GRAPH THEORY

Basic properties of graph:

1. Distance b/w 2 vertices
   - Shortest path b/w x & y [no. of edges]
2. Eccentricity
   - Maximum distance b/w 2 vertices.
3. Radius
   - Minimum eccentricity
4. Diameter
   - Maximum eccentricity
5. Centre, central points
   - vertices w/ eccentricity = radius
6. Circumference
   - longest cycle.
7. Girth
   - shortest cycle.

No. of edges in a complete graph: $n(n-1)/2$

Determine the order $|V|$ of the graph (V, E) in the following -

(i) G is a cubic graph (degree 3) w/ 9 edges.

$\Rightarrow \quad 3n = 2 \times e$

$$n = \frac{2 \times 9}{3} = 6.$$

(ii) G is regular w/ 15 edges.

$$k \cdot n = 2 \times e$$

$$k \cdot n = 2 \times 15$$

$$n = 30/k$$

(iii) G has 10 edges w/ 2 vertices of degree 4 and all others of deg. 3.

$$= (2 \times 4) + (3 \times n) = 2 \times e.$$

$$3n = 20 - 8$$

$$n = 4.$$

Binary Relations

Equivalence relation

1. Reflexive
2. Symmetric
3. Transitive

TREES

Let T be a full m-ary tree with $|V| = n$.

$$n = mi + 1$$

$$l = \frac{(m-n)n+1}{m}$$

CUTSETS

Theorem:
Prove that every cutset in connected graph G must contain at least one branch of every spanning tree.

Proof by contradiction

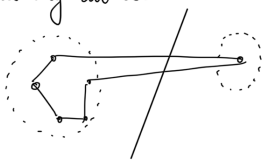
Let G be a connected graph

S → cutset of G.

T → spanning tree

↳ T is completely contained in G−S
↳ G must be connected

Theorem:
Every circuit has an even number of edges in common with any cut set.



Fundamental Cut Set

Consider a spanning tree T of a connected graph G. Let b be a branch in T.

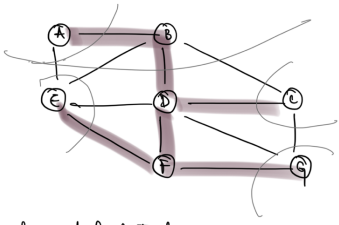
Fundamental Cutset S will contain only 1 branch. Basic cutset

Ring sum of 2 graphs

Graph consisting of V(G) ∪ V(H) & of edges that are either in G or in H but not in both.

$\{d, e, f\} \triangle \{f, g, h\}$

↳ $\{d, e, g, h\}$



∴ Fundamental Cutsets —

Only one branch from spanning tree.

Fleury's & Hierholzer's algorithms.

GRAPH COLOURING

A graph G is k colourable if it has a k-colouring.
The smallest integer k for which the graph is k-colourable is called the chromatic number, $\chi(G)$

$\chi(K_n) = n$ for all $n \geq 1$

$\chi(C_n)_{even} = 2$

$\chi(C_n)_{odd} = 3$.

$\chi(\text{Bipartite}) = 2$

$\chi(\text{Tree}) = 2$

Graph colouring is an NP complete problem

Welch Powell Algorithm.

Kuratowski's Theorem

A graph G is said to be planar iff
- G does not contain K5 or $K_{3,3}$ as subgraph

Petersen graph ⟶ contains $K_{3,3}$

Elementary tests for planarity —

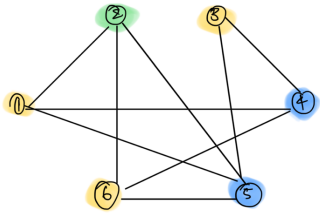
Planar → $e < 9$, $n < 5$ } $n - e + r = 2$

Non planar → $3n - 6 < e$

GRAPH ISOMORPHISM

Two graphs are said to be isomorphic to each other if

- there's a one to one correspondence b/w their vertices and edges such that
- the incidence relationship is preserved.



3 – chromatic

If F is a forest with k components (trees), prove that $n = m + k$

Total number of vertices in the forest = n.
In a tree of order n, no. of edges, $m = n - 1$

A forest is a group of k-disconnected components where each one is a tree.

$n - k = m \implies n = m + k$.

CHROMATIC PARTITIONING

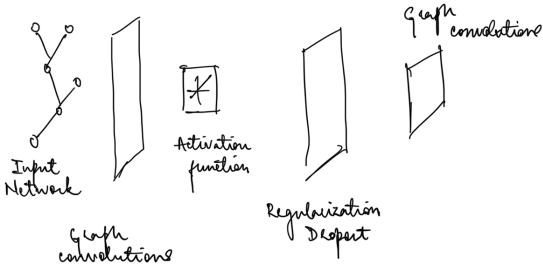
Independent set – no two adjacent edges.

Maximal independence set / Maximal internally stable set – no other vertex added to the set.

Independence no: Max number of vertices in ind. set.

Uniquely colorable graph – graph w/ only one chromatic partioning.

V4
ML w/ GRAPHS



Nodes in a graph are not independent & identically distributed (i.i.d)
↳ semi supervised

Community detection → Unsupervised clustering
eg: functional modules in genetic networks
fraudulent users in financial transaction netwo

Tradional ML with Graphs

Given, $G(V, E)$

learn a function $f: V \longrightarrow \mathbb{R}$

Characterize the structure & position of a node in network

1. Node degree – no. of edges incident on a node.
2. Node centrality ——— Closeness centrality, Betweenness centrality, Eigenvector centrality (
3. Clustering coefficient
4. Graphlets

(i) Eigenvector centrality.

A node v is important if it's surrounded by important neighbouring nodes.

$$c_v = \frac{1}{\lambda} \sum_{u \in N} c_u$$

$\lambda \longrightarrow$ normalization constant
Largest eigenvalue of A.

$$\lambda c = Ac$$

(ii) Betweenness centrality

A node is important if it lies on many shortest paths between other nodes.

$$c_v = \sum \frac{\#(\text{shortest paths b/w } s \text{ \& } t \text{ w/ } v)}{\#(\text{total \# of shortest paths b/w } s \text{ \& } t)}$$

(iii) Closeness centrality

A node is important if it has small shortest path lengths to all other nodes.

$$c_v = \frac{1}{\sum \text{shortest path length b/w } u \text{ \& } v}.$$

3. Clustering Coefficient

Measures how connected v's neighboring nodes are.

$$e_v = \frac{\#(\text{edges among neighboring nodes})}{{}^{k}C_2}$$

GDV $\longrightarrow$ Graphlet Degree Vector provides a measure of a node's local network topology.

Graphlets $\longrightarrow$ Rooted, connected induced non-isomorphic subgraphs.

Kernel Methods for graph level prediction

Design kernels instead of feature vectors.

Kernel SVM used to make predictions

Graph kernels – measure similarity b/w kernels.

(i) Graphlet kernel

(ii) Weisfeiler – Lehman kernel

Bag of $*$ representation of graph (instead of node degrees)

(i) Graphlet Kernel.

Given two graphs G & G', the graphlet kernel is computed as –

$$K(G, G') = f_G^T f_{G'}$$

Normalize each feature vector.

$$h_G = \frac{f_G}{\text{num}(f_G)}$$

$$K(G, G') = h_G^T h_{G'}$$

→ NP hard problem, computationally expensive, n^k

↳ Goal: Define an efficient graph feature descriptor.
Use Colour refinement algorithm.

(ii) Weisfeiller Lehman kernel

Assign an initial colour $c^0(v)$ to every node v.

Iteratively refine node colours.

$$c^{k+1}(v) = \text{Hash}\left(\left\{c^k v, \{c^k(u)\}_{u \in N}\right\}\right)$$

→ Computationally efficient
Linear time, linear in #(edges)

Input Graph → Feature engineering → Structured features
[node level, edge level, graph level]
Learning model (SVM, NN) that maps features to labels.

Map nodes into an embedding space
Encode network information
Used for many downstream predictions

Encoder - Decoder framework

1. Encoder maps from nodes to embeddings
2. Define a node similarity function
3. Decoder maps from embeddings to similarity score
4. Optimize parameters of the encoder.

Node $\xrightarrow{\text{Encoder}}$ Embeddings $\xrightarrow{\text{Decoder}}$ similarity score

similarity $(u, v) \approx z_v^T z_u$
original
↳ optimize similarity function

$$\text{ENC}(v) = z_v \longrightarrow \text{DEC}(z_v^T z_u)$$

embedding lookup

Random walk → Expressivity & efficiency
flexible stochastic def of node similarity

Limitations of shallow embeddings

1. Does not share parameters b/w nodes
Statistically & computationally inefficient
2. Do not leverage node features in encoder
3. Inherently transductive
Only generate embeddings for nodes present during training phase.

Graph embedding → $z_G = \sum_{v \in V} z_v$

- Simple
- Virtual
- Anonymous walk embedding

Graph Neural Network

Map nodes to d-dimensional embeddings such that similar nodes in the graph are embedded close to each other.

Link level features — local neighbourhood overlap.

1. Common neighbours — $|N(v_1) \cap N(v_2)|$
2. Jaccards coefficient — $\frac{|N(v_1) \cap N(v_2)|}{|N(v_1) \cup N(v_2)|}$
3. Adamic Agar index — $|N(v_1) \cap N(v_2)| \frac{1}{\log(C)}$

Normalization, Recenson: $\frac{2\,|N(u) \cap N(v)|}{d_u + d_v}$

Salton index = $\frac{2\,|N(u) \cap N(v)|}{\sqrt{d_u d_v}}$

Global neighbourhood — Katz index

$S = \sum_{i=1}^{\infty} \beta^i A^i = (I - \beta A)^{-1} - I$

U4

Social Network Analysis

Network level measures — Size & Density

Path level measures - Length & Distance

Standard Clustering models

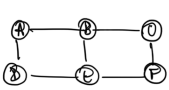
1. Hierarchical (Agglomerative)
2. Point assignment

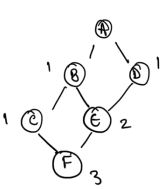
Community detection algorithms

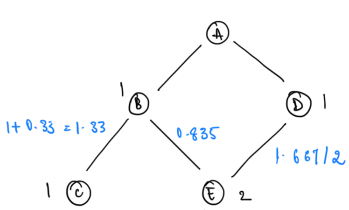
1. Girvan Newman
2. Fluid communities
3. Label propagation
4. Clique percolation
5. Kernighan Lin

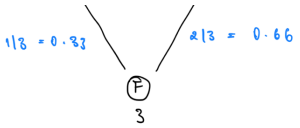
Girvan Newman —

1. Convert to BFS tree.









GRAPH DATABASES

Graph database models –

- RDF [Resource description framework graphs]
- Labelled property graphs

HUNGARIAN METHOD

	J1	J2	J3	J4
C1	4	2	5	7
C2	8	8	10	8
C3	12	5	4	5
C4	6	3	7	14

① Subtract minimum value from each row.

2	0	3	5
5	0	7	5
8	1	0	1
3	0	4	11

② Subtract minimum value from each column.

0	0	3	4
3	0	7	4
6	1	0	0
1	0	4	10

Subtract the smallest element from the uncovered elements and add it to the intersections.

0	1	3	4
2	0	6	3
6	2	0	0
0	0	3	9

	J1	J2	J3	J4
C1	(0)	1	0	1
C2	2	(0)	3	0
C3	9	5	0	(0)
C4	0	0	(0)	6

Trees

If the degree of every non pendant vertex is 3, the number of vertices is an even number.

No. of pendant vertices → n

$$2e = 3(n-n) + n$$
$$2e = 3n - 3n + n$$
$$2e = 3n - 2n$$

$$2e + 2n = 3n$$

$$= \frac{2\left[e+x\right]}{3} = n$$

$2v \longrightarrow 2$.
$4v \longrightarrow 3$
$3v \longrightarrow 4$
$? \longrightarrow 1$

$2e = 4 + 12 + \_ + x$.

$x = 2e - 28$

$e = n - 1$

$n = 2 + 4 + 3 + x = 9 + x$

$e = 8 + x$

$e = 2e - 28 + 8$.

Chromatic Polynomial $P_n(\lambda)$ of a graph w/ n vertices

$P(G, \lambda) = \sum c_i \; {}^{\lambda}C_i$

$P(G, \lambda) = \frac{c_1 \lambda}{1!} + \frac{c_2 [\lambda(\lambda-1)]}{2!} \ldots$.

for a graph with $n = 3$

$c_1 = 0$
$c_2 = 0$
$c_3 = 3! = 6$.

$P(G, \lambda) = c_3 \frac{\lambda(\lambda-1)(\lambda-2)}{3!}$

$= \lambda(\lambda-1)(\lambda-2)$

For a graph w/ $n = 5$.

$c_3 = 3! = 6$.

$c_4 = {}^4P_3 \times 2 = 48$.

$c_5 = 5! = 120$.