Jure Leskovec slides - Stanford

http://cs224w.stanford.edu

Machine Learning with Graphs

$\boxed{\text{01-intro}}$

Book = Will Hamilton - Graph Representation Learning

Google Colab

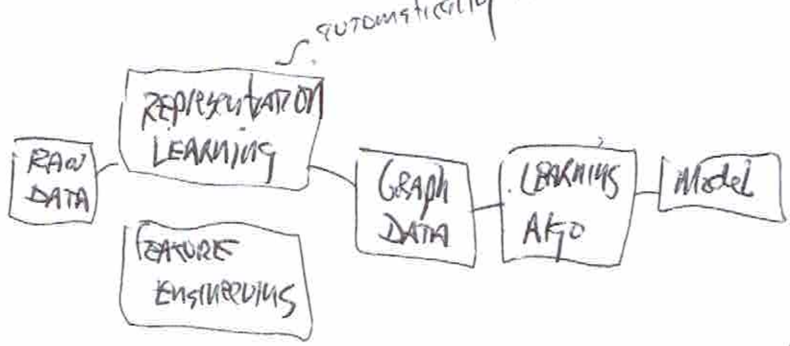
PyTorch = Facebook. (−)	Keras/Tensorflow. = Google. $\boxed{\text{Keras}}$ (+) = NN $\boxed{\text{Tensorflow}}$ general	Theano = U. Montreal

PyG = PyTorch Geometric
LIBRARY for Graph Neural Networks.

Graph Gym.

SNAP, NetworkX

Traditional NN



Map Nodes to d-dimensional embeddings s.t. similar nodes are embedded close together

$f: u \rightarrow \mathbb{R}^d$ Representation.

Feature representation = embedding.

Various Topics:
- Traditional Methods —— Graphlets, Graph Kernels.
- Node Embeddings Methods — DeepWalk, Node2Vec.
- Graph NN —— GCN, GraphSAGE, GAT, Theory & GNNs
- Knowledge Graphs = TransE, BetaE
- Deep Generative Models = ~~TransE~~, ~~BetaE~~ GraphRNN
- Applications

TASKS (GRAPH)

GRAPH-LEVEL prediction — node, edge, Subgraph.
GRAPH generation.

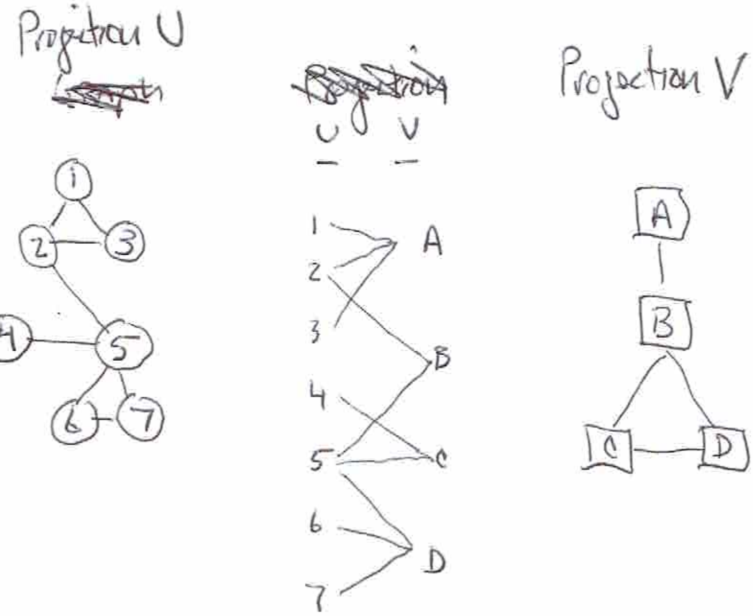
TASKS (ML)

Node classification → Predict Property of node → online ~~buyers~~/items
Link Prediction → Link exist / missing → knowledge graph completion.
Graph classification →
CLUSTERING
Generation
EVOLUTION -

Heterogeneous Graphs

Bipartite
Design
"Projected" Bipartite Graph.

Projected Bipartite Graph



Adjacency Matrix.
Adjacency List,

Node/Edge Attributes...
Weights
Multigraph

Connected
Block-Diagonal form (Non-connected, "Components")
Strongly/Weakly Connected,
"In-Component" Out component
SCC "Strongly Connected Component"

02 - Traditional - ML

Node, Link, graph Prediction.

Traditional ML Pipeline:

Node features $\in \mathbb{R}^D$ Graph features $\in \mathbb{R}^D$

Link features $\in \mathbb{R}^D$

Given a new ⟨Node, Link, Graph⟩ Predict ...

Features are key...

Traditional	(New)
Hand-Designed features	(?)

$\boxed{\text{features}}$ $\boxed{\text{objects}}$ $\boxed{\text{objective function}}$

Node

Node classification

Structure & Position of node in network. = degree, centrality, clustering coefficient, Graphlet = △ □

Katz Index = # of walks of length l

Graph

Kernel Method ... design kernels instead of feature vectors.

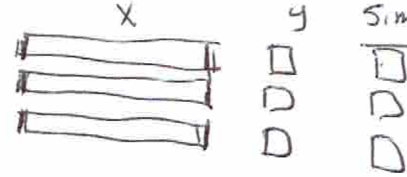
- Remember the i-th training example
- Instance-based learning.

$$\begin{pmatrix} \bar{x}_n \\ y_n \end{pmatrix} \rightarrow w_n$$

"Similarity function" = "kernel"

given input x''

Training   X   y   Similarity function ()



Replaces S.V.M.

$\boxed{\text{Bag-of-}*}$ for our graph kernel

~~Bag-of-degree~~...

Bag-of-degree →

[1 2 1]

[0 2 2]

"Graphlet kernel" and Weisfeiler-Lehman ... Kernel is based on colour assignment computationally efficient

Count graphlets ...

g_1 g_2 g_3 g_4

feature vector is
count of graphlets.

Embeddings: Why?

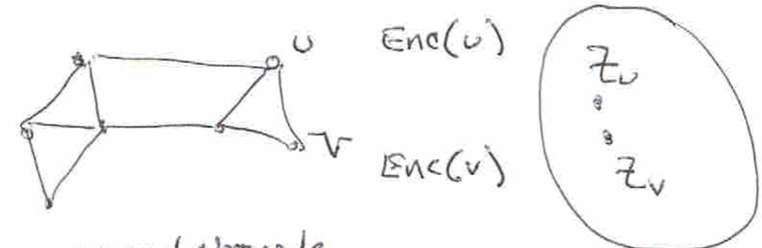
MAP nodes → Embeddings

Similarity of embeddings indicates similarity

[| | | | | | | | |] embeddings → TASKS:

- Node classification
- Link prediction
- Graph classification
- Clustering
- ...

→ GOAL is to map node to embedding space s.t. similarity (dot product) approximates similarity in graph.



original Network.

embedding Space

$$\text{Similarity}(u, v) \approx z_v^T z_u$$

$$(1 \times N)(N \times 1) \Rightarrow 1$$

Learning Embeddings

Encoder — node → Embedding

Similarity function = similarity in original Network

Decoder — Embeddings → similarity score. $DEC(z_v^T z_u)$

Optimize parameters of encoder s.t.

$$\underbrace{\text{Similarity}(u, v)}_{\text{orig. NW.}} \approx \underbrace{z_v^T z_u}_{\text{embedding space.}} \quad (*)$$

Shallow encoding — each node assigned a unique vector

Random-walk-embeddings ...
Stochastic Gradient Descent ...
node2vec ...

} see Goyal and Ferrara 2017 Survey

"Embedding entire graphs"/Subgraphs
- Run on subgraph then average
- VIRTUAL node to represent subgraph
- Anonymous walk

Hierarchical Embeddings (CLUSTERS)

---

04 - Pagerank

Pagerank
- Spider Trap, Teleport.
- Node proximity (consumer, item)
- MATRIX factorization.

---

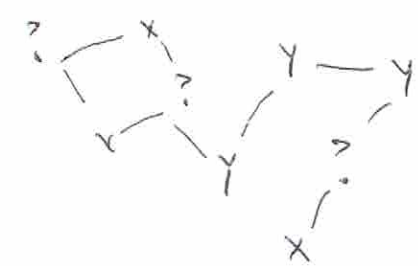
05 - Message Passing

Message Passing & Node classification

Given a network with labels on some nodes how do we assign labels on other nodes?

"Semi-Supervised"

X = trusted
Y = fraudster
? = unknown/Predict.



Method 1 = node embeddings.
Method 2 (today) = message passing
- Relational classification
- Iterative classification
- Correct & Smooth

Correlations in network

Guilt-by-Association

Homophily	Influence
Individual → Social	Social → Individual
Tendency to Bond. Birds of a feather	Social connections can influence individual. I try what my friends like.

05 - message

Probabilistic Relational Classifier = Propagate node labels across network → (does not use node attributes - only labels!)

Iterative Classification - classify node based on attributes as well as labels of neighbor nodes

two classifiers

Correct & Smooth

Given partially labeled graph
1. Train base ~~collector~~ predictor
2. use base predictor to predict node labels
3. Post-process using graph structure to get final node values

---

06 - Graph Neural Networks

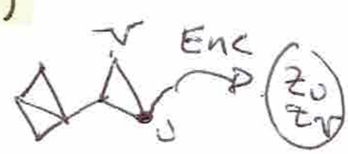
Intuition = map nodes to d-dimensional embeddings s.t. similar nodes are "close"

$f($  $) =$ 

input graph

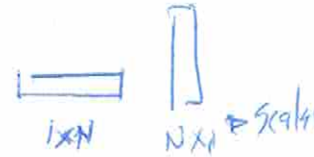
z → node embeddings

Encoder
Encode (v, ~~...~~)
Similarity function
Similarity (v, v)



5 Similarity(u, v) $\approx z_v^T z_u$ (dot product of embeddings)

similarity in original network



7 "Shallow" Encoder "Encoder"

o → [1 | | | | | | |] one-hot

06-GRAPH-NN

v → ENC(v) → embedding

a deep neural network

8 Deep Graph Encoders.

9

"nodes" → representation → nodes "embeddings"

:

1. Instance vs class (1st order vs 2nd order)
2. Forward | Backward (Causal & Diagnostic)
3.

"Aggregate Neighbors"

Every node defines a computation graph based on neighbors.

"Deep Encoder"

66 How to Train a GNN... (Slide 66)

GNN vs CNN

GNN vs Transformer

BASICS of Deep Learning

15, input LABEL

x Y

vector

Sequence

Matrices

Graph

Optimization Problem

$$\min_{\theta} \mathcal{L}(y, f(x))$$

- Loss function
- Set of parameters we optimize

Example Loss function = Cross Entropy

y = categorical encoding

| 0 | 0 | 1 | 0 | 0 | "one hot"

$$CE(y, f(x)) = -\sum_{i=1}^{C} (\underbrace{y_i}_{\text{Actual}} \log \underbrace{f(x)_i}_{\text{Predicted}})$$

17 (*)

How to Optimize...

$$\nabla_\theta \mathcal{L} = \frac{\partial \mathcal{L}}{\partial \theta_n}$$

the lower the loss the closer the prediction is to one-HOT.

30 **Deep LEARNING for Graphs**

- Local Network Neighborhoods
- STACKING multiple Layers

* Graph G = V A $X \in \mathbb{R}^{m \times |V|}$

A: Adjacency Matrix; X: node features

v is a node in V

$N(v)$ is neighbors of v

NAIVE APPROACH:

Join Adjacency matrix + features

(→ $O(|V|)$ parameters.

	A	B	C	D	f_1	f_2
A	0	1	1	0	1	0
B	1	1	0	1	0	1
C	1	1	1	0	1	1
D	1	0	0	0	1	1

(features: f_1, f_2)

CONVOLUTION = generalize and apply across all.

(But graph has no "Locality" or "Sliding window"

→ Graph is "permutation invariant" ≥ NO PRE-set ordering.

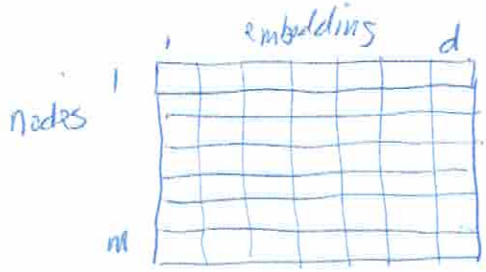
"Order Plan" O-Plan 1 = $A_1 X_1$ O. Plan 2 = $A_2 X_2$

42 A permutation invariant function is needed.

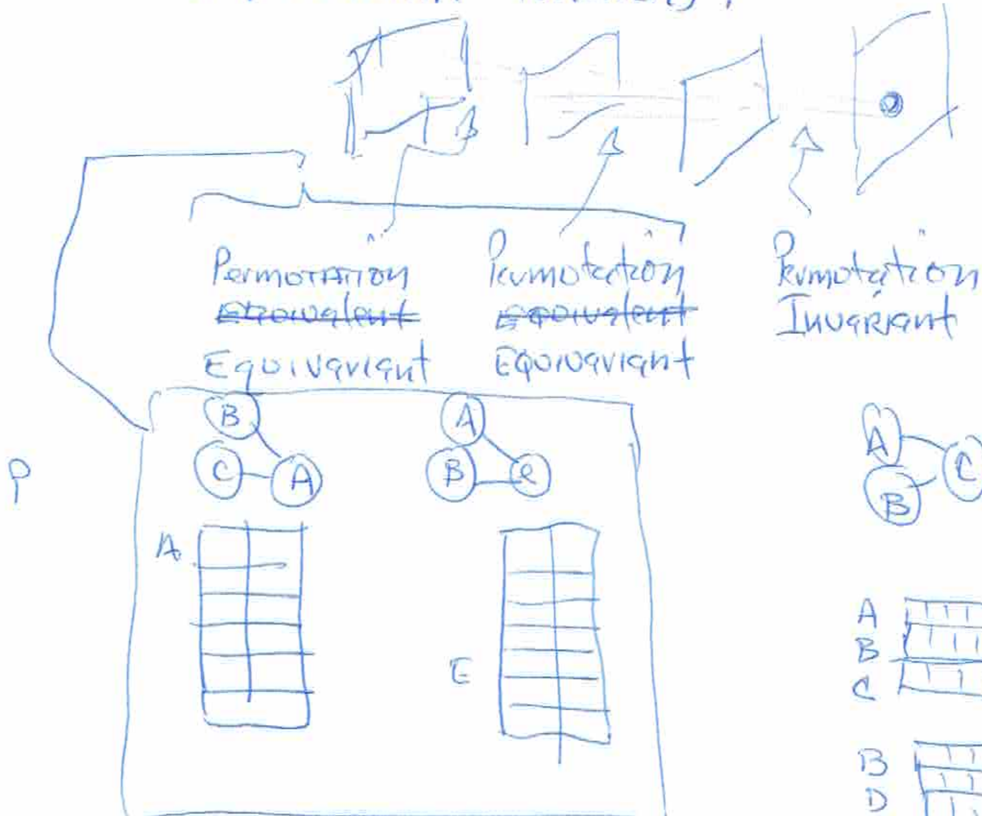
PERMUTATION EQUIVARIANCE

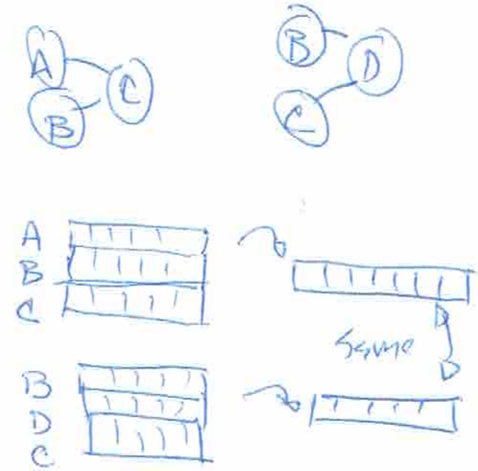
"Graph Representation" ... function maps graph $G = (A, X)$ to $\mathbb{R}^d$

"Node Representation" ... function maps node to $\mathbb{R}^{m \times d}$



A Graph Neural Network consists of multiple Permutation Equivalent / Invariant functions.



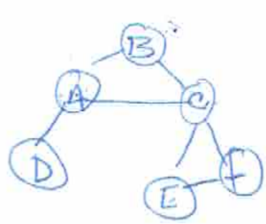


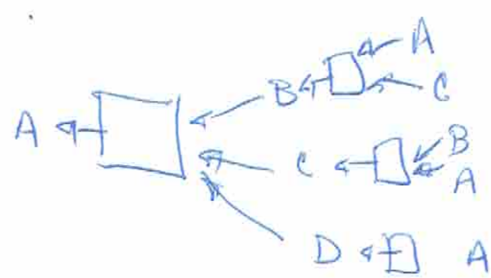
51 Graph Convolutional Network

~~IDEA: Computational Neighborhood~~

IDEA
- Nodes neighborhood defines a "computational Graph"
- Generate node embeddings based on LOCAL network neighborhood.
- Nodes aggregate information from their neighbors

54



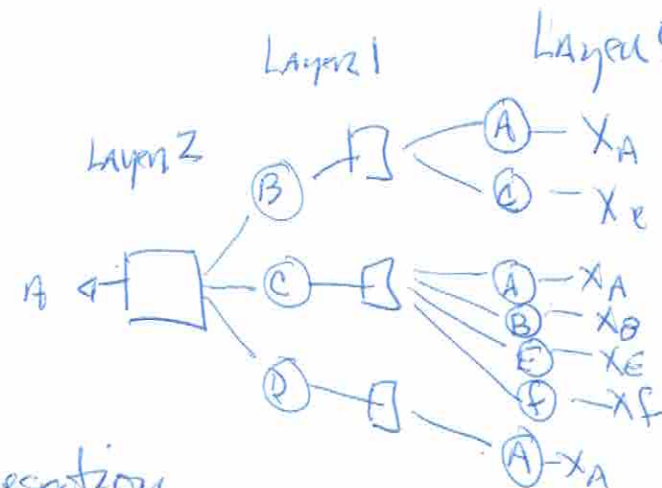


56 Deep Model: Many Layers

Nodes have embeddings at each layer

Layer 0 embedding of node is its feature set x_v

Layer-k embedding gets information from nodes that are k-hops away.



Neighborhood Aggregation

BASIC: Average from neighbors.

Deep Encoder:

Trainable Parameters

Trainable Parameters.

$$h_v^0 = x_v$$

59 $$h_v^{k+1} = \underbrace{\sigma}_{\text{Non-Linearity (Re-LU)}}\Big(W_k \underbrace{\sum_{u \in N} \frac{h_u^k}{|N(v)|}}_{\text{Average of Prior Layer Embeddings}} + B_k \underbrace{h_v^k}_{\text{Embeddings @ k}}\Big)$$

63 $\forall k \in \{0..K-1\}$

ReLU

$z_v = h_v^K$

final embedding

06 Graph NN

How To Train?

64 Need A Loss function...

Matrix formulation...

66 ~~Super~~ How to train:

Supervised... $\mathcal{L}$ could be L2 (numeric) or cross-entropy (nominal)

Unsupervised... use the graph structure.

74 Inductive capability...

76 GNNs subsume CNNs and Transformers

CNN can be seen as a special case of GNN where neighbor (structure) is fixed.
CNN is not permutation independent.

Transformer can be seen as a GNN that runs on a fully connected "word" graph.

07 Graph NN – 2

Project: Graph NN using PyG.

PyG.org – Library for graph learning pipelines

Slack channel

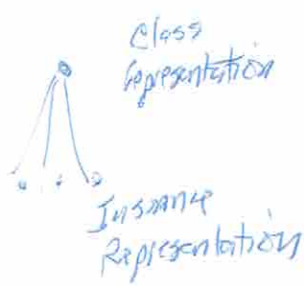
Q: Conferences are ICML NeurIPS ICLR KDD WWW

Blog Posts: Should Contain

1. Step-by-step explanation of graph ML techniques
   Assume reader familiar w/ ML, deep learning, PyTorch.
   Not familiar w/ graph ML, PyG.
2. Visualization – more = better.
   Code snippets of PyG / PyTorch.
3. *** Link to Google Colab.
4. Finding Graph ML Models ← ogb.stanford.edu/docs/leader_nodeprop

Open Graph Benchmark → LEADERBOARD – OBG.

*** → OGB is like Kaggle for Graph NN, and run by Stanford.



First comes in as an instance.
Several come to establish a class

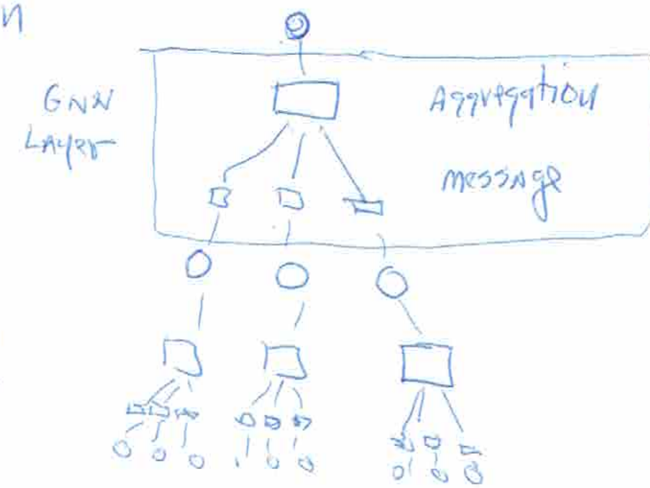
Instances get spawned into "context". Appear – disappear.
Each instance is associated with "tentacles" to the class, and to the evidence

class → Instance
↑
evidence

Recap: Deep Graph Encoder
~~Graph Neural Network~~.
Node's neighborhood defines a "computation graph"
Aggregate from neighbors.

23 GNN Layer = Message + Aggregation

GCN, GraphSage, GAT



31 Message = $m_v^{(l)} = MSG^{(l)}\left(h_v^{(l-1)}\right)$

each node creates a message which will be sent to other nodes later.

$$m_v^{(l)} = W^{(l)} h_v^{(l-1)}$$

AGGREGATION

Node aggregates messages from neighbors.
Information from node itself could get lost

$$h_v^{(l)} = CONCAT\left(AGG\left(\{m_u^l, u \in N(v)\}\right), \underbrace{m_v^l}_{self}\right)$$

Nonlinearity ($\sigma(\cdot)$, ReLU($\cdot$), Sigmoid($\cdot$))

Example:

GCN

Graph SAGE

Long Short-term Memory

38 Aggregation functions = Mean, Pool, LSTM, L2 Normalization

GAT = Graph Attention Network. ?

07 GRAPH NN-2

Graph Attention Networks

42 Attention Mechanism

GNN Layers in Practice

Batch normalization — stabilize re-center (zero-mean)

Dropout — Regularize to prevent overfitting,

Attention / Gating

Activation — ReLU Sigmoid Parametric ReLU

*** Graph Gym

59 Over-smoothing — all the embeddings converge to the same value

Receptive Field = set of nodes that determine (contribute to) the embedding of a node.

⇒ In a K-layer GNN each node has a receptive field of K-hop neighborhood.

▷ Avoid too many layers.

65 Expressive Power for Shallow GNNs

within each layer — each aggregation/transformation be a NN.

Skip connections

08-GNN Application

GNN Augmentation & Training

1 Stacking layers
Over-smoothing
Receptive field
Do not set # layers too large.
Shallow GNN.
14. Skip connections

17 Graph Augmentation for GNNs

IDEA: Raw input graph ≠ computational graph.

Reason to consider input graph ≠ computational graph.

a) graph too sparse
b) graph too dense.
c) graph too large.

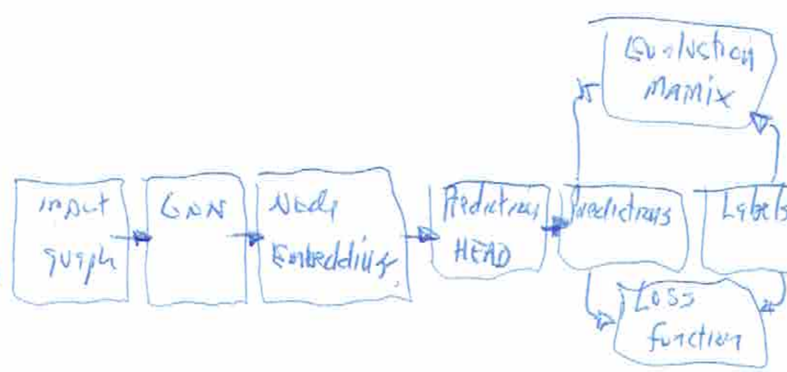
Lacks features → feature augmentation

To sparse, dense, large
Sparse → add virtual nodes
Dense → ~~add~~ sample neighbors.
Large → sample subgraphs.

23 feature augmentation
28 Add virtual nodes
30 Node neighborhood sampling

34 Prediction with GNNs

Training Pipeline 1- Prediction HEAD



38 Prediction Heads

Different tasks require different prediction heads.

Node prediction ... directly
Edge prediction — pair-wise
Graph prediction — using all nodes.

Global pooling loses information:
global mean, max, sum.

Hierarchical Global Pooling

Training Pipeline (2) = Predictions, Labels

Supervised, unsupervised.

Training Pipeline (3) = Loss Function.

54 Classification Loss vs Regression Loss.

Classification Loss → Cross-entropy $= y_j^{(n)} \log(y_j^{(n)})$

n th data point

j th class

· Regression Loss (mean squared error) MSE
a.k.a. L2 loss.

$$= \sum_{j=1}^{K} \left(y_j^{(n)} - \bar{y}_j^{(n)} \right)^2$$

Training Pipeline (4) = Evaluation Metrics.

Evaluation Metrics: Regression
(use sklearn)

RMSE Root Mean-Squared-Error

$$\sum_{n=1}^{N} \frac{(y^{(n)} - \hat{y}^{(n)})^2}{N}$$

MAE Mean Absolute Error

$$\frac{\sum |y^{(n)} - \hat{y}^{(n)}|}{N}$$

Evaluation Metrics: Classification

① Multiclass classification (Accuracy) $\frac{1[\arg\max(\hat{y}^{(n)} = y^{(n)}]}{N}$

② Binary Classification

Accuracy ~~Recall~~
Precision / Recall.
If range of prediction is [0, 1] we use 0.5

Metric Agnostic to classification threshold

ROC AUC

Recall = $\frac{TP}{TP + FN}$ ~~number predicted~~ ~~+ of~~ of actual + the number correctly predicted.

Precision = $\frac{TP}{TP + FP}$ of those predicted + how many are actually +

		Actual +	Actual −
Predicted	+	TP	FP
	−	FN	TN

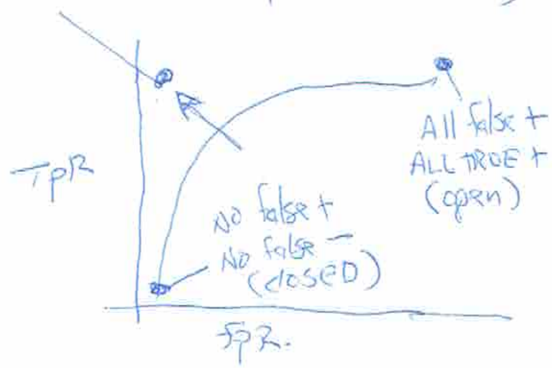
Accuracy = $\frac{TP + TN}{|\text{Dataset}|}$

F1-Score $\frac{2P * R}{P + R}$

*** sklearn classification Report

08 GNN Application

ROC CURVE: Captures the tradeoff in TPR + FPR as classification threshold varies (for a binary classifier)



$TPR = Recall = \frac{TP}{TP + FN}$

$FPR = \frac{FP}{FP + TN}$

Actual

		+	−
Predict	+	TP	FP
	−	FN	TN

$ROC = \frac{TPR}{FPR} = \frac{TP}{TP + FN} \cdot \frac{FP + TN}{FP}$

Dataset split — this is different for graphs...

Image Database – each (independent)

Graph – not independent.

Node Prediction ~~Transd~~

Solution 1 = Transductive Setting.

- only split the node labels.
  - ALL EDGES (entire graph)
  - only certain nodes' data

Solution 2

- BREAK up the graph to get multiple graphs

Graph Prediction: for graph classification only inductive setting.

Link Prediction: Hide some edges.

87 Debugging Deep Networks
Loss/Accuracy not converging
overfit on subset of training data
scrutinize & loss visualizations

Resources
Graph Gym
PyG
DGL
PyTorch
Graph Nets
TensorFlow

85 When things Dont go as planned...
data preprocessing
ADAM optimizer
Activation f
include BIAS term
dimensions

89 RESOURCES

09 Theory

26 How Expressive are GNNs

· Expressive Power
Characterized by neighbor aggregation function used

"GCN" = mean-pool - uses element-wise mean pooling over neighbouring node features
(Kipf, Welling)

$$\text{Mean}\left(\{ X_u \}_{\underbrace{u \in N(v)}} \right)$$

over the neighbour nodes
mean of features

"GraphSAGE" = max-pool - uses element-wise pooling over neighbouring node-features
(Hamilton et al)

(above are shown to be "not injective" - thus not maximally powerful

62 Injective Multi-set

10 – Knowledge Graph

4 Heterogeneous Graph : $G = (V\ E\ R\ T)$

Nodes $v_i \in V$

Edges $(v_i, r, v_j) \in E$ ← directed

Node Type $T(v_i)$ ← Node has type

Relation Type $r \in R$ ← Edge has type

8 Relational GCN directed w/ $\underline{\underline{1}}$ Relation

28 Knowledge Graph entities (nodes) w/ Relation (edges)

48 TransE "Completion"?

56 TransR

60 DistMul

69 ComplEx

"REASONING" in Knowledge Graphs

Can we "complete" the K.G?

Given a head, relation we predict the missing tails ...

e.g Predict the tail "science fiction" for "J.K. Rowling" "genre"

- multi-hop queries

Query2box

- Path queries

ANSWERING Predictive Queries

34 KGs Using Box Embeddings

67 TRAINING Query2Box

80 VISUALIZING

*** "We use t-SNE to reduce the embedding space to a 2-dimensional space to visualize the query results

Fast Neural Subgraphs

Building Blocks -
Characterize
Discriminate

7 ① Node-induced subgraph
subset of nodes, all edges induced by them.

② Edge induced subgraph -
subset of edges and all corresponding nodes.

"Network Building Blocks

11 "Graph Isomorphism"
Check whether two graphs are identical
sub

15 Network Motifs
Recurring significant patterns -
ie. "Induced Subgraphs"

18 Subgraph Frequency
Motif Significance - null model → Test → "Significance"
Random Graph i.e. Erdos Renyi

33 Neural Subgraph Representations
Neural architecture for subgraph matching

60 Finding Frequent Subgraphs
Solution = Representation Learning   Combinatorial Explosion → oversize search space.

*** SpMiner = identify frequent Motif,   Subgraph Isomorphism

13

13 - Recommender Systems

GNNs for Recommender Systems

Netflix Amazon Spotify YouTube Pinterest.
movies Products music videos Pins (images)

Bipartite graph

- NGCF - Wang — Neural Graph Collaborative Filtering
- LightGCN - He
- PinSAGE - Ying

TRADITIONAL Collaborative filtering - BASED on MATRIX factorization.
USES "shallow encoders for users/items

Limitations = only first-hop relations expressed.
= DOES NOT consider graph structure.
GNN addresses both.

32 NGCF

39 Light GCN

55 Pin SAGE

14 Communities

Community Detection in Networks -

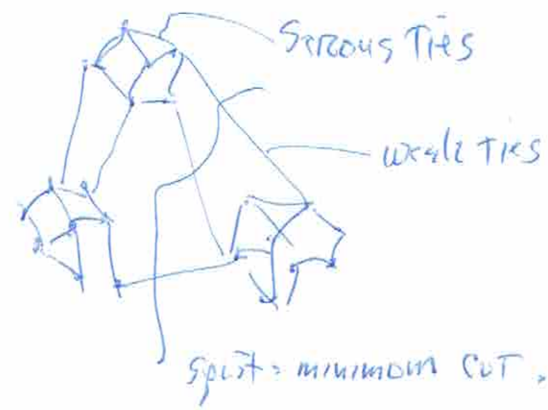
Granovetter - friendship is structural, Interpersonal

Triadic closure

clusters/groups/modules/
communities

Zachary's Karate Club Network

find micro-markets.



26 Null Model

30 Louvain Algorithm = Greedy Algorithm for Community Detection

$O(n \log n)$ run time

Algorithm greedily maximises "modularity"

① Local changes to node - community

② communities are aggregated into super-nodes

47 Detecting Overlapping Communities

Community Affiliation Graph Model (AGM)

NOCD model

Deep Generative Models for Graphs

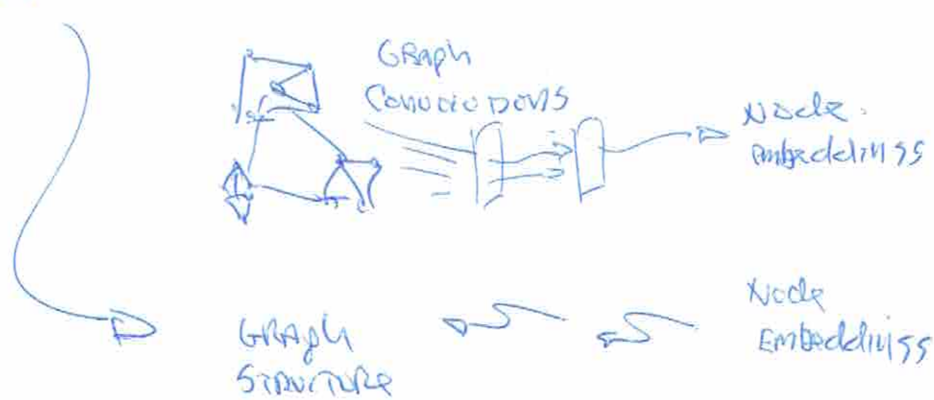
IDEA: So far we have been learning from graphs.
ASSUMED GRAPHS are given

- We want to generate realistic graphs using graph generative models.

Synthetic graph → insights
→ Predictions
→ Simulations
→ Anomaly

History: Properties of Real-world graphs
Traditional graph generative
Deep graph generative models
learn graph formulation process from the data

7 "Decoders"



9 GOAL: generate graph similar to set of graphs

15 GRAPH RNN
(GRAPH as sequence)

44 Scaling & Evaluating Graph Generation

58 Application: DEEP Graph Generative Model to Molecule Generation