843
Hd = 1 / 2.1702 = 0.4608

HITS Algorithm

Example



Previous Iteration Results

Calculate Normalized Hubs Score
Ha = 0.2128 / 2.1702 = 0.0981
Hb = 0.3404 / 2.1702 = 0.1569
Hc = 0.6170 / 2.1702 = 0.2843
Hd = 1 / 2.1702 = 0.4608

Iteration 3:

Calculate Authorities Score (Incomings)
Aa = Hb + Hc + Hd = 0.9020
Ab = Hc + Hd = 0.7451
Ac = Ha + Hd = 0.5589
Ad = Hd = 0.4608

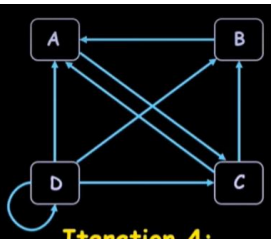
Calculate Hubs Score (Outgoings)
Ha = Ac = 0.2096
Hb = Aa = 0.3382
Hc = Aa + Ab = 0.3382 + 0.2794 = 0.6176
Hd = Aa + Ab + Ac + Ad = 1

Calculate Normalized Authorities Score
Aa = 0.9020 / 2.6668 = 0.3382
Ab = 0.7451 / 2.6668 = 0.2794
Ac = 0.5589 / 2.6668 = 0.2096
Ad = 0.4608 / 2.6668 = 0.1782

Calculate Normalized Hubs Score
Ha = 0.2096 / 2.1654 = 0.0968
Hb = 0.3382 / 2.1654 = 0.1562
Hc = 0.6176 / 2.1654 = 0.2852
Hd = 1 / 2.1654 = 0.4618

HITS Algorithm

Example



Previous Iteration Results

Calculate Normalized Hubs Score
Ha = 0.2096 / 2.1654 = 0.0968
Hb = 0.3382 / 2.1654 = 0.1562
Hc = 0.6176 / 2.1654 = 0.2852
Hd = 1 / 2.1654 = 0.4618

Iteration 4:

Calculate Authorities Score (Incomings)
Aa = Hb + Hc + Hd = 0.9032
Ab = Hc + Hd = 0.7470
Ac = Ha + Hd = 0.5586
Ad = Hd = 0.4618

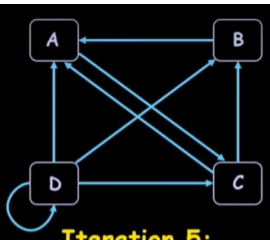
Calculate Hubs Score (Outgoings)
Ha = Ac = 0.2092
Hb = Aa = 0.3382
Hc = Aa + Ab = 0.3382 + 0.2797 = 0.6179
Hd = Aa + Ab + Ac + Ad = 1

Calculate Normalized Authorities Score
Aa = 0.9032 / 2.6706 = 0.3382
Ab = 0.7470 / 2.6706 = 0.2797
Ac = 0.5586 / 2.6706 = 0.2092
Ad = 0.4618 / 2.6706 = 0.1729

Calculate Normalized Hubs Score
Ha = 0.2092 / 2.1653 = 0.0966
Hb = 0.3382 / 2.1653 = 0.1562
Hc = 0.6179 / 2.1653 = 0.2854
Hd = 1 / 2.1653 = 0.4618

HITS Algorithm

Example



Previous Iteration Results

Calculate Normalized Hubs Score
Ha = 0.2092 / 2.1653 = 0.0966
Hb = 0.3382 / 2.1653 = 0.1562
Hc = 0.6179 / 2.1653 = 0.2854
Hd = 1 / 2.1653 = 0.4618

Iteration 5:

Calculate Authorities Score (Incomings)
Aa = Hb + Hc + Hd = 0.9034
Ab = Hc + Hd = 0.7472
Ac = Ha + Hd = 0.5584
Ad = Hd = 0.4618

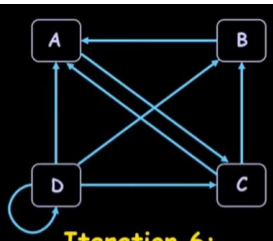
Calculate Hubs Score (Outgoings)
Ha = Ac = 0.2091
Hb = Aa = 0.3383
Hc = Aa + Ab = 0.3383 + 0.2798 = 0.6181
Hd = Aa + Ab + Ac + Ad = 1

Calculate Normalized Authorities Score
Aa = 0.9034 / 2.6708 = 0.3383
Ab = 0.7472 / 2.6708 = 0.2798
Ac = 0.5584 / 2.6708 = 0.2091
Ad = 0.4618 / 2.6708 = 0.1729

Calculate Normalized Hubs Score
Ha = 0.2091 / 2.1655 = 0.0966
Hb = 0.3383 / 2.1655 = 0.1562
Hc = 0.6181 / 2.1655 = 0.2854
Hd = 1 / 2.1655 = 0.4618

HITS Algorithm

Example



Previous Iteration Results

Calculate Normalized Hubs Score
Ha = 0.2092 / 2.1653 = 0.0966
Hb = 0.3382 / 2.1653 = 0.1562
Hc = 0.6179 / 2.1653 = 0.2854
Hd = 1 / 2.1653 = 0.4618

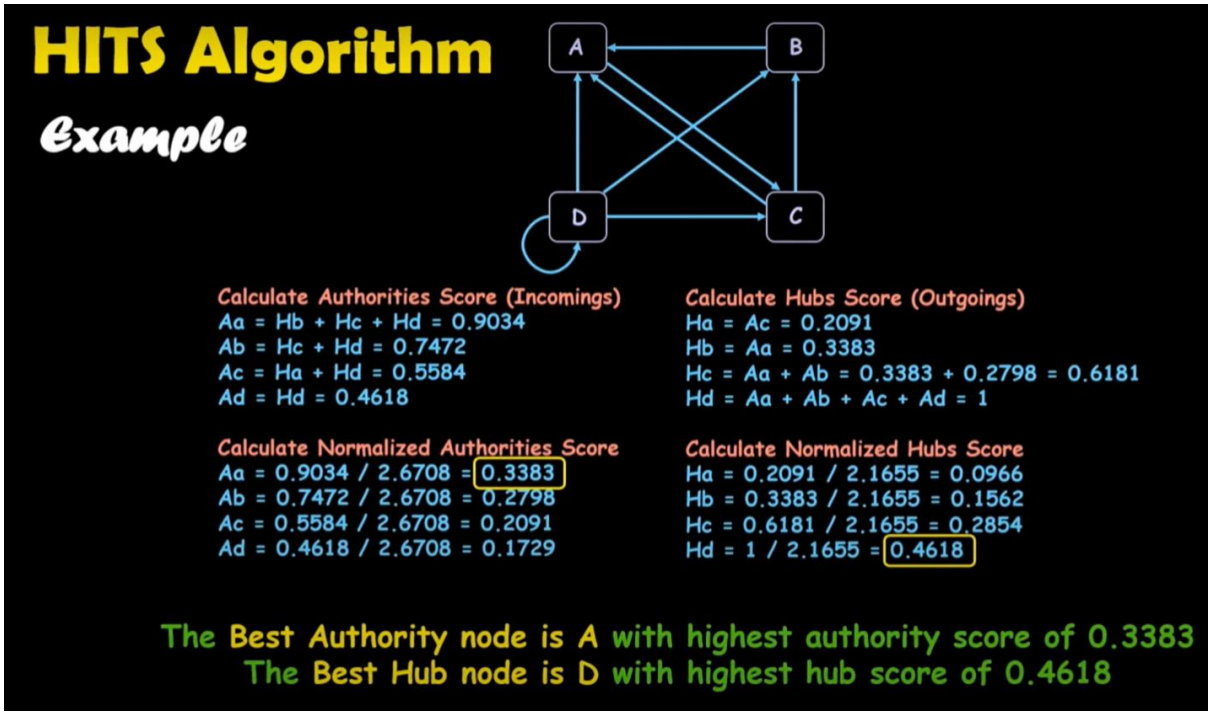
Iteration 6:

Calculate Authorities Score (Incomings)
Aa = Hb + Hc + Hd = 0.9034
Ab = Hc + Hd = 0.7472
Ac = Ha + Hd = 0.5584
Ad = Hd = 0.4618

Calculate Hubs Score (Outgoings)
Ha = Ac = 0.2091
Hb = Aa = 0.3383
Hc = Aa + Ab = 0.3383 + 0.2798 = 0.6181
Hd = Aa + Ab + Ac + Ad = 1

Calculate Normalized Authorities Score
Aa = 0.9034 / 2.6708 = 0.3383
Ab = 0.7472 / 2.6708 = 0.2798
Ac = 0.5584 / 2.6708 = 0.2091
Ad = 0.4618 / 2.6708 = 0.1729

Calculate Normalized Hubs Score
Ha = 0.2091 / 2.1655 = 0.0966
Hb = 0.3383 / 2.1655 = 0.1562
Hc = 0.6181 / 2.1655 = 0.2854
Hd = 1 / 2.1655 = 0.4618

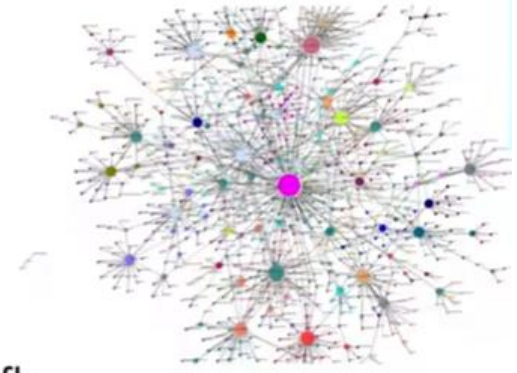


PageRank Algorithm | Link Analysis

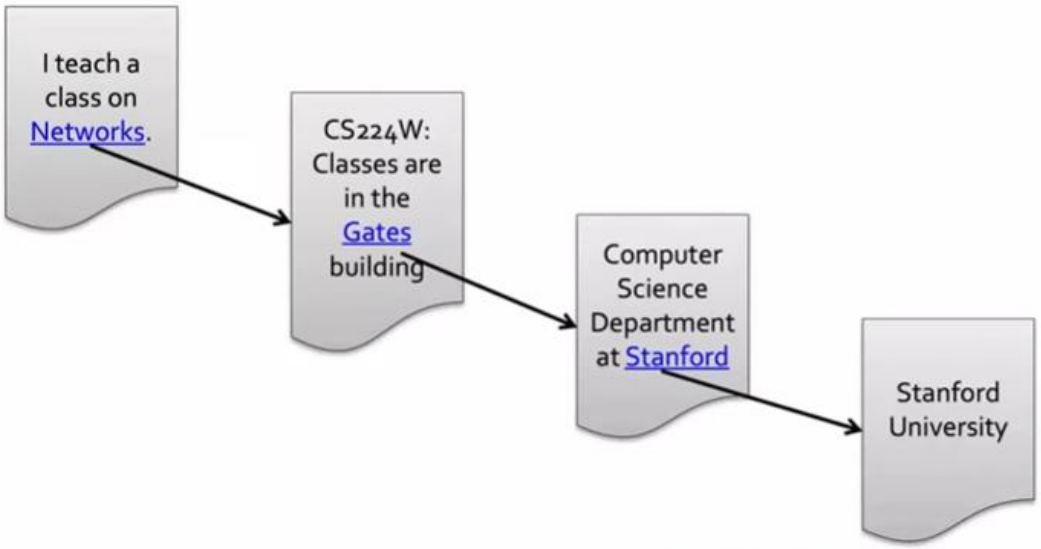
PageRank algorithm, fully explained | by Amrani Amine | Towards Data Science
linear algebra - Why is PageRank an eigenvector problem? - Mathematics Stack Exchange
model evaluations - Is there any PageRank-like method on weighted graph? - Data Science Stack Exchange
page rank algorithm stanford university - YouTube

PageRank Algorithm - YouTube

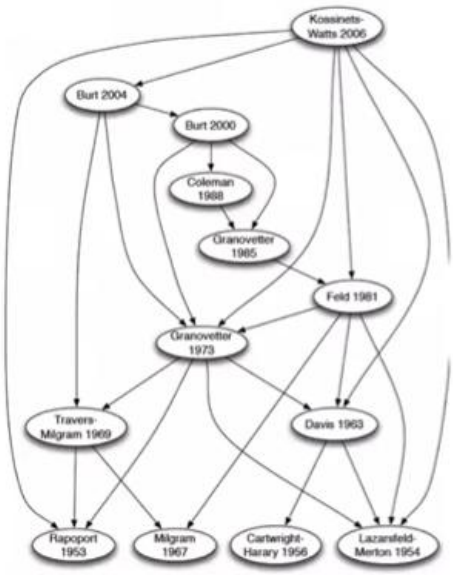
Q: What does the Web "look like" at a global level?



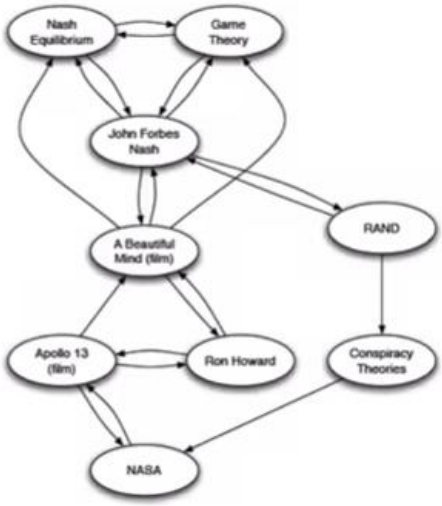
- **Web as a graph:**
  - Nodes = web pages
  - Edges = hyperlinks
  - **Side issue:** What is a node?
    - Dynamic pages created on the fly
    - "dark matter" – inaccessible database generated pages



- In early days of the Web links were **navigational**
- Today many links are **transactional** (used not to navigate from page to page, but to post, comment, like, buy, …)



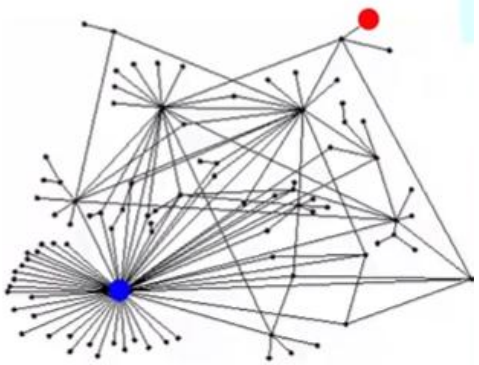
Citations



References in an Encyclopedia

- **All web pages are not equally "important"**
  thispersondoesnotexist.com vs. www.stanford.edu

- There is large diversity in the web-graph node connectivity.
- **So, let's rank the pages using the web graph link structure!**



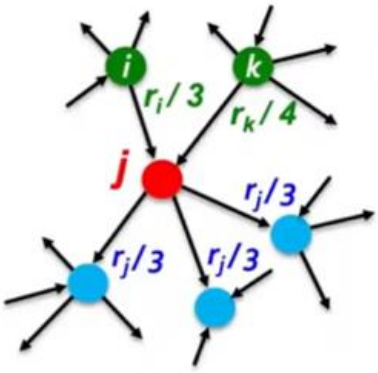
- We will cover the following **Link Analysis approaches** to compute the **importance** of nodes in a graph:
  - PageRank
  - Personalized PageRank (PPR)
  - Random Walk with Restarts

Links as Votes

- **Idea: Links as votes**
  - **Page is more important if it has more links**
    - In-coming links? Out-going links?
- **Think of in-links as votes:**
  - www.stanford.edu has 23,400 in-links
  - thispersondoesnotexist.com has 1 in-link
- **Are all in-links equal?**
  - Links from important pages count more
  - Recursive question!

PageRank: The "Flow" Model

- **A "vote" from an important page is worth more:**
  - Each link's vote is proportional to the **importance** of its source page
  - If page ***i*** with importance r_i has d_i out-links, each link gets r_i / d_i votes
  - Page ***j***'s own importance r_j is the sum of the votes on its in-links



$r_j = r_i/3 + r_k/4$

PageRank: The "Flow" Model

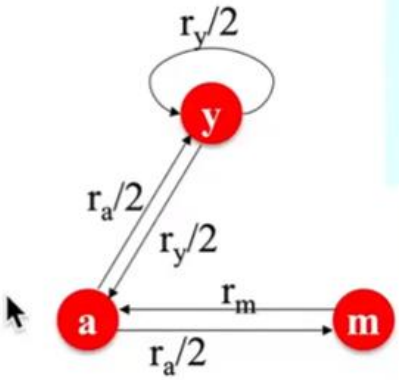
- **A page is important if it is pointed to by other important pages**
- **Define "rank" r_j for node j**

$$r_j = \sum_{i \to j} \frac{r_i}{d_i}$$

d_i … out-degree of node i

You might wonder: Let's just use Gaussian elimination to solve this system of linear equations. Bad idea!

The web in 1839



"Flow" equations:

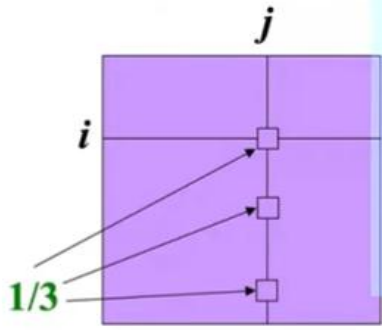
$r_y = r_y/2 + r_a/2$

$r_a = r_y/2 + r_m$

$r_m = r_a/2$

PageRank: Matrix Formulation

- **Stochastic adjacency matrix M**
  - Let page j have d_j out-links
  - If $j \rightarrow i$, then $M_{ij} = \frac{1}{d_j}$
    - M is a **column stochastic matrix**
      - **Columns** sum to **1**

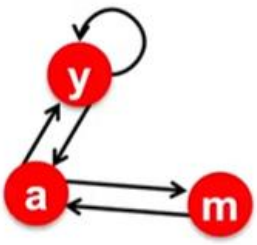


M

- **Rank vector r:** An entry per page
  - r_i is the importance score of page i
  - $\sum_i r_i = 1$
- **The flow equations can be written**

$$r = M \cdot r \qquad r_j = \sum_{i \rightarrow j} \frac{r_i}{d_i}$$

Example: Flow Equations & M



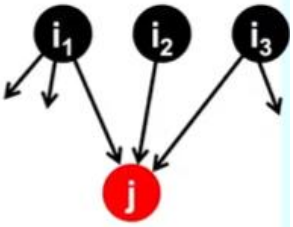
	r_y	r_a	r_m
r_y	½	½	0
r_a	½	0	1
r_m	0	½	0

$$r_y = r_y/2 + r_a/2$$
$$r_a = r_y/2 + r_m$$
$$r_m = r_a/2$$

$$\begin{bmatrix} r_y \\ r_a \\ r_m \end{bmatrix} = \begin{bmatrix} ½ & ½ & 0 \\ ½ & 0 & 1 \\ 0 & ½ & 0 \end{bmatrix} \begin{bmatrix} r_y \\ r_a \\ r_m \end{bmatrix}$$

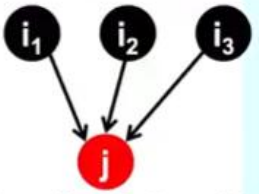
r M r

Connection to Random Walk



- **Imagine a random web surfer:**
  - At any time t, surfer is on some page i
  - At time $t+1$, the surfer follows an out-link from i uniformly at random
  - Ends up on some page j linked from i
  - Process repeats indefinitely
- **Let:**
  - $p(t)$ ... vector whose i^{th} coordinate is the prob. that the surfer is at page i at time t
  - So, $p(t)$ is a probability distribution over pages

The Stationary Distribution



$p(t+1) = M \cdot p(t)$

- **Where is the surfer at time $t+1$?**
  - Follow a link uniformly at random

$$p(t+1) = M \cdot p(t)$$

- Suppose the random walk reaches a state

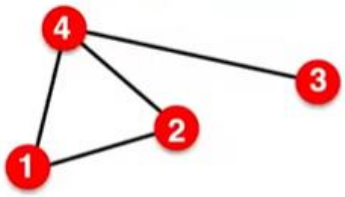
$$p(t+1) = M \cdot p(t) = p(t)$$

then $p(t)$ is **stationary distribution** of a random walk

- **Our original rank vector r** satisfies $r = M \cdot r$
  - **So, r is a stationary distribution for the random walk**

Recall Eigenvector of A Matrix

- Recall from lecture 2 (eigenvector centrality), let $A \in \{0,1\}^{n \times n}$ be an adj. matrix of undir. graph:



$$A = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

- Eigenvector of adjacency matrix: vectors satisfying $\lambda c = Ac$
- c: eigenvector; λ: eigenvalue

- Note:
  - This is the definition of eigenvector centrality (for undirected graphs).
  - PageRank is defined for directed graphs

Eigenvector Formulation

- **The flow equation:**

$$1 \cdot r = M \cdot r$$

$$\begin{bmatrix} r_y \\ r_a \\ r_m \end{bmatrix} = \begin{bmatrix} ½ & ½ & 0 \\ ½ & 0 & 1 \\ 0 & ½ & 0 \end{bmatrix} \begin{bmatrix} r_y \\ r_a \\ r_m \end{bmatrix}$$

r M r

- So the **rank vector** r is an **eigenvector** of the stochastic ajd. matrix M (with eigenvalue 1)
  - Starting from any vector u, the limit $M(M(\dots M(M\,u)))$ is the **long-term distribution** of the surfers.
    - **PageRank** = Limiting distribution = **principal eigenvector** of M
    - Note: If r is the limit of the product $MM \dots Mu$, then r satisfies the **flow equation** $1 \cdot r = Mr$
    - So r is the **principal eigenvector** of M with eigenvalue 1
- **We can now efficiently solve for *r*!**
  - The method is called **Power iteration**

PageRank Algorithm

Overview

- Google's founders developed the algorithm.
- Algorithm used to rank web pages.
- Counts links as "votes" for authority.
- Determines page score based on links.
- High scores = more relevance.
- Scores updated periodically for accuracy.

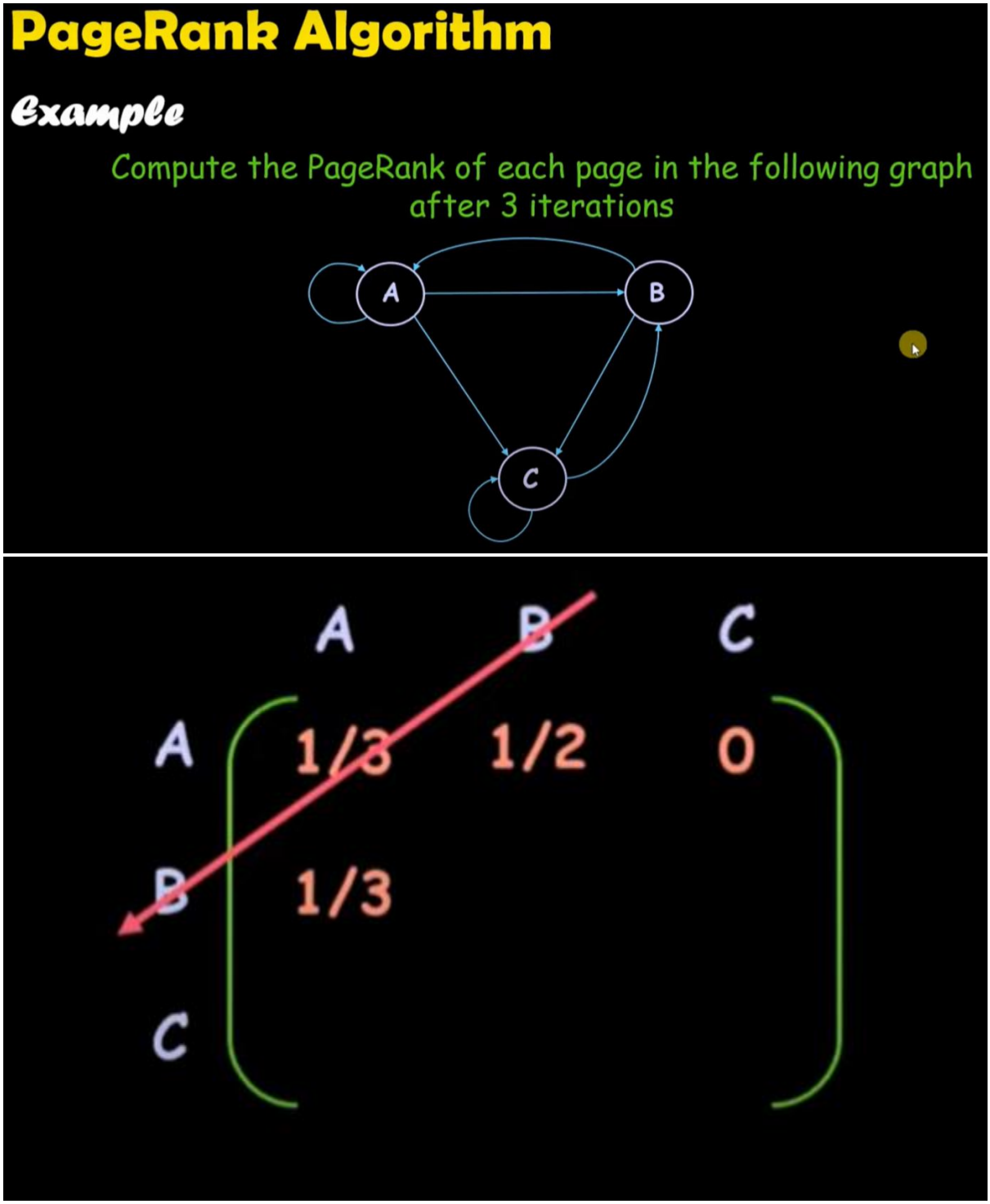
PageRank Algorithm

Algorithm

1. Consider a website with n different webpages
2. Set vector r_0 as 1/n for each webpage
3. Compute transition matrix M
4. For calculating next vector r_1: $r_1 = M*(r_0)$
5. For calculating next vector r_2: $r_2 = M*(r_1)$
6. Hence in general next vector can be calculated as:

$$r_{i+1} = M*(r_i)$$

7. Distribution vector r is the PageRank of each webpage



The numbers in the matrix are what's the probability of going from node A to node A. So we check the outlinks from A which are 3 and there's one outlink which comes back to A itself. So probability of A going to A is ⅓.

$$M$$

	A	B	C
A	1/3	1/2	0
B	1/3	0	1/2
C	1/3	1/2	1/2

$$M \qquad r_0$$

	A	B	C
A	1/3	1/2	0
B	1/3	0	1/2
C	1/3	1/2	1/2

$$\begin{pmatrix} 1/3 \\ 1/3 \\ 1/3 \end{pmatrix}$$

Example

Formula, $r_{i+1} = M \times r_i$

Iteration 1

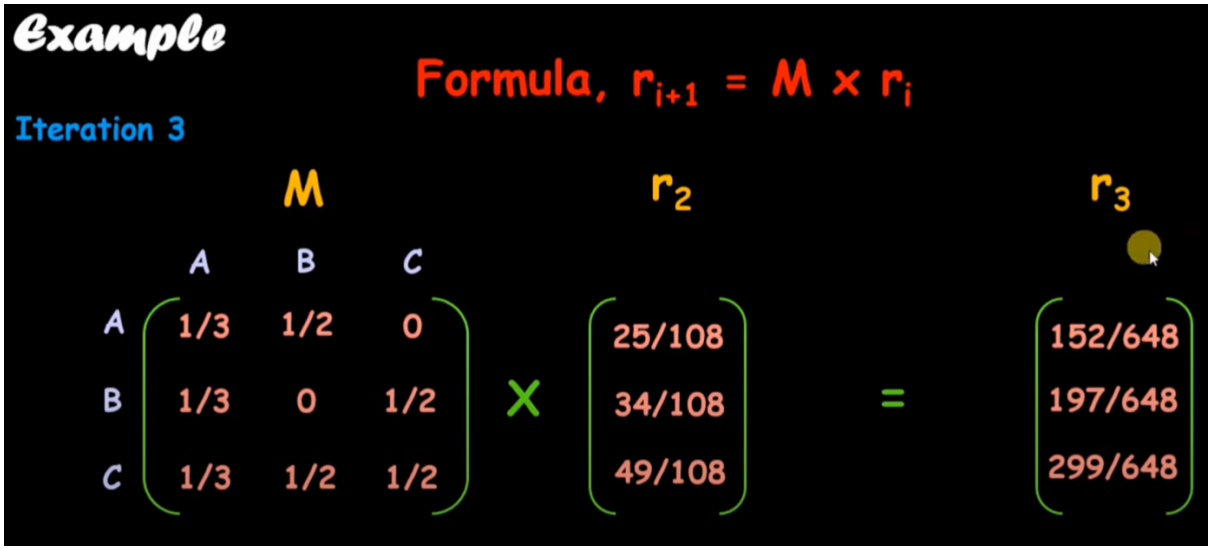
	M				r_0		r_1
	A	B	C				
A	1/3	1/2	0		1/3		5/18
B	1/3	0	1/2	X	1/3	=	5/18
C	1/3	1/2	1/2		1/3		8/18

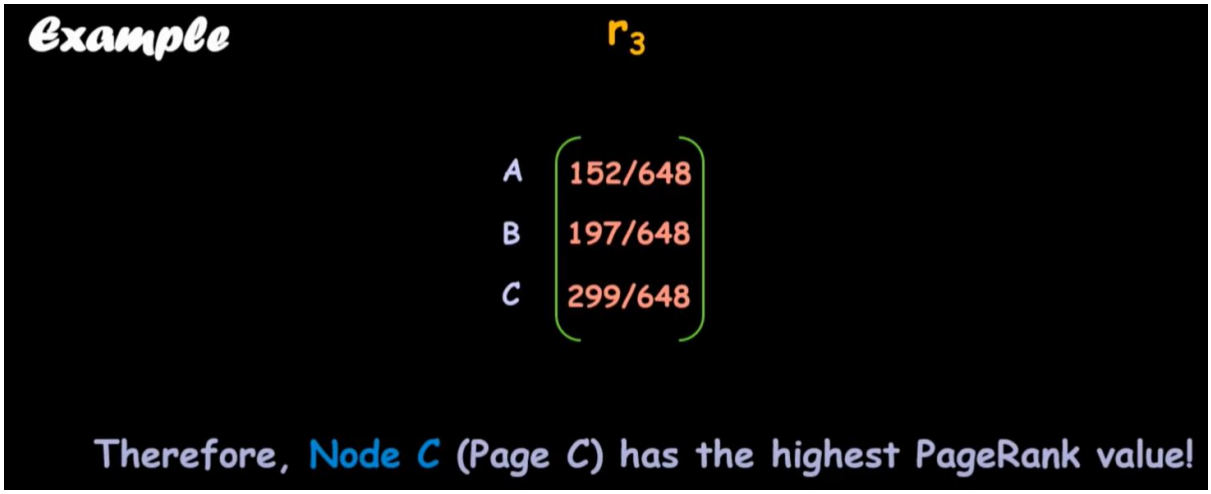
Example

Formula, $r_{i+1} = M \times r_i$

Iteration 2

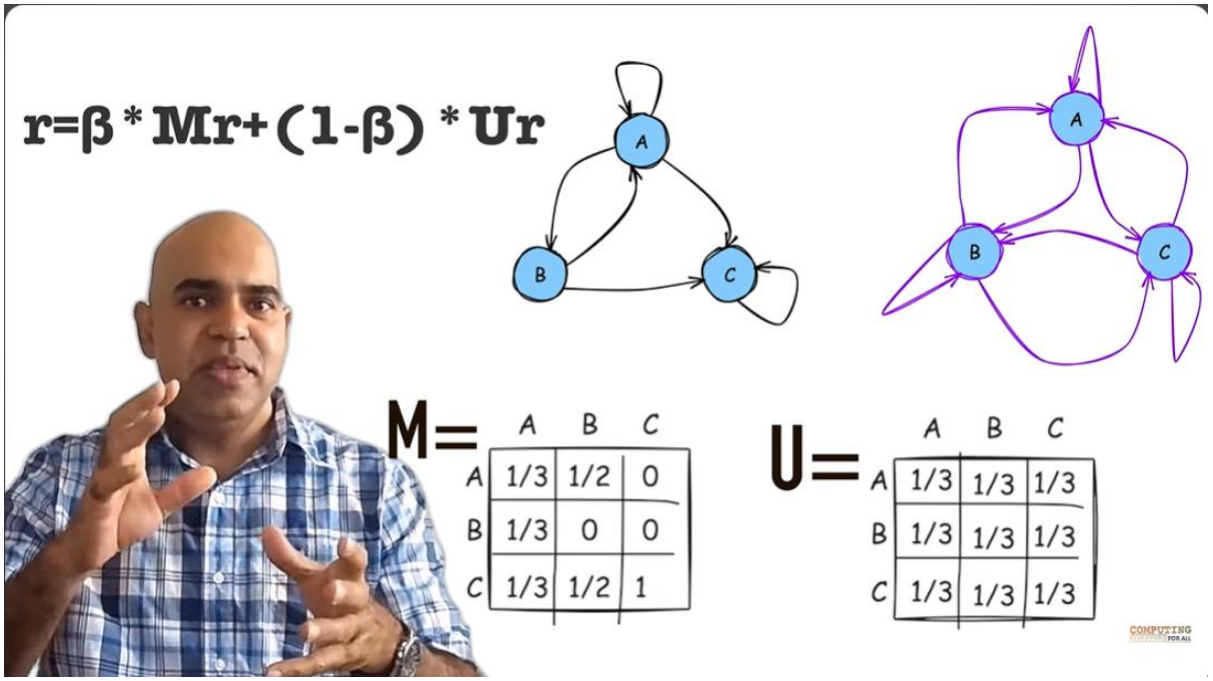
	M				r_1		r_2
	A	B	C				
A	1/3	1/2	0		5/18		25/108
B	1/3	0	1/2	X	5/18	=	34/108
C	1/3	1/2	1/2		8/18		49/108





Topic Specific Page rank

For Dead ends and Spider traps



Topic-Specific PageRank

- **Instead of generic popularity, can we measure popularity within a topic?**
- **Goal:** Evaluate Web pages not just according to their popularity, but by how close they are to a particular topic, e.g. "sports" or "history"
- **Allows search queries to be answered based on interests of the user**
  - **Example:** Query "Trojan" wants different pages depending on whether you are interested in sports, history and computer security

Topic-Specific PageRank

- Random walker has a small probability of teleporting at any step
- **Teleport can go to:**
  - **Standard PageRank: Any page with equal probability**
    - To avoid dead-end and spider-trap problems
  - **Topic Specific PageRank: A topic-specific set of "relevant" pages (teleport set)**
- **Idea: Bias the random walk**
  - When walker teleports, she pick a page from a set S
  - S contains only pages that are relevant to the topic
    - E.g., Open Directory (DMOZ) pages for a given topic/query
  - For each teleport set S, we get a different vector r_S

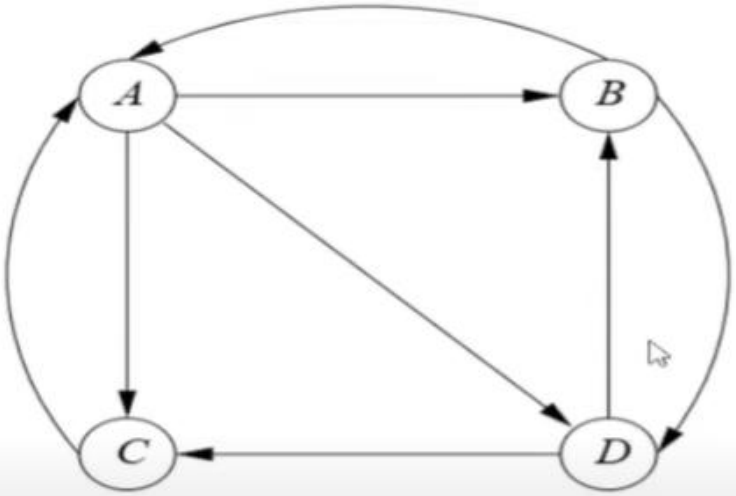
Matrix Formulation

- **To make this work all we need is to update the teleportation part of the PageRank formulation:**

$$A_{ij} = \begin{cases} \beta M_{ij} + (1-\beta)/|S| & \text{if } i \in S \\ \beta M_{ij} + 0 & \text{otherwise} \end{cases}$$

  - A is stochastic!
- We weighted all pages in the teleport set S equally
  - **Could also assign different weights to pages!**
- **Compute as for regular PageRank:**
  - Multiply by M, then add a vector
  - Maintains sparseness

Numerical#01



- the teleport set $S = \{B, D\}$
- Assume $\beta = 0.8$

$$\beta M = \begin{bmatrix} 0 & 2/5 & 4/5 & 0 \\ 4/15 & 0 & 0 & 2/5 \\ 4/15 & 0 & 0 & 2/5 \\ 4/15 & 2/5 & 0 & 0 \end{bmatrix}$$

$$\mathbf{v}' = \begin{bmatrix} 0 & 2/5 & 4/5 & 0 \\ 4/15 & 0 & 0 & 2/5 \\ 4/15 & 0 & 0 & 2/5 \\ 4/15 & 2/5 & 0 & 0 \end{bmatrix} \mathbf{v} + \begin{bmatrix} 0 \\ 1/10 \\ 0 \\ 1/10 \end{bmatrix}$$

$$\begin{bmatrix} 0/2 \\ 1/2 \\ 0/2 \\ 1/2 \end{bmatrix}, \begin{bmatrix} 2/10 \\ 3/10 \\ 2/10 \\ 3/10 \end{bmatrix}, \begin{bmatrix} 42/150 \\ 41/150 \\ 26/150 \\ 41/150 \end{bmatrix}, \begin{bmatrix} 62/250 \\ 71/250 \\ 46/250 \\ 71/250 \end{bmatrix}, \ldots, \begin{bmatrix} 54/210 \\ 59/210 \\ 38/210 \\ 59/210 \end{bmatrix}$$

PageRank Algorithm with Taxation

PageRank Algorithm with Taxation

Overview

- Addresses the drawbacks of PageRank Algorithm
- Accounts for the possibility that user may randomly jump to another page
- Introduces a probability (damping factor) of randomly jumping to another page
- Helps to mitigate issues with convergence, bias towards larger websites, and vulnerability to link farms.
- More efficient than PageRank without Taxation

Algorithm

1. Consider a website with n different webpages
2. Set vector r_0 as 1/n for each webpage
3. Compute transition matrix M
4. Compute the matrix A

$$A = \beta * M + (1-\beta)[1/n]_{nxn}$$ where, β is taxation factor

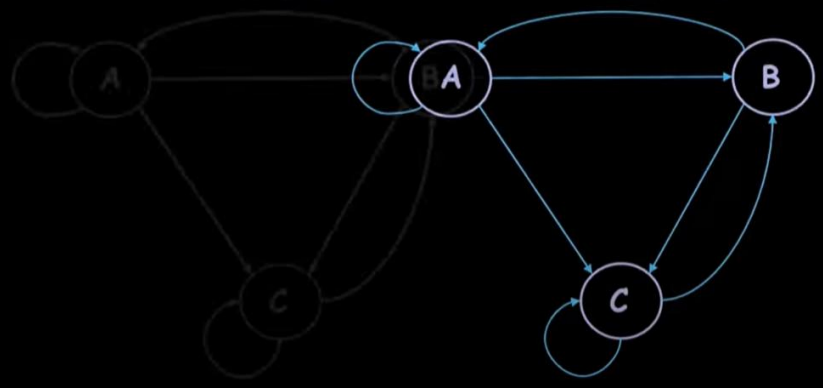
5. For calculating next vector r_1: $r_1 = A*(r_0)$
6. For calculating next vector r_2: $r_2 = A*(r_1)$
7. Hence in general next vector can be calculated as:

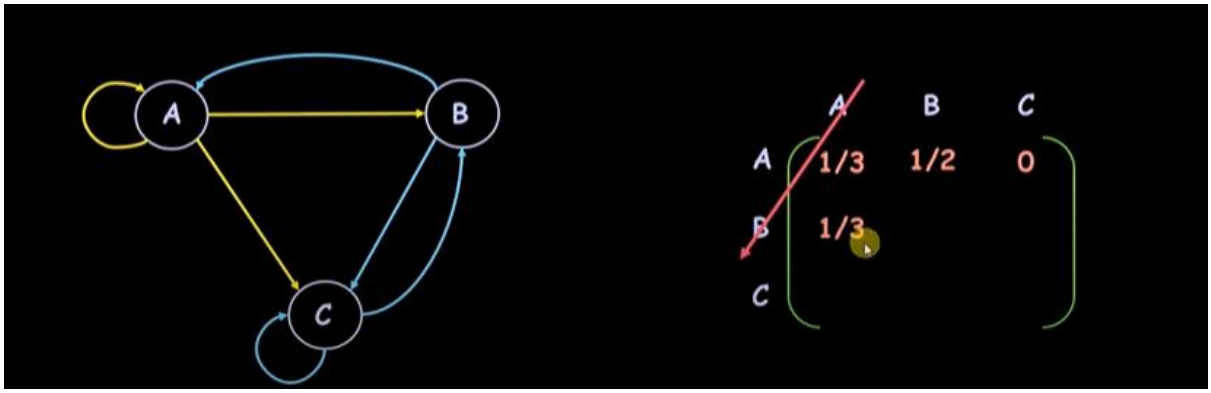
$$r_{i+1} = A*(r_i)$$

8. Distribution vector r is the PageRank of each webpage

Example

Compute the PageRank with a taxation factor of 0.8 of each page in the following graph after 3 iterations





M

	A	B	C
A	1/3	1/2	0
B	1/3	0	1/2
C	1/3	1/2	1/2

Formula, $A = \beta * M + (1-\beta)[1/n]_{n \times n}$

β M $1-\beta$ $[1/n]_{n \times n}$

$$0.8 \times \begin{array}{c} \\ A \\ B \\ C \end{array} \begin{array}{c} \begin{array}{ccc} A & B & C \end{array} \\ \begin{pmatrix} 1/3 & 1/2 & 0 \\ 1/3 & 0 & 1/2 \\ 1/3 & 1/2 & 1/2 \end{pmatrix} \end{array} \quad (1-0.8) \times \begin{pmatrix} 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \end{pmatrix}$$

3 x 3

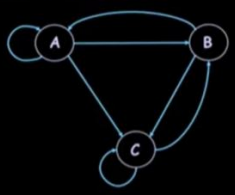
$\beta * M$ $\qquad$ $(1-\beta) * [1/n]_{nxn}$

$$\begin{pmatrix} 4/15 & 4/10 & 0 \\ 4/15 & 0 & 4/10 \\ 4/15 & 4/10 & 4/10 \end{pmatrix} + \begin{pmatrix} 1/15 & 1/15 & 1/15 \\ 1/15 & 1/15 & 1/15 \\ 1/15 & 1/15 & 1/15 \end{pmatrix}$$

$$\begin{pmatrix} 5/15 & 7/15 & 1/15 \\ 5/15 & 1/15 & 7/15 \\ 5/15 & 7/15 & 7/15 \end{pmatrix}$$

PageRank Algorithm with Taxation

Example



Formula, $r_{i+1} = A \times r_i$

Iteration 1

A $\qquad$ r_0 $\qquad$ r_1

$$\begin{array}{c} \\ A \\ B \\ C \end{array} \begin{array}{c} \begin{array}{ccc} A & B & C \end{array} \\ \begin{pmatrix} 5/15 & 7/15 & 1/15 \\ 5/15 & 1/15 & 7/15 \\ 5/15 & 7/15 & 7/15 \end{pmatrix} \end{array} \times \begin{pmatrix} 1/3 \\ 1/3 \\ 1/3 \end{pmatrix} = \begin{pmatrix} 13/45 \\ 13/45 \\ 19/45 \end{pmatrix}$$

Example

Formula, $r_{i+1} = A \times r_i$

Iteration 2

	A				r_1		r_2
	A	B	C				
A	5/15	7/15	1/15		13/45		175/675
B	5/15	1/15	7/15	X	13/45	=	211/675
C	5/15	7/15	7/15		19/45		289/675

Example

Formula, $r_{i+1} = A \times r_i$

Iteration 3

	A				r_2		r_3
	A	B	C				
A	5/15	7/15	1/15		175/675		2641/10125
B	5/15	1/15	7/15	X	211/675	=	3109/10125
C	5/15	7/15	7/15		289/675		4375/10125

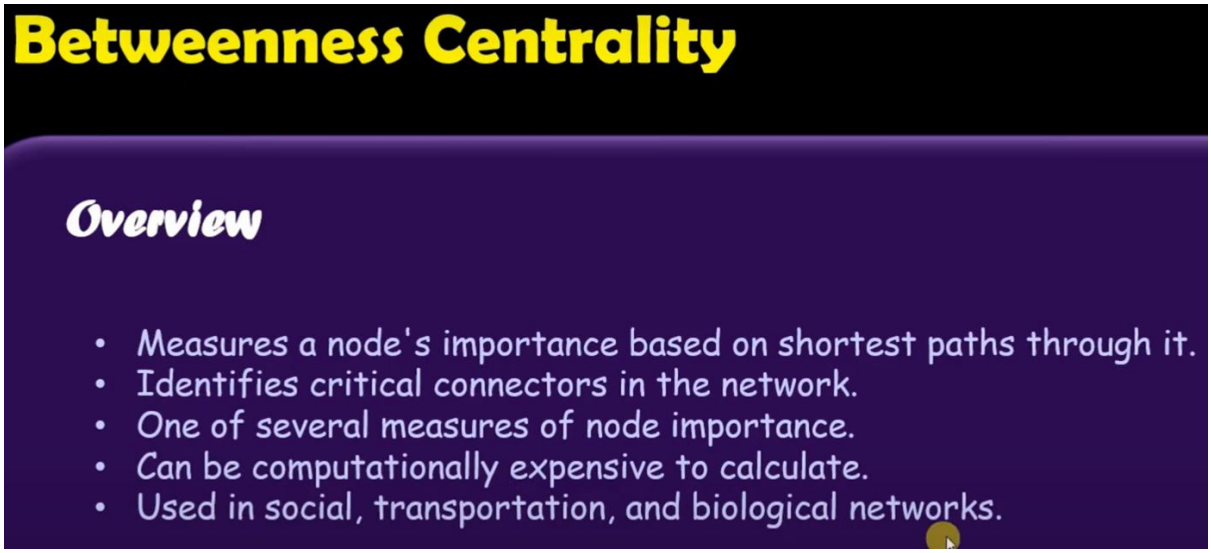
Example

r_3

A	2641/10125
B	3109/10125
C	4375/10125

Therefore, Node C (Page C) has the highest PageRank value!

Betweenness Centrality



Example

Suppose you have a social network of 10 people, and you want to identify who are the key people connecting different parts of the network. You calculate the betweenness centrality of each person in the network and find that person A has the highest betweenness centrality. This means that person A acts as a bridge or connector between different groups of people in the network and is critical for information flow and communication. Removing person A from the network would likely have a significant impact on the network's overall structure and function.

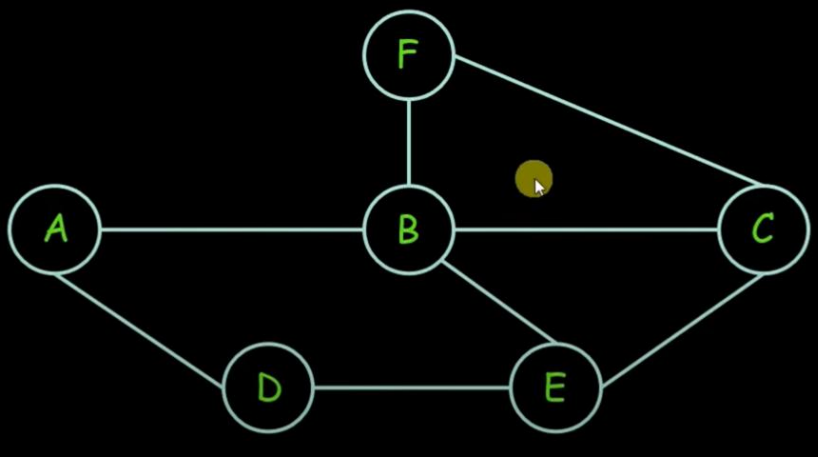
Formula

Betweenness centrality BC(y) of a vertex y is given as:

$$BC(y) = \sum_{x<z} \frac{\sigma_{xz}(y)}{\sigma_{xz}}$$

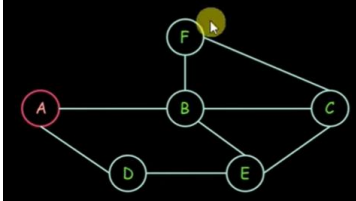
where, $\sigma_{xz}(y)$ = Total shortest paths from x to z containing node y in between
σ_{xz} = Total shortest paths from x to z

Calculate the betweenness centrality of all the nodes and find the node with highest betweenness centrality



Betweenness Centrality

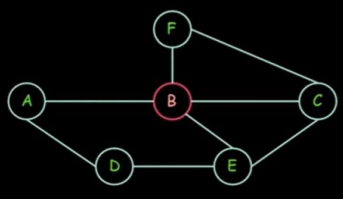
Node A



$$BC(y) = \sum_{x<z} \frac{\sigma_{xz}(y)}{\sigma_{xz}}$$

Paths	$\sigma_{xz}(y)$	σ_{xz}	$\sigma_{xz}(y)$ / σ_{xz}
(B,C)	0	1	0
(B,D)	1	2	0.5
(B,E)	0	1	0
(B,F)	0	1	0
(C,D)	0	1	0
(C,E)	0	1	0
(C,F)	0	1	0
(D,E)	0	1	0
(D,F)	1	3	0.3333
(E,F)	0	2	0
Total			0.8333

Node B

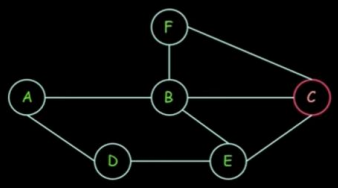


$$BC(y) = \sum_{x<z} \frac{\sigma_{xz}(y)}{\sigma_{xz}}$$

Paths	$\sigma_{xz}(y)$	σ_{xz}	$\sigma_{xz}(y)$ / σ_{xz}
(A,C)	1	1	1
(A,D)	0	1	0
(A,E)	1	2	0.5
(A,F)	1	1	1
(C,D)	0	1	0
(C,E)	0	1	0
(C,F)	0	1	0
(D,E)	0	1	0
(D,F)	2	3	0.6667
(E,F)	1	2	0.5
Total			3.6667

Betweenness Centrality

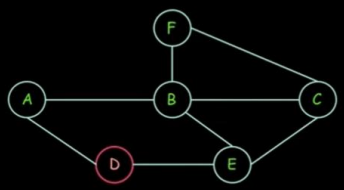
Node C



$$BC(y) = \sum_{x<z} \frac{\sigma_{xz}(y)}{\sigma_{xz}}$$

Paths	$\sigma_{xz}(y)$	σ_{xz}	$\sigma_{xz}(y)$ / σ_{xz}
(A,B)	0	1	0
(A,D)	0	1	0
(A,E)	0	2	0
(A,F)	0	1	0
(B,D)	0	2	0
(B,E)	0	1	0
(B,F)	0	1	0
(D,E)	0	1	0
(D,F)	1	3	0.3333
(E,F)	1	2	0.5
Total			0.8333

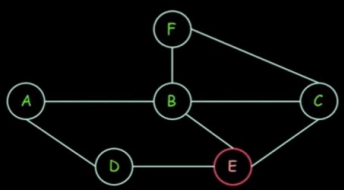
Node D



$$BC(y) = \sum_{x<z} \frac{\sigma_{xz}(y)}{\sigma_{xz}}$$

Paths	$\sigma_{xz}(y)$	σ_{xz}	$\sigma_{xz}(y)$ / σ_{xz}
(A,B)	0	1	0
(A,C)	0	1	0
(A,E)	1	2	0.5
(A,F)	0	1	0
(B,C)	0	1	0
(B,E)	0	1	0
(B,F)	0	1	0
(C,E)	0	1	0
(C,F)	0	1	0
(E,F)	0	2	0
Total			0.5

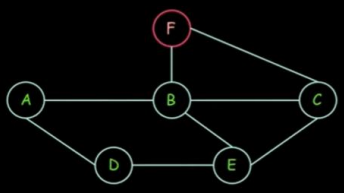
Node E



$$BC(y) = \sum_{x<z} \frac{\sigma_{xz}(y)}{\sigma_{xz}}$$

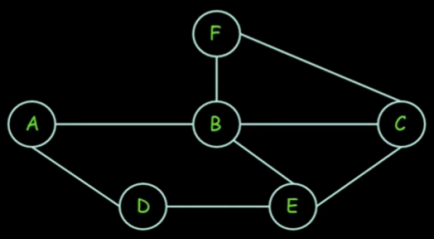
Paths	$\sigma_{xz}(y)$	σ_{xz}	$\sigma_{xz}(y)$ / σ_{xz}
(A,B)	0	1	0
(A,C)	0	1	0
(A,D)	0	1	0
(A,F)	0	1	0
(B,C)	0	1	0
(B,D)	1	2	0.5
(B,F)	0	1	0
(C,D)	1	1	1
(C,F)	0	1	0
(D,F)	2	3	0.6667
Total			2.1667

Node F



$$BC(y) = \sum_{x<z} \frac{\sigma_{xz}(y)}{\sigma_{xz}}$$

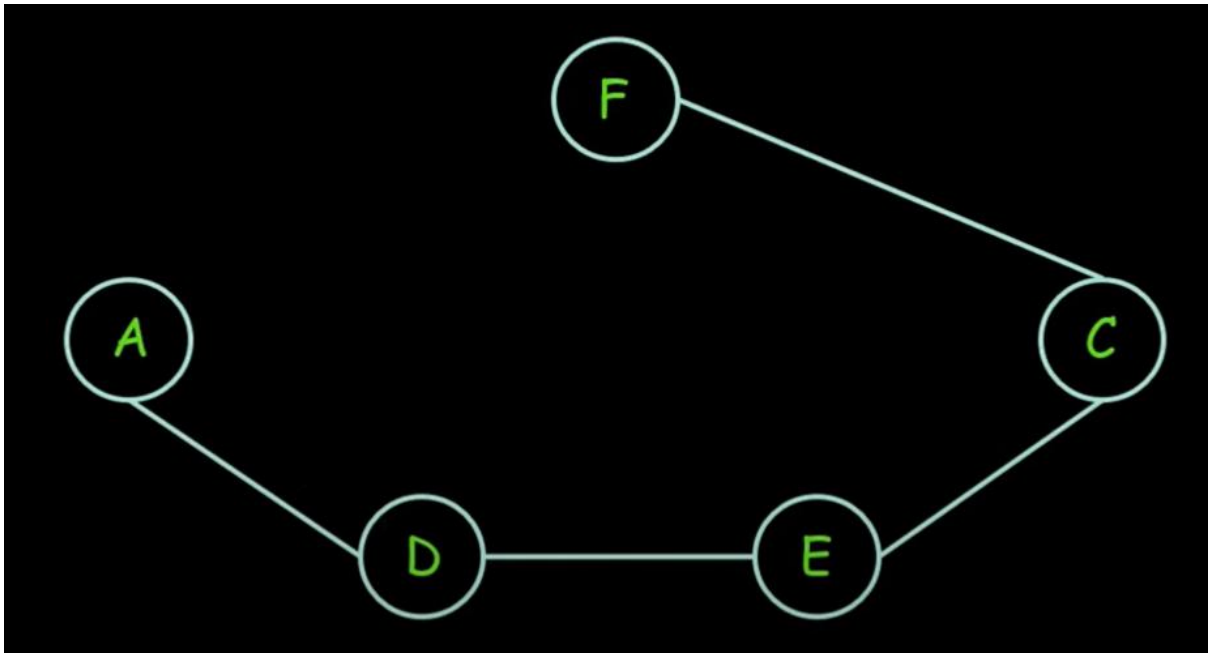
Paths	$\sigma_{xz}(y)$	σ_{xz}	$\sigma_{xz}(y)$ / σ_{xz}
(A,B)	0	1	0
(A,C)	0	1	0
(A,D)	0	1	0
(A,E)	0	2	0
(B,C)	0	1	0
(B,D)	0	2	0
(B,E)	0	1	0
(C,D)	0	1	0
(C,E)	0	1	0
(D,E)	0	1	0
Total			0



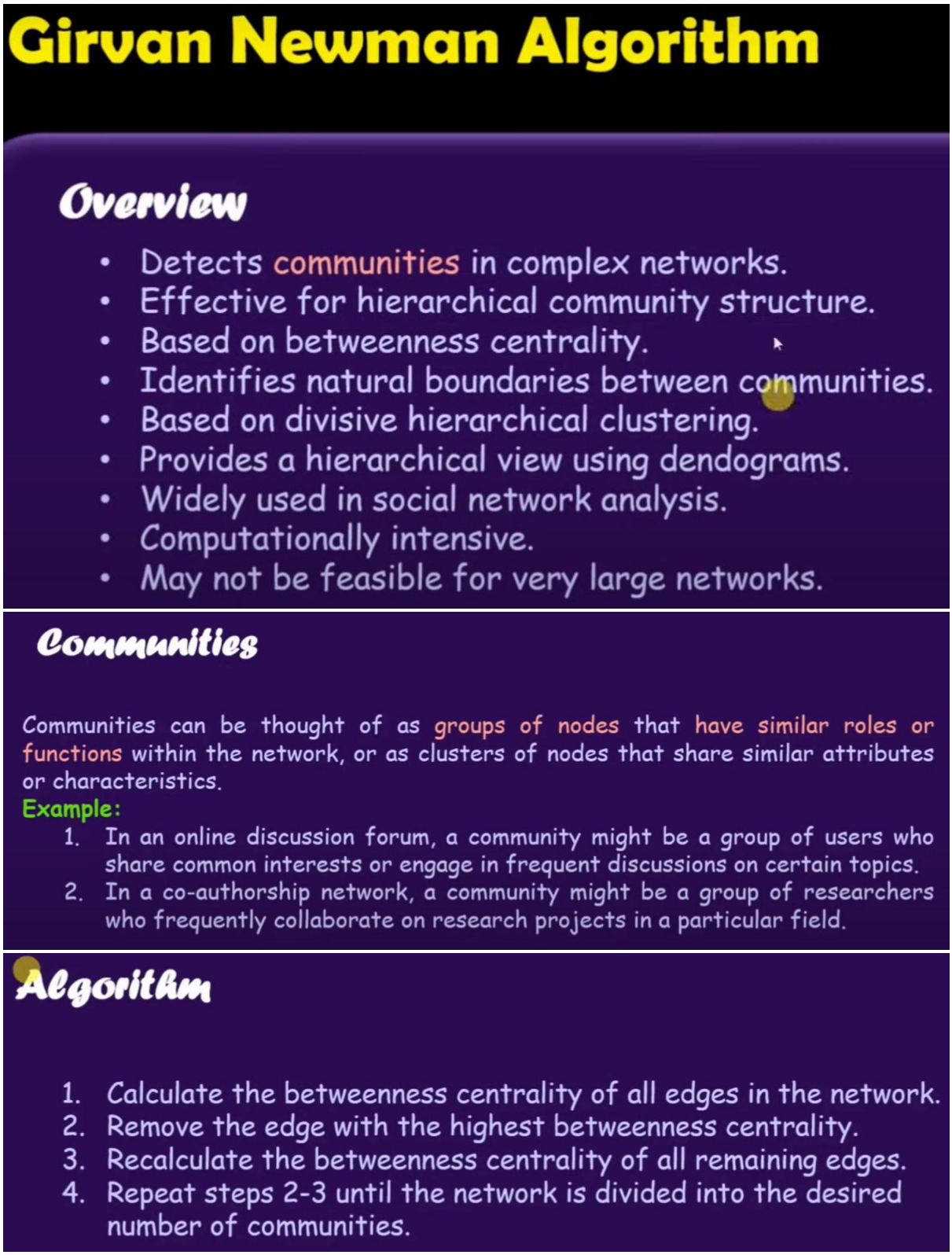
$$BC(y) = \sum_{x<z} \frac{\sigma_{xz}(y)}{\sigma_{xz}}$$

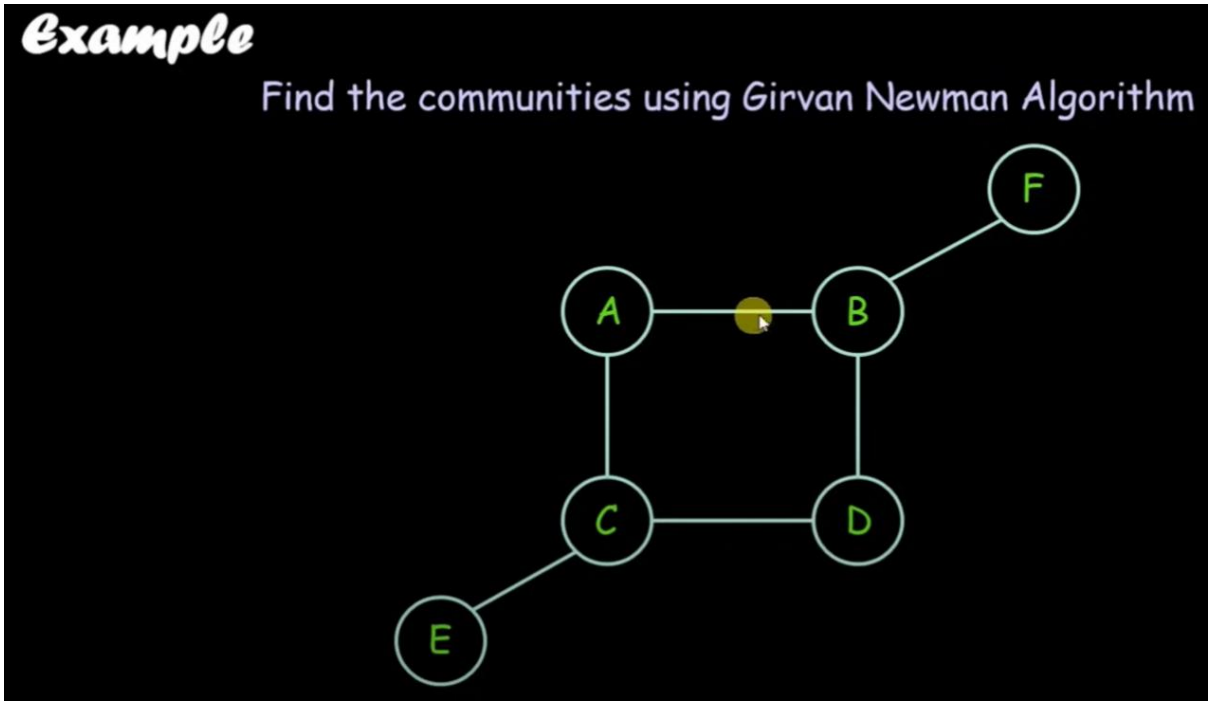
Nodes	Betweenness Centrality
A	0.8333
B	3.6667
C	0.8333
D	0.5
E	2.1667
F	0

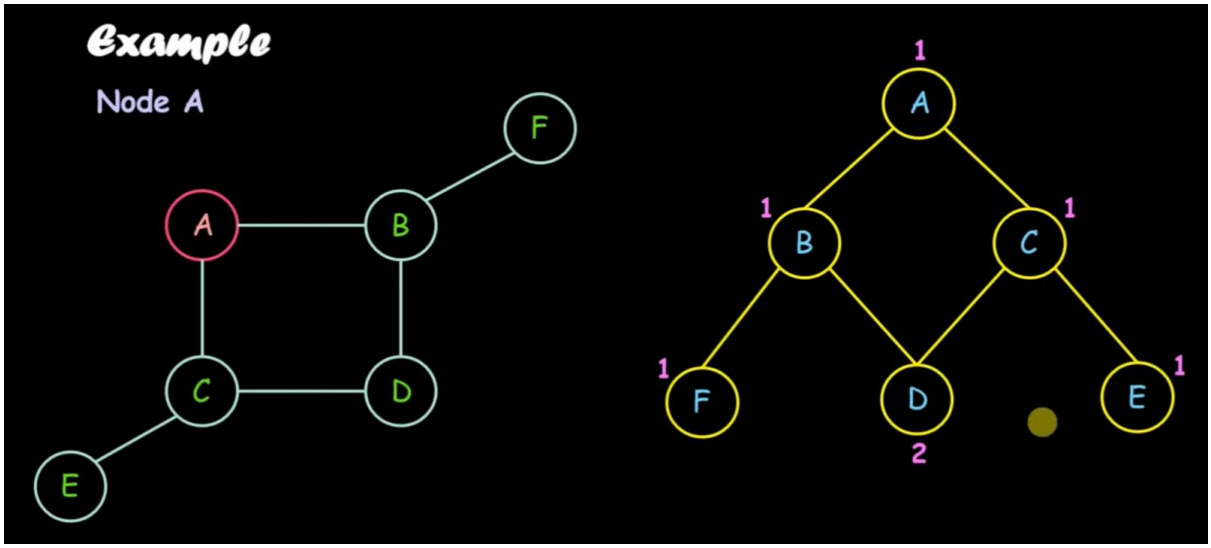
Node B has the highest betweenness centrality



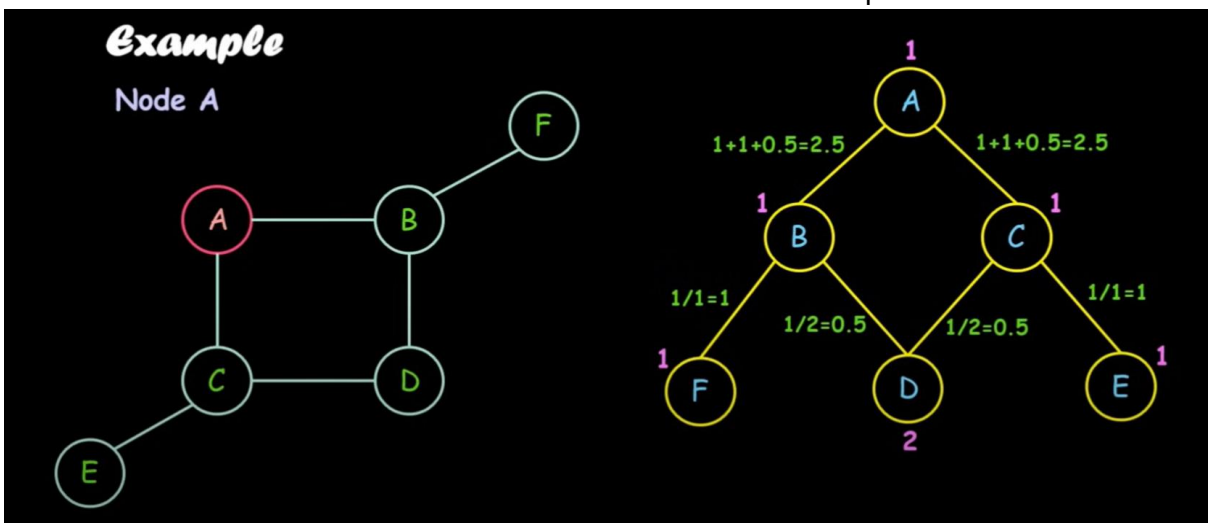
Girvan Newman Algorithm | Finding Communities | Mining Social Network Graph







The numbers written above each node are the number of shortest paths to reach that node.



To calculate the betweenness centrality start from the leaf node F.

For Edges attached to leaf node:

Edge wt. = Parent node wt. / Leaf node wt.

For e.g.:

Edge wt. of B-F = (wt. of B / wt. of F)

= 1/1 = 1

For Edges not attached to leaf nodes and having only node wt. as 1:

Edge wt. = (Edge wt.s of all edges of the child node) + (wt. of parent node/ wt. of child node)

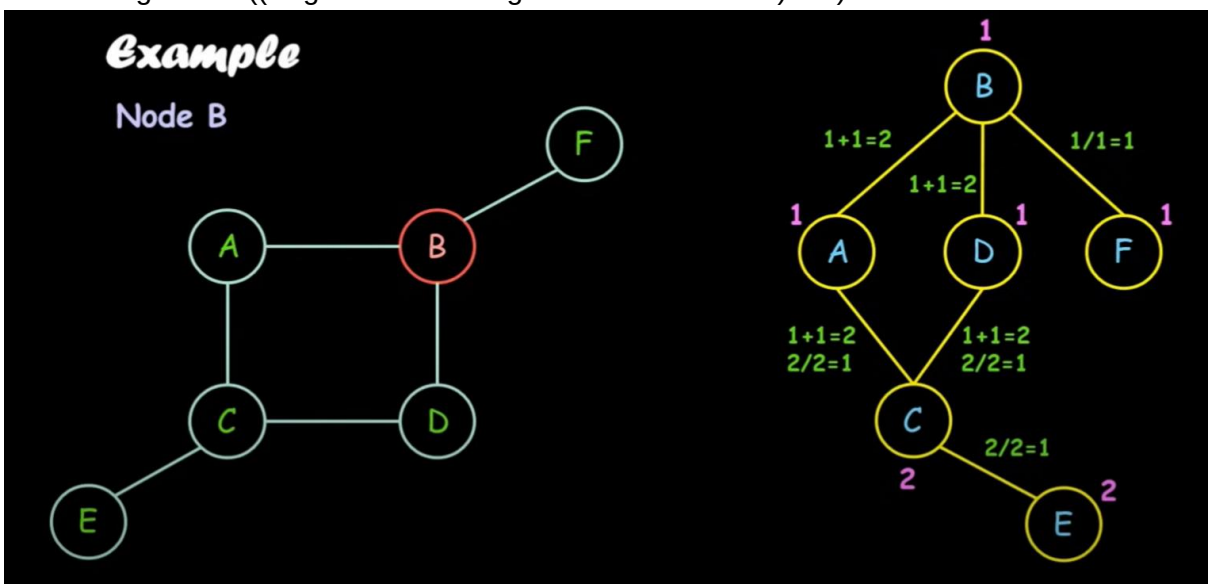
For e.g.:

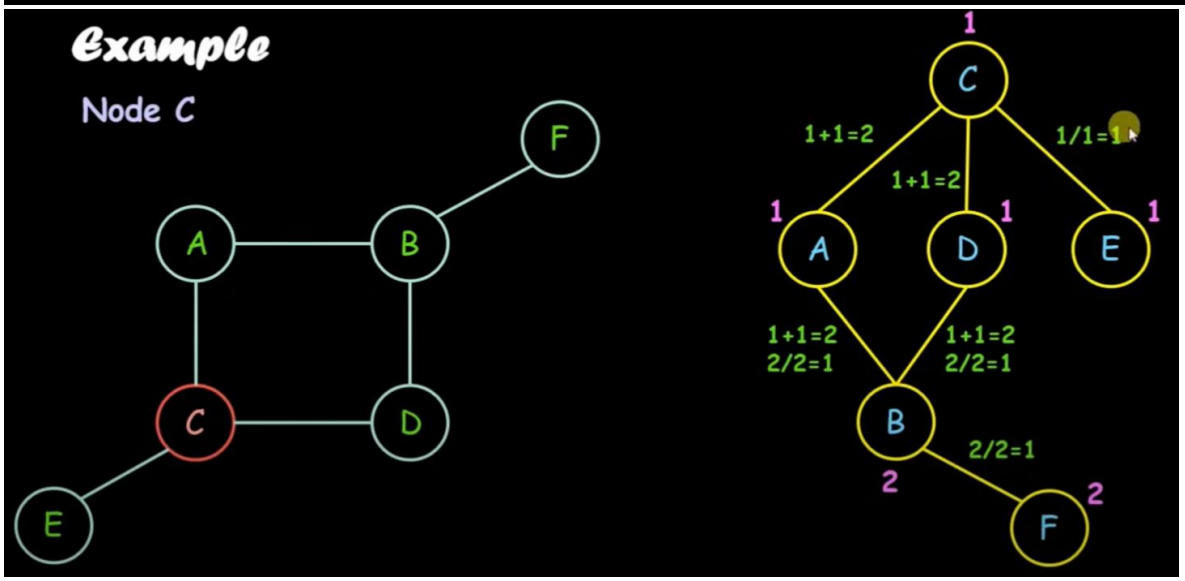
Edge wt. of A-B = edge wt. of B-F + edge wt. of B-D + (wt. of A/ wt. of B)

Edge wt. of A-B = 1 + 0.5 + 1/1 = 2.5

For Edges not attached to leaf nodes and having only node wt. > 1 (let's say n):

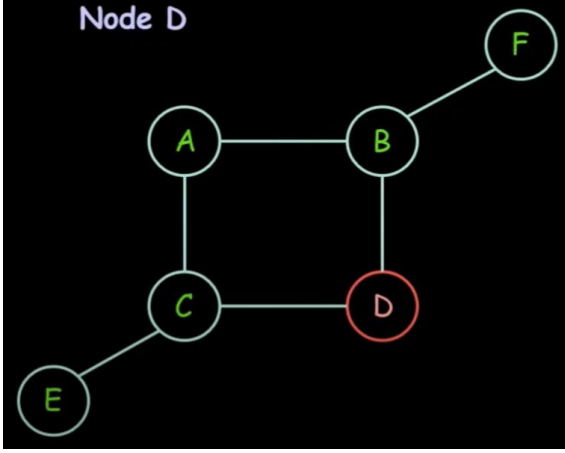
Edge wt. = ((Edge wt.s of all edges of the child node) + 1) / n

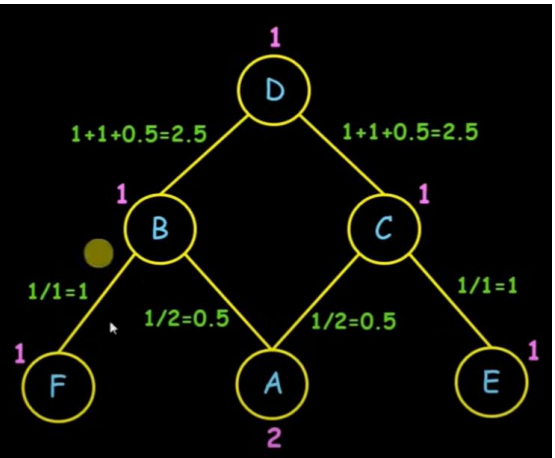




Example

Node D

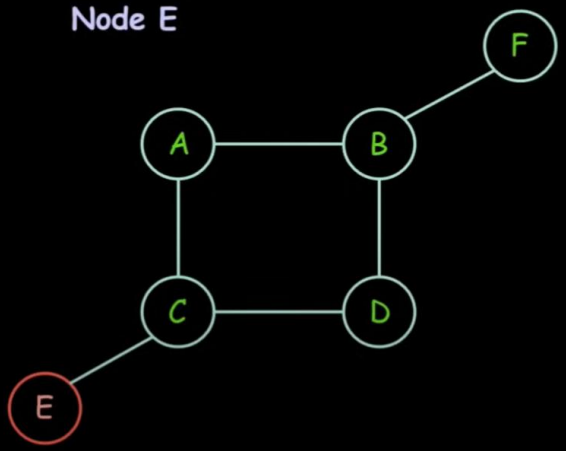


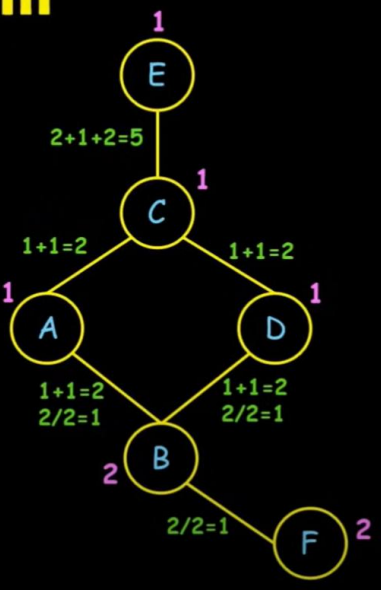


Girvan Newman Algorithm

Example

Node E

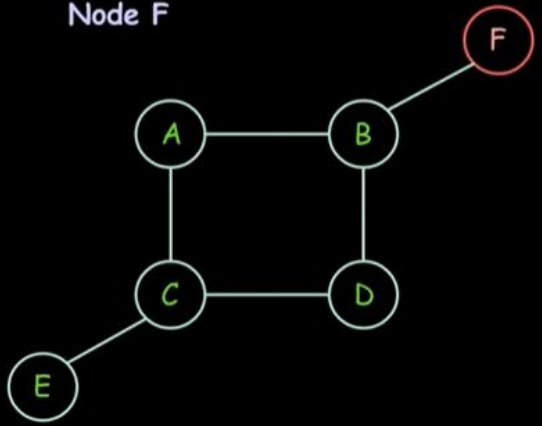


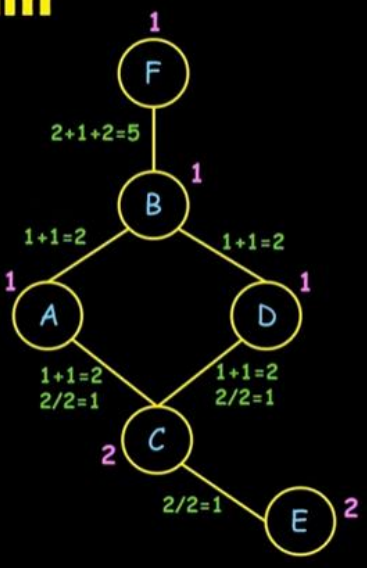


Girvan Newman Algorithm

Example

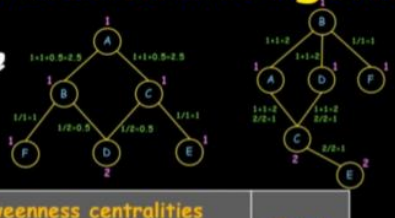
Node F



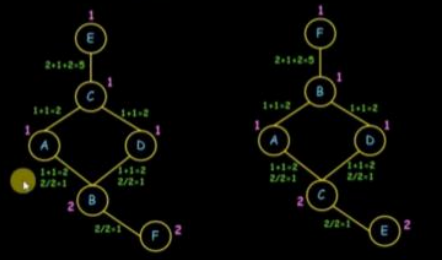


Girvan Newman Algorithm

Example

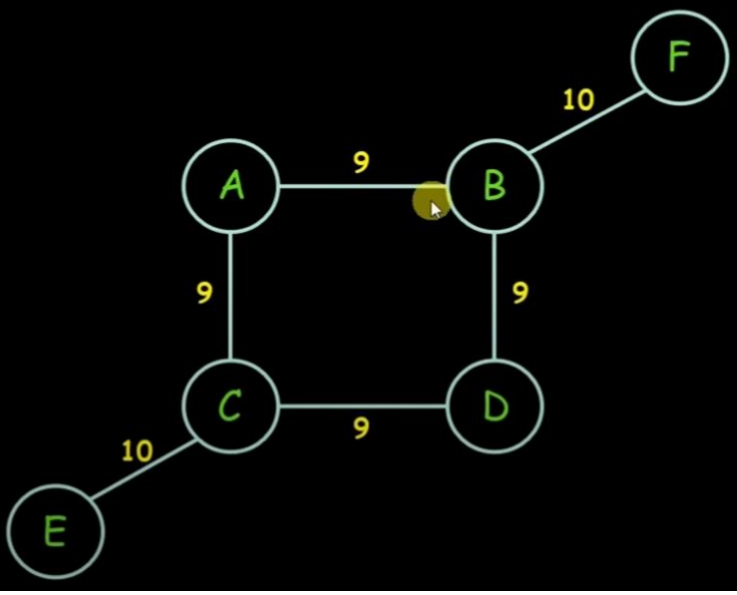


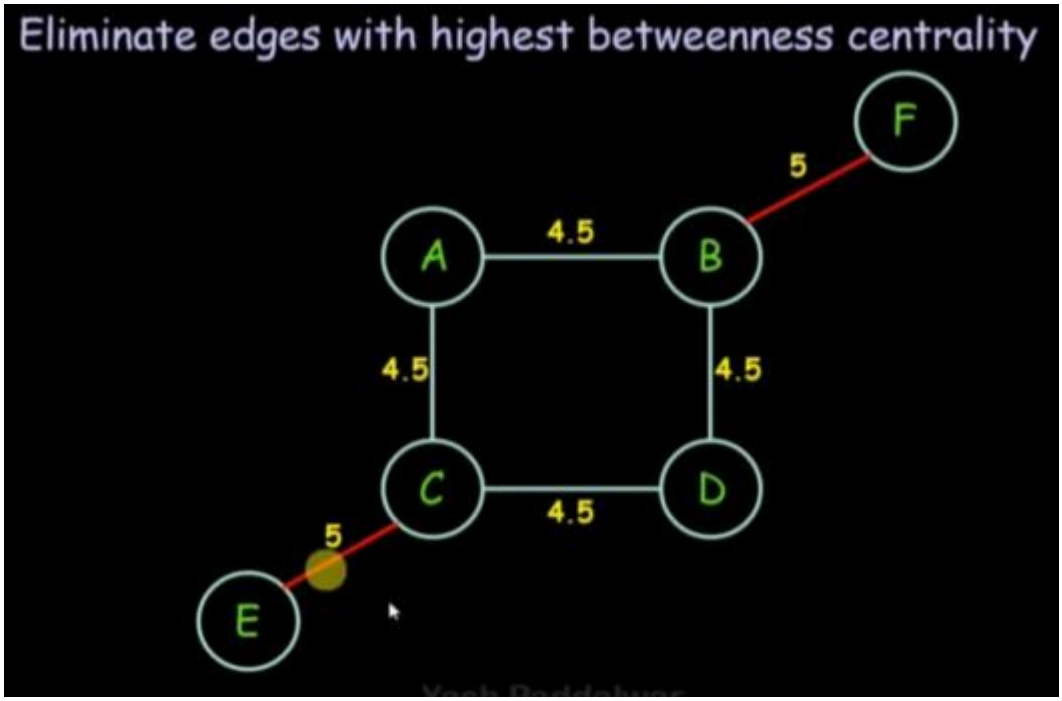
Edges	Betweenness centralities						Total
	A	B	C	D	E	F	
AB	2.5	2	1	0.5	1	2	9
AC	2.5	1	2	0.5	2	1	9
BD	0.5	2	1	2.5	1	2	9
BF	1	1	1	1	1	5	10
CD	0.5	1	2	2.5	2	1	9
CE	1	1	1	1	5	1	10

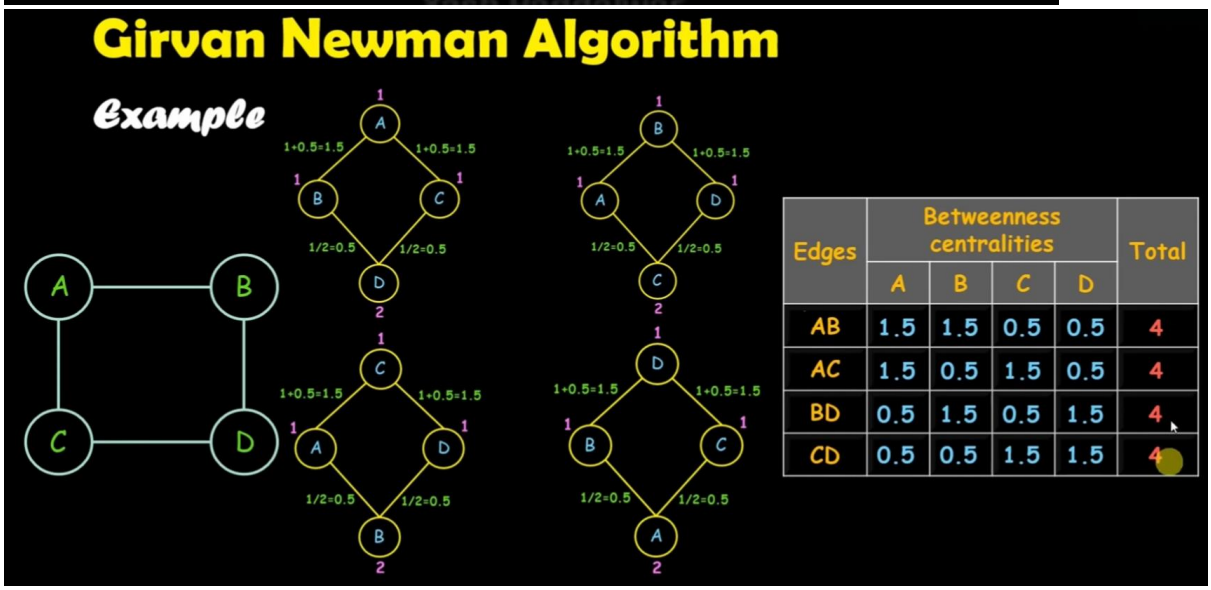


Yash Paddalwar

Since, edges are bi-directional, betweenness centrality of each edge will be reduced by half

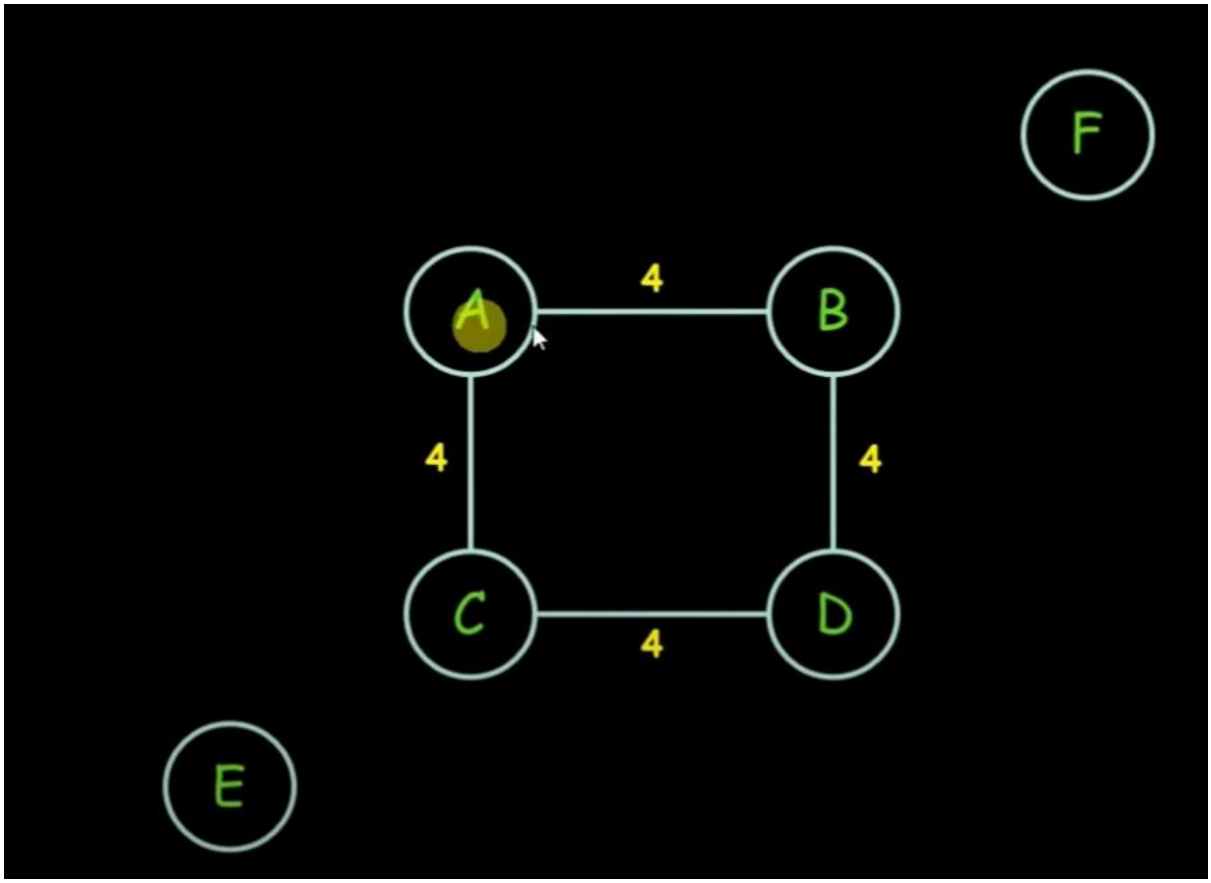


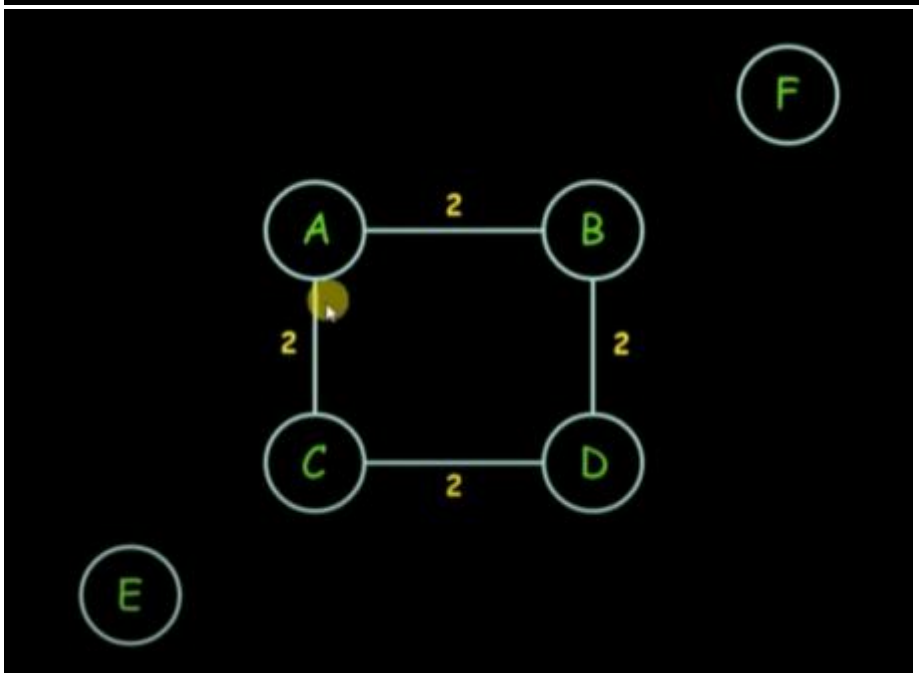


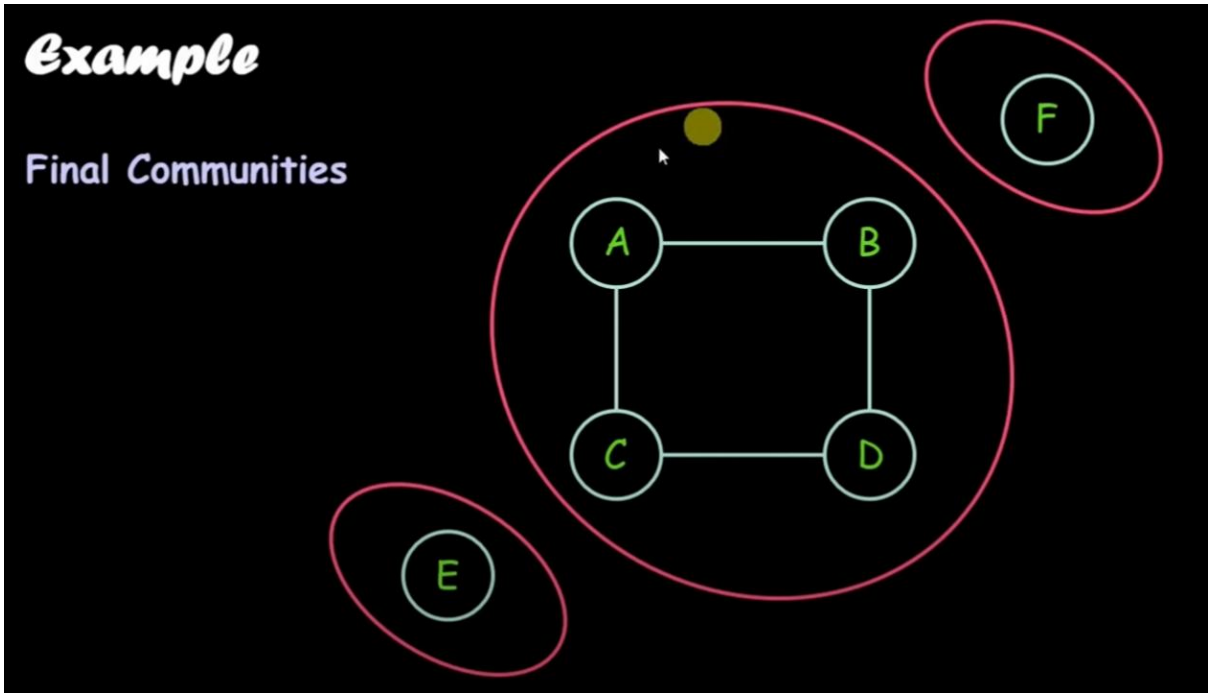


Edges	Betweenness centralities				Total
	A	B	C	D	
AB	1.5	1.5	0.5	0.5	4
AC	1.5	0.5	1.5	0.5	4
BD	0.5	1.5	0.5	1.5	4
CD	0.5	0.5	1.5	1.5	4

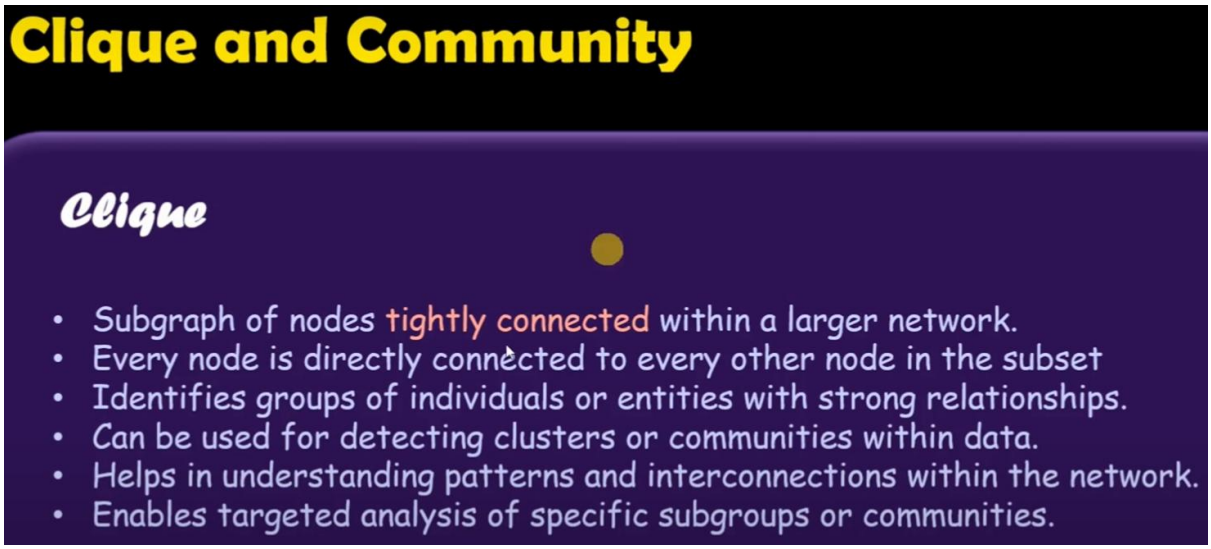
Same steps as above

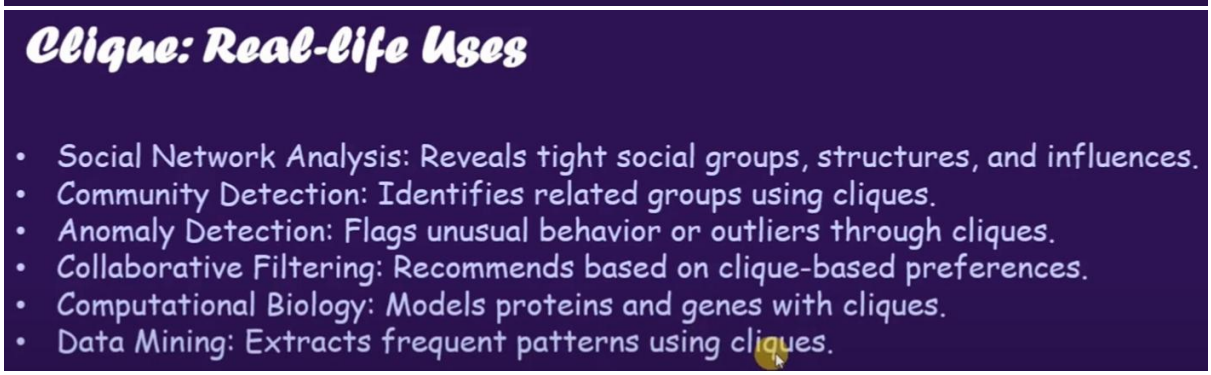

F
A
4
B
4
4
C
4
D
E

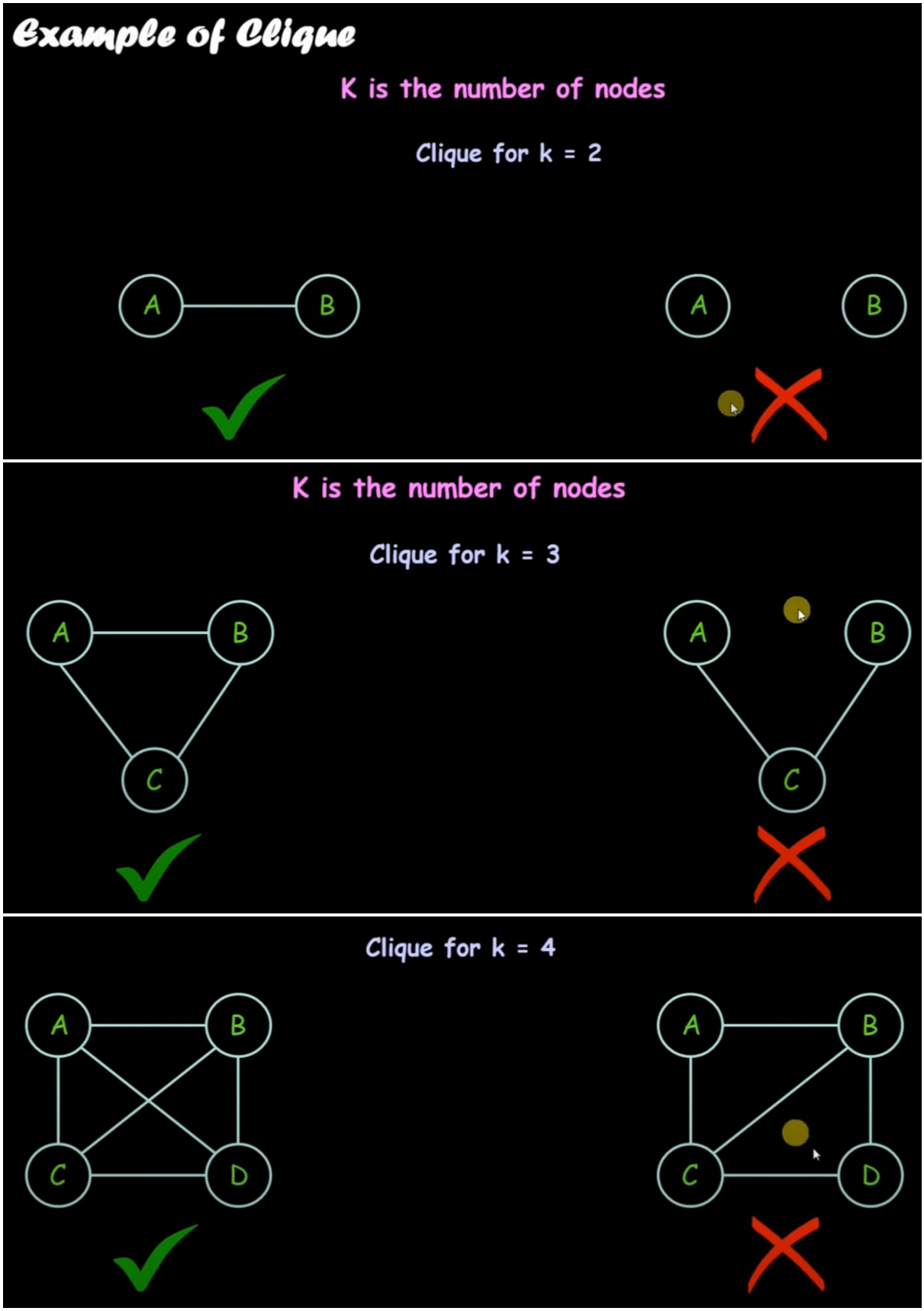

F
A
2
B
2
2
C
2
D
E



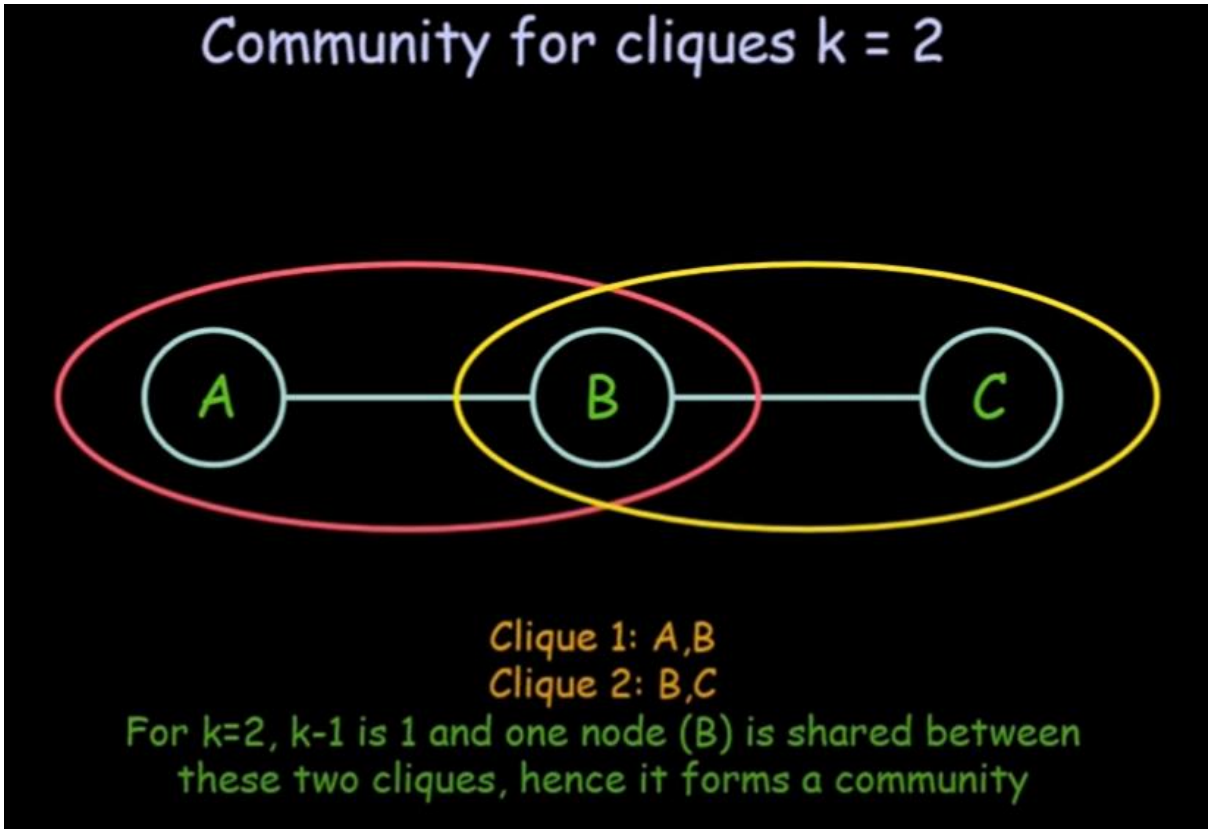
Clique and Community | Clique Percolation Method



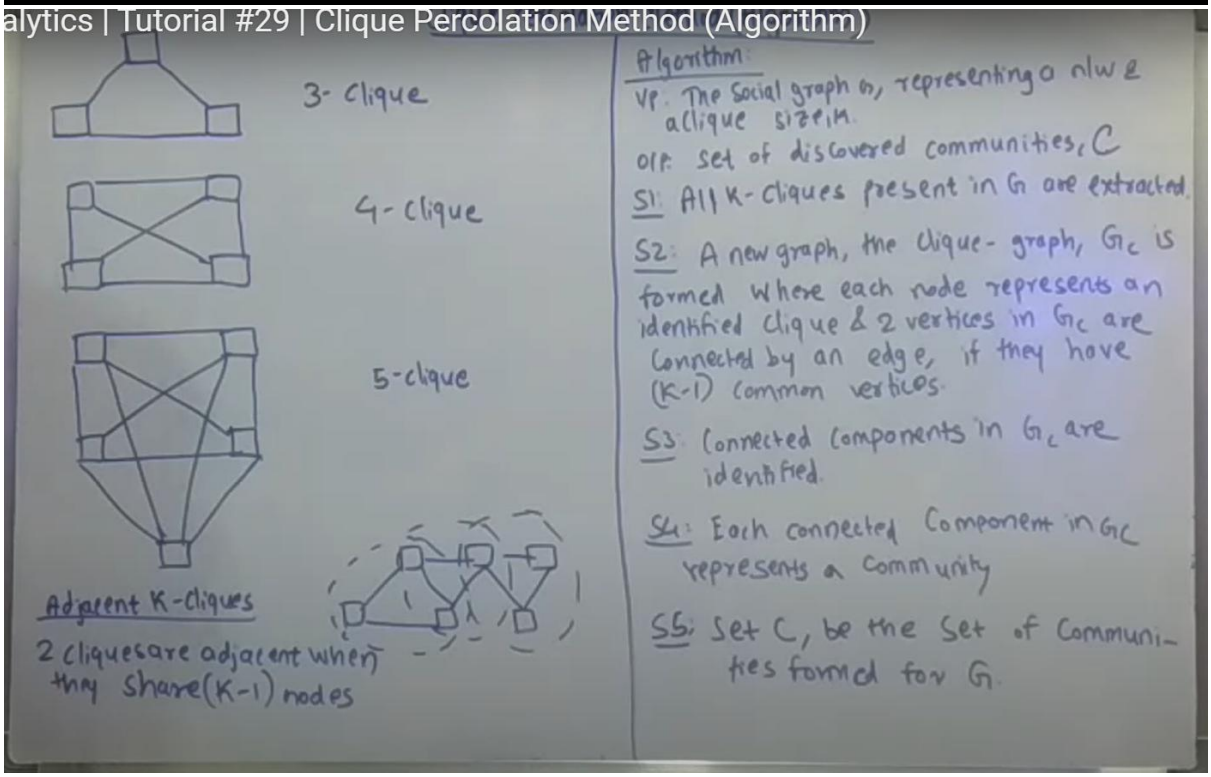




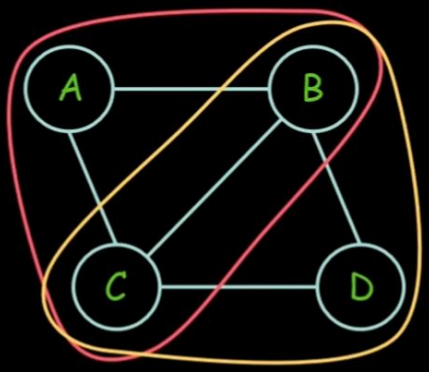
Community - Group of cliques that share k-1 nodes in common



alytics | Tutorial #29 | Clique Percolation Method (Algorithm)



Community for cliques k = 3



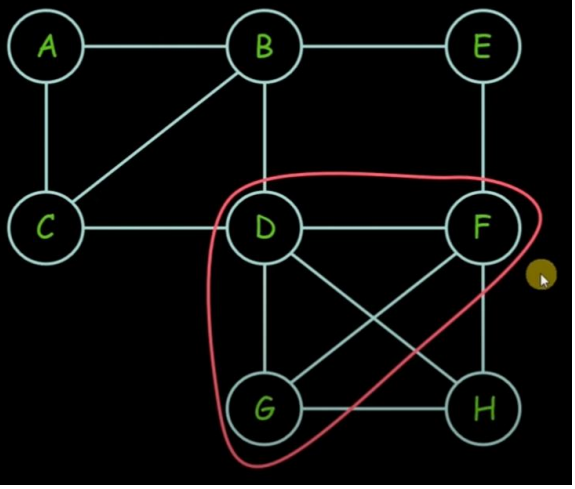
Clique 1: A,B,C
Clique 2: B,C,D
For k=3, k-1 is 2 and two nodes (B and C) are shared between these two cliques, hence it forms a community

Find the cliques and communities for k=3 and k=4

For k=3

Cliques are as follows:

- Clique 1: A,B,C
- Clique 2: B,C,D
- Clique 3: D,G,H
- Clique 4: F,G,H
- Clique 5: D,F,H
- Clique 6: D,F,G

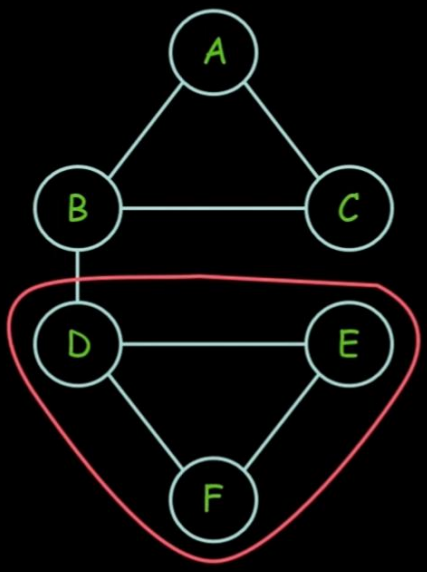


Find the cliques and communities for k=3

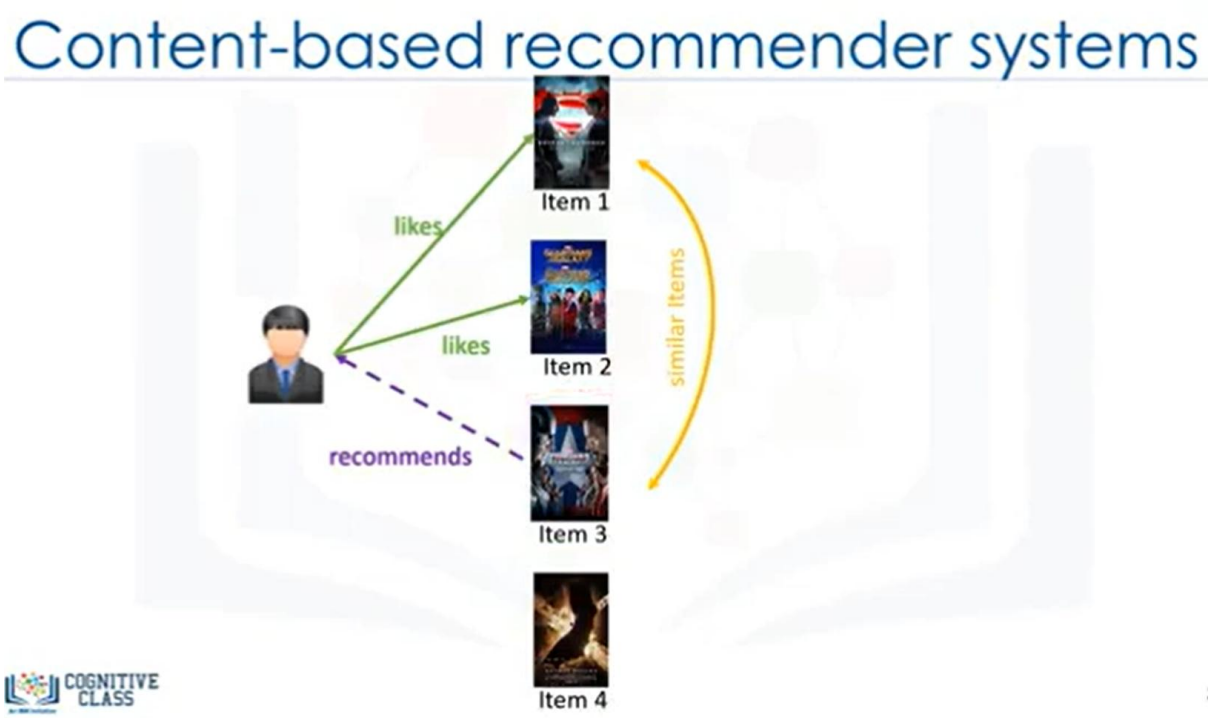
Cliques are as follows:

- Clique 1: A,B,C
- Clique 2: D,E,F

No Communities



Content based Recommendation System



Weighing the genres

	2
	10
	8

Input User Ratings

	Comedy	Adventure	Super Hero	Sci-Fi
	0	1	1	0
	1	1	1	1
	1	0	1	0

Movies Matrix

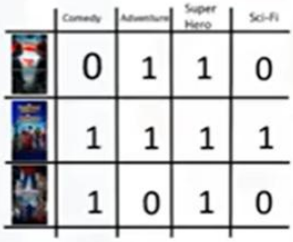
COGNITIVE CLASS

4

Weighing the genres

Input User Ratings

X



Movies Matrix

=

Weighted Genre Matrix

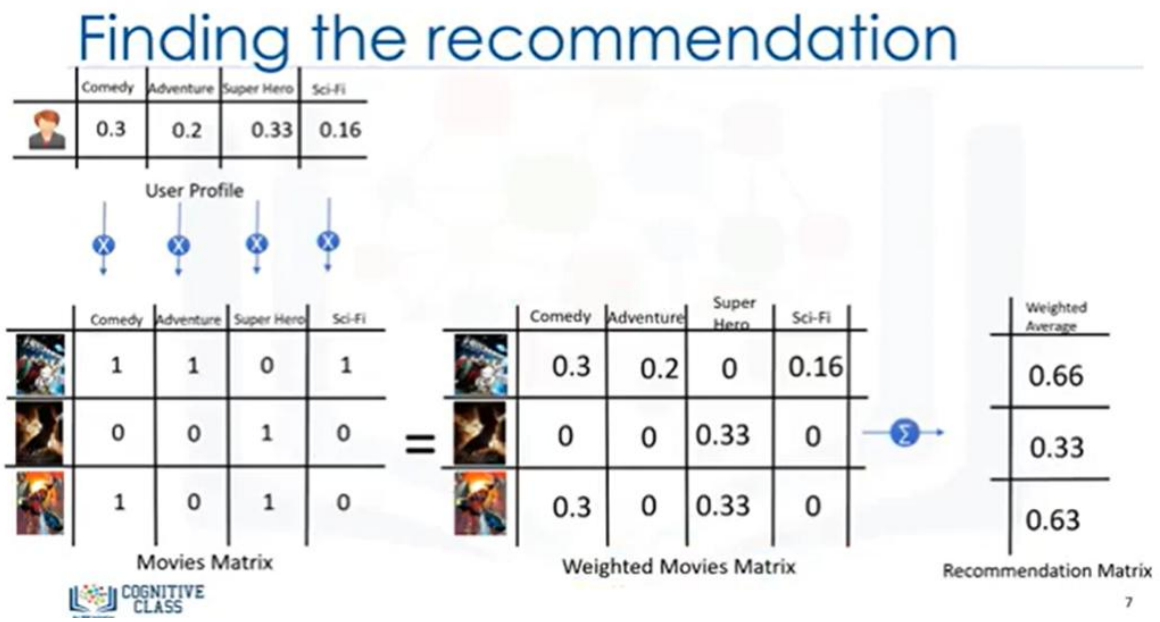
	Comedy	Adventure	Super Hero	Sci-Fi
	0	2	2	0
	10	10	10	10
	8	0	8	0

	Comedy	Adventure	Super Hero	Sci-Fi
	0.3	0.2	0.33	0.16

User Profile

COGNITIVE CLASS

5



What is Content based filtering..?

- **Content-based Filtering** is a Machine Learning technique that uses similarities in features to make decisions.
- This technique is often used in recommender systems, which are algorithms designed to advertise or recommend things to users based on knowledge accumulated about the user.

Methodology

- Comparing user interests to product features.

- Most overlapping features with user interests are what's recommended.

Two Method Followed:

- Firstly, users can be given a list of features out of which they can choose whatever they identify with the most.

- Secondly, the algorithm can keep track of the products the user has chosen before and add those features to the users' data.

Numerical Example:

	Feature 1	Feature 2	Feature 3	Feature 4
Product 1	1		1	1
Product 2		1	4	
Product 3	3			1
User Interest	2		1	1

$$User\ Interest\ Level\ = \sum_{i=1}^{d} p_i u_i$$

where
p_i is the product feature value and u_i user interest value in column i

Numerical Example:

	Feature 1	Feature 2	Feature 3	Feature 4
Product 1	1		1	1
Product 2		1	4	
Product 3	3			1
User Interest	2		1	1

$$User\ Interest\ Level\ Product\ 1 = 2 * 1 + 1 * 1 + 1 * 2 = 5$$

$$User\ Interest\ Level\ Product\ 2 = 1 * 4 = 4$$

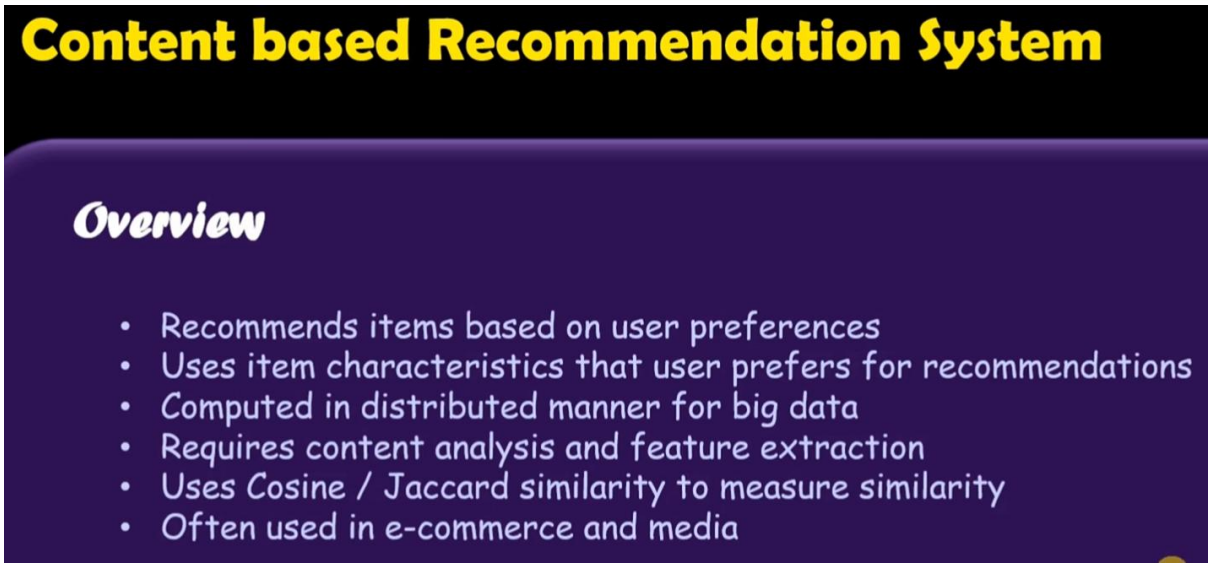
$$User\ Interest\ Level\ Product\ 3 = 2 * 3 + 1 * 1 = 7$$

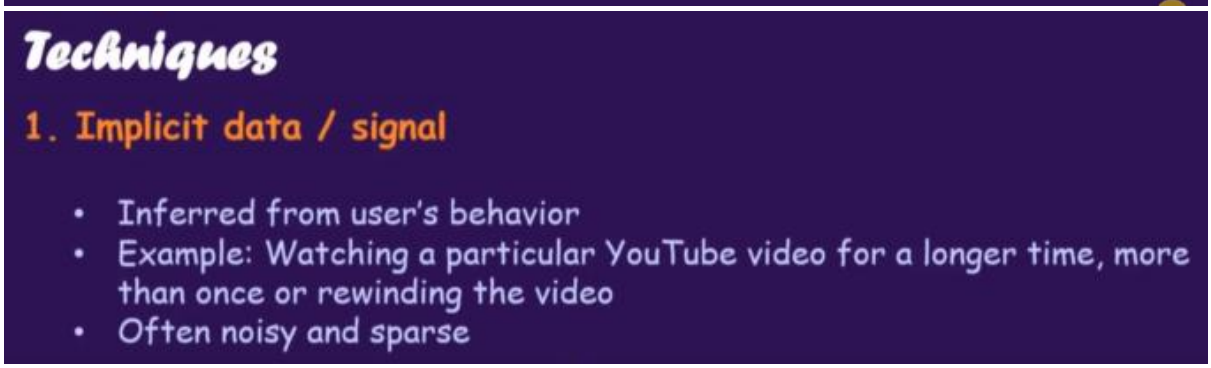
Pros:

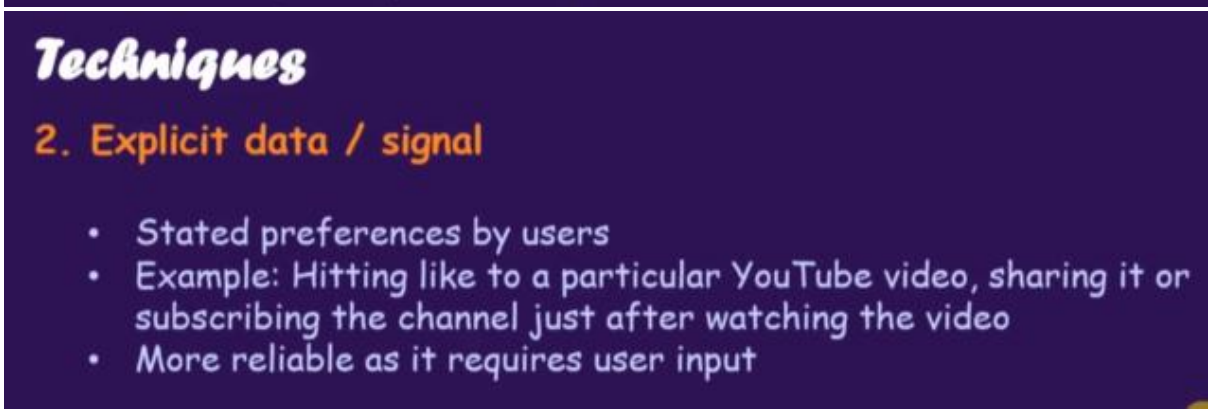
- Easily scalable due to low amounts of data.
- Unlike other models, this does not need to compare with other users' data.

Cons:

- Model requires a fair amount of domain knowledge.
- Content-based filtering depends greatly on previously known user interests.



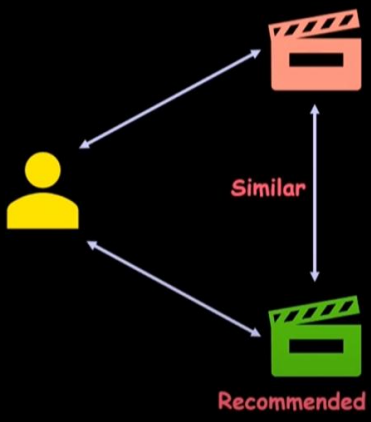




Content based Recommendation System

Example

User Vector information:
Relevant keywords / features from the watched videos in the form of a vector/array containing importance scores



Item1 Vector information:
Matrix Multiplication using MapReduce
Features / Keywords:
- Matrix Multiplication
- MapReduce
- Big Data Analytics
- At A Glance!

Stores this in the form of array/vector with importance score

Item2 Vector information:
MapReduce in Hadoop
Features / Keywords:
- MapReduce
- Hadoop
- Big Data Analytics
- At A Glance!

Stores this in the form of array/vector with importance score

Similarity Measures

1. Cosine Similarity

$$\text{cosine}(x, y) = \frac{\sum_{i=1}^{n} x_i y_i}{\sum_{i=1}^{n} x_i^2 \; * \; \sum_{i=1}^{n} y_i^2}$$

where, xi and yi are the vectors

- Best suited when high dimensional features are present
- Suitable for continuous or ordinal features
- Insensitive to the scale and magnitude of features
- Captures the direction and magnitude of similarity
- Range of similarity values from -1 to 1

Similarity Measures

2. Jaccard Similarity

$$Jaccard(x, y) = \frac{|x \cap y|}{|x \cup y|}$$

where,

$$|x \cup y| = |x| + |y| - |x \cap y|$$

$|x \cap y|$ = Common scores in both vectors
$|x|$ = No. of scores in vector x
$|y|$ = No. of scores in vector y

- Not much effective for content based recommendation systems
- Suitable for sparse and imbalanced data
- Focuses on the presence or absence of interactions
- Ignores the magnitude or value of interactions
- Range of similarity values from 0 to 1

Advantages

- Does not rely on historical user-item interactions
- Can make recommendations for new or unpopular items
- Only user's preferences are considered

Disadvantages

- Difficult to provide recommendations for new users with no historical interaction
- Cannot account for social influence or context
- Content description needed

Collaborative Filtering

User-User Collaborative Filtering

https://www.geeksforgeeks.org/user-based-collaborative-filtering/

Collaborative Filtering
Version 1: "User-User" Collaborative Filtering

Consider user ***x***
and unrated item ***i***

Find set ***N*** of other users whose ratings are "**similar**" to ***x***'s ratings

Estimate ***x***'s ratings for ***i*** based on ratings for ***i*** of users in ***N***

Stanford

Collaborative filtering

Find similar users and recommend items that they like:

- Represent users by their rows in the **utility matrix**
- Two users are similar if their vectors are similar!

	Harry Potter			Twilight	Star Wars		
	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	4			5	1		
B	5	5	4				
C				2	4	5	
D		3					3

Finding Similar Users

Let r_x be the vector of user x's ratings

r_x = [*, _, _, *, ***]
r_y = [*, _, **, **, _]

r_x = {1, 0, 0, 1, 3}
r_y = {1, 0, 2, 2, 0}

Cosine similarity measure

- $\text{sim}(x, y) = \cos(r_x, r_y) = \frac{r_x \cdot r_y}{||r_x|| \, ||r_y||}$

Problem: This representation leads to unintuitive results

Problems with raw utility matrix cosine

	Harry Potter			Twilight	Star Wars		
	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	4			5	1		
B	5	5	4				
C				2	4	5	
D		3					3

Intuitively we want: sim(*A*, *B*) > sim(*A*, *C*)

$$\text{sim(A,B)} = \frac{4 \times 5}{\sqrt{4^2+5^2+1^2}\sqrt{5^2+5^2+4^2}} = 0.380$$

$$\text{sim(A,C)} = \frac{5 \times 2 + 1 \times 4}{\sqrt{4^2+5^2+1^2}\sqrt{2^2+4^2+5^2}} = 0.322$$

Yes, 0.380 > 0.322
But only barely works...

Problem with raw cosine

	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	4			5	1		
B	5	5	4				
C				2	4	5	
D		3					3

- Problem with cosine:
  - C really loves SW
  - A hates SW
  - B just hasn't seen it
- Another problem: we'd like to normalize the raters
  - D rated everything the same; not very useful

Mean-Centered Utility Matrix: subtract the means of each row

	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	4			5	1		
B	5	5	4				
C				2	4	5	
D		3					3

	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	$2/3$			$5/3$	$-7/3$		
B	$1/3$	$1/3$	$-2/3$				
C				$-5/3$	$1/3$	$4/3$	
D		0					0

- Now a 0 means no information
- And negative ratings means viewers with opposite ratings will have vectors in opposite directions!

Stan

Modified Utility Matrix: subtract the means of each row

	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	$2/3$			$5/3$	$-7/3$		
B	$1/3$	$1/3$	$-2/3$				
C				$-5/3$	$1/3$	$4/3$	
D		0					0

$$\text{Cos(A,B)} = \frac{(2/3) \times (1/3)}{\sqrt{(2/3)^2 + (5/3)^2 + (-7/3)^2}\sqrt{(1/3)^2 + (1/3)^2 + (-2/3)^2}} = 0.092$$

$$\text{Cos(A,C)} = \frac{(5/3) \times (-5/3) + (-7/3) \times (1/3)}{\sqrt{(2/3)^2 + (5/3)^2 + (-7/3)^2}\sqrt{(-5/3)^2 + (1/3)^2 + (4/3)^2}} = -0.559$$

Stanford

Now A and C are (correctly) way further apart than A,B

Finding similar users with overlapping-item mean-centering

Let r_x be the vector of user x's ratings

r_x = {1, 0, 0, 1, 3}
r_y = {1, 0, 2, 2, 0}

r_x = [*, _, _, *, ***]
r_y = [*, _, **, **, _]

Mean-centering:

- For each user x, let $\bar{r_x}$ be mean of r_x (ignoring missing values)
- $\bar{r_x}$ = (1 + 1 + 3)/3 = 5/3 $\bar{r_y}$ = (1 + 2 + 2)/3 = 5/3
- Subtract this average from each of their ratings
  - (but do nothing to the "missing values"; they stay "null").
  - mean centered r_x = {-2/3, 0, 0, -2/3, 4/3}

One new idea: Keep only items they both rate (unlike 2 slides ago)

r_x = {-2/3, [redacted] -2/3, [redacted]} r_y = {-2/3, [redacted], 1/3, [redacted]}

r_x = {-2/3, -2/3} r_y = {-2/3, 1/3}

Now take cosine:

- Now compute cosine between user vectors
- cos([-2/3, -2/3], [-2/3, 1/3])

Stanford

Mean-centered overlapping-item cosine similarity

Let r_x be the vector of user x's ratings,and $\bar{r_x}$ be its mean (ignoring missing values)

Instead of basic cosine similarity measure

- $sim(x, y) = cos(r_x, r_y) = \frac{r_x \cdot r_y}{||r_x||\ ||r_y||}$

Mean-centered overlapping-item cosine similarity

- S_{xy} = items rated by both users x and y

$$sim(x,y) = \frac{\sum_{s \in S_{xy}} (r_{xs} - \bar{r_x})(r_{ys} - \bar{r_y})}{\sqrt{\sum_{s \in S_{xy}} (r_{xs} - \bar{r_x})^2} \sqrt{\sum_{s \in S_{xy}} (r_{ys} - \bar{r_y})^2}}$$

Stanford

Rating Predictions

From similarity metric to recommendations for an unrated item *i*:

Let r_x be the vector of user ***x***'s ratings

Let ***N*** be the set of ***k*** users most similar to ***x*** who have rated item ***i***

Prediction for item *i* of user *x*:

- Rate i as the mean of what k-people-like-me rated i

$$r_{xi} = \frac{1}{k} \sum_{y \in N} r_{yi}$$

- Even better: Rate i as the mean weighted by their similarity to me ...

$$r_{xi} = \frac{\sum_{y \in N} s_{xy} \, r_{yi}}{\sum_{y \in N} s_{xy}}$$

Shorthand:

$$s_{xy} = sim(x, y)$$

Stanford

Item-Item Collaborative Filtering

https://www.geeksforgeeks.org/item-to-item-based-collaborative-filtering/

Collaborative Filtering Version 2: Item-Item Collaborative Filtering

So far: User-user collaborative filtering

Alternate view that often works better: Item-item

- For item ***i***, find other similar items
- Estimate rating for item ***i*** based on ratings for those similar items
- Can use same similarity metrics and prediction functions as in user-user model
- "Rate i as the mean of my ratings for other items, weighted by their similarity to i"

$$r_{xi} = \frac{\sum_{j \in N(i;x)} s_{ij} \, r_{xj}}{\sum_{j \in N(i;x)} s_{ij}}$$

N(i;x)...set of items rated by ***x*** *and* similar to ***i***

s_{ij}... similarity of items ***i*** and ***j***

r_{xj}...rating of user ***x*** on item ***j***

Stanford

Item-Item CF (|N|=2)

users

movies	1	2	3	4	5	6	7	8	9	10	11	12	sim(1,m)
1	1		3		?	5			5		4		1.00
2			5	4			4			2	1	3	..
3	2	4		1	2		3		4	3	5		?
4		2	4		5			4			2		..
5			4	3	4	2					2	5	..
6	1		3		3			2			4		

Neighbor selection:
Identify movies similar to movie **1**, **rated by user 5**

Here we use mean centered item-overlap cosine as similarity:
1) Subtract mean rating m_i from each movie i between rows
2) Compute (item-overlapping) cosine similarities

Subtract mean rating m_i from each movie i

$m_1 = (1+3+5+5+4)/5 = 18/5$

	1	2	3	4	5	6	7	8	9	10	11	12
1	-13/5		-3/5		?	7/5			7/5		2/5	
3	-1	1		-2	-1		0		1	0	2	
6	-8/5		2/5		2/5			-3/5			7/5	

Item-Item CF (|N|=2)

users

movies	1	2	3	4	5	6	7	8	9	10	11	12	sim(1,m)
1	-13/5		-3/5		?	7/5			7/5		2/5		1.00
2			5	4			4			2	1	3	..
3	-1	1		-2	-1		0		1	0	2		?
4		2	4		5			4			2		..
5			4	3	4	2					2	5	..
6	-8/5		2/5		2/5			-3/5			7/5		?

Neighbor selection:
Identify movies similar to movie **1**, **rated by user 5**

Here we use mean centered item-overlap cosine as similarity:
1) Subtract mean rating m_i from each movie i
2) Compute (item-overlapping) cosine similarities between rows

Compute Cosine Similarity:

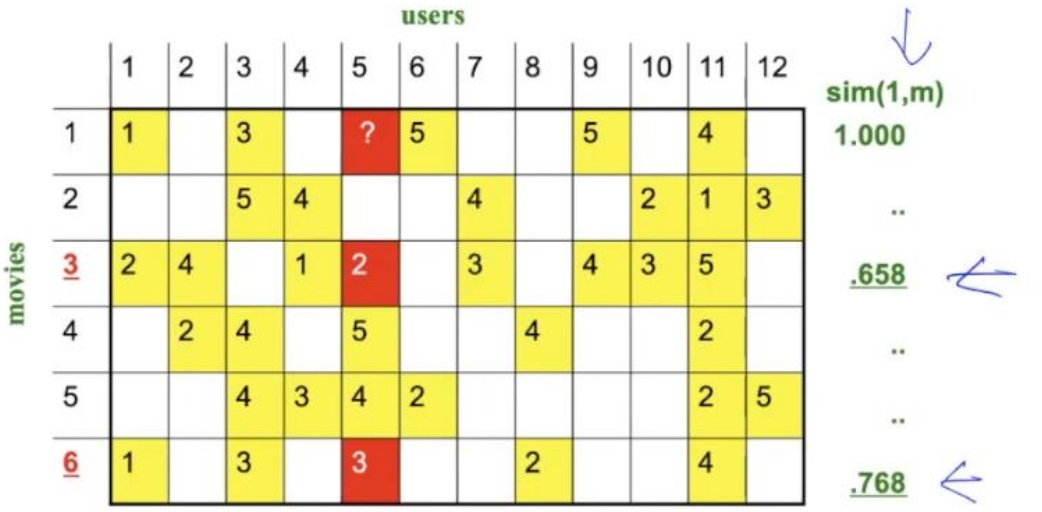
For rows 1 and 3, they both have values for users 1, 9 and 11.

$$\text{sim}(1,3) = \frac{(-13/5)(-1)+(7/5)(1)+(2/5)(2)}{\sqrt{(-13/5)^2+(7/5)^2+(2/5)^2}\sqrt{(-1)^2+(1)^2+(2)^2}} \approx 0.658$$

For rows 1 and 6, they both have values for users 1, 3 and 11.

$$\text{sim}(1,6) = \frac{(-13/5)(-8/5)+(-3/5)(2/5)+(2/5)(7/5)}{\sqrt{(-13/5)^2+(-3/5)^2+(2/5)^2}\sqrt{(-8/5)^2+(2/5)^2+(7/5)^2}} \approx 0.768$$

Item-Item CF (|N|=2)



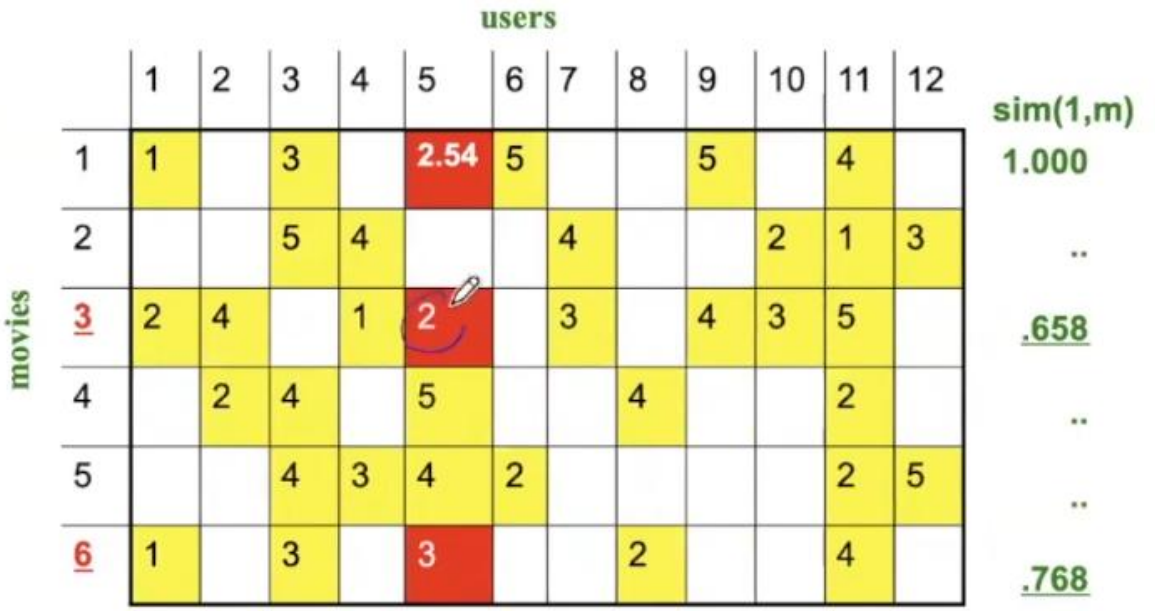
Compute similarity weights:

$s_{1,3}$=.658, $s_{1,6}$=.768 (we compute $s_{1,2}$, $s_{1,4}$, $s_{1,5}$ too; let's assume those are smaller)

Stanford

Slides adapted from Jure Leskovec, CS246 and J. Leskovec, A. Rajaraman, J. Ullman: *Mining of Massive Datasets*

Item-Item CF (|N|=2)
Approximate rating with weighted mean



Predict by taking weighted average:

$$r_{ix} = \frac{\sum_{j \in N(i;x)} s_{ij} r_{jx}}{\sum s_{ij}}$$

$r_{1,5}$ = (0.658*2 + 0.768*3) / (0.658+0.768) = 2.54

Stanford

Slides adapted from Jure Leskovec, CS246 and J. Leskovec, A. Rajaraman, J. Ullman: *Mining of Massive Datasets*

Item-Item vs. User-User

- **In practice, item-item often works better than user-user**
- **Why?** Items are simpler, users have multiple tastes
  - (People are more complex than objects)

Pros/Cons of Collaborative Filtering

+ Works for any kind of item

- No feature selection needed

- Cold Start:

- Need enough users in the system to find a match

- Sparsity:

- The user/ratings matrix is sparse
- Hard to find users that have rated the same items

- First rater:

- Cannot recommend an item that has not been previously rated

- Popularity bias:

- Cannot recommend items to someone with unique taste
- Tends to recommend popular items

- Ethical and social issues:

- Can lead to filter bubbles and radicalization spirals

Collaborative Filtering

Overview

- Recommends based on similar user's preferences.
- No content or domain knowledge needed.
- Requires large amounts of user-item interaction data
- Can be computationally expensive for large datasets

Collaborative Filtering

Utility Matrix

In collaborative filtering utility table/matrix is maintained

	MapReduce	DGIM	PageRank
P	1	1	1
Q	0	1	1
R	1	1	0

1: Watched and 0: Not watched

Advantages

- Personalized recommendations.
- Works well with sparse data.
- No domain knowledge required.
- User behavior analysis.

Disadvantages

- Cold start problem.
- Popularity bias.
- Scalability issues.
- New item problem.
- Privacy concerns.