11-741/11-441:

Machine Learning with Graphs

Introduction

Yiming Yang

1

Outline

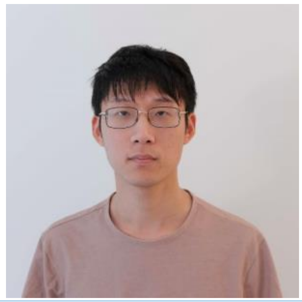
- Administrative Stuff
- Course Contents Overview

2

Instructor and Teaching Assistant

- Yiming Yang (LTI & MLD)
- Office hours by appointment (GHC 6717 or via zoom)
- yiming@cs.cmu.edu

- Zhiqing Sun (PhD in LTI)
- Office hours: See piazza
- <zhiqings@andrew.cmu.edu>



3

Background

- **Prerequisites**
  - CS courses like data structures, algorithms, programming (e.g., 15-213)
  - Linear algebra (e.g., 21-241 or 21-341), introductory probability (e.g., 21-325)I
- **Preferred but not required**
  - Introductory Machine Learning (e.g., 10-701 or 10-601)
  - Neural network courses
  - This course is mostly self-contained on ML background.

4

Sections

- **11-741** (graduate level, 12 units)
  - Previously 11-741 (PhD level) and 11-641 (MS level) are now merged into one without distinction
  - 100% homework (5 HWs) & 100% of the Exam Questions (midterm exam and final exam)
- **11-441** (undergraduate level, 9 units)
  - 80% homework (4 out of the 5HWs) by your own choices; if you do all the 5 HWs, the top-4 scores will be used in grading.
  - 70% of the total exam questions by your own choices; if you choose to do more, only the 70% of the best answered questions will be used in grading.

Grading Policies

	11-741/641 (Grad Level)	11-441 (UG Level)
Midterm Exam	15%	14%
Final Exam	15%	14%
HWs	14% x 5 = 70%	18% x 4 = 72%

Homework

- Programming assignments in Python
- Submission: Due by 11:59pm of the due date.
  - It must be submitted by Gradescope. If Gradescope is down, it must be submitted by email to the TA.
  - A 10% penalty is applied for each day beyond the deadline.

Cheating, Copying, Plagiarism, Etc

- You must be the author of everything that you submit for a grade
- Revising or modifying someone else's work does not make you the author
- It is okay to discuss homework with other students, share ideas, experience, and lessons learned
- Sign the cheating policy form (as the condition to be graded)

Exams

- Open book, with a set of questions (about 10) and a list of possible answers to choose from per question.
- Mid-term exam will cover the 1st half of the lecture contents of the semester, and the final exam will cover the 2nd half.
- The exams will not focus on the contents of the HW assignments.
- No arrangement
  - if you cannot attend the exams, you will just lose the points

Course Materials Online

- **Syllabus** (publicly available)

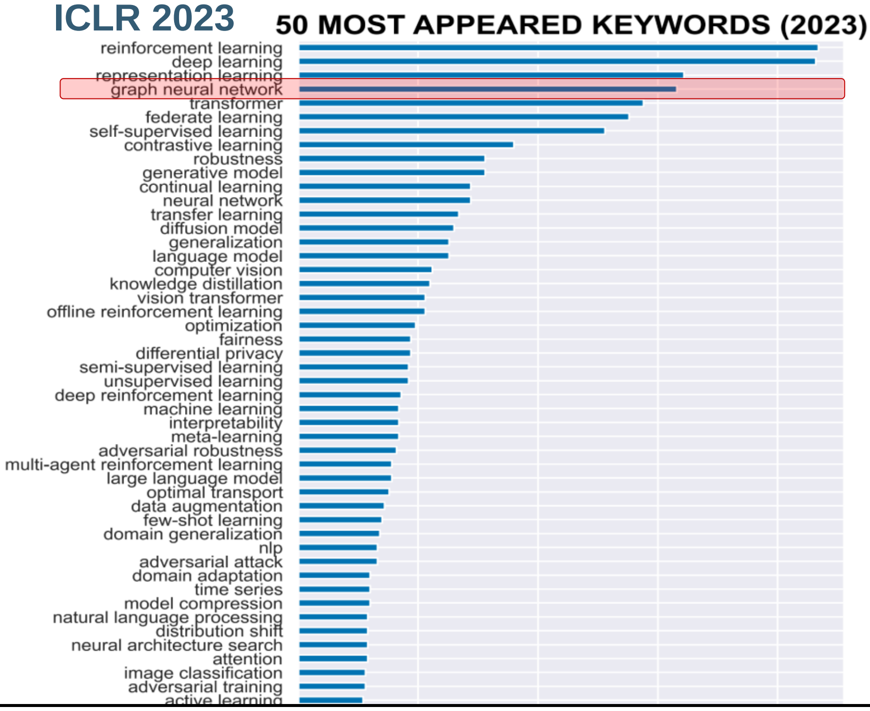
  https://cmu-ml4graph.github.io/s2024/
- **Lecture Slides** (password protected)
  - URLs listed on the schedule of the lectures
  - Login information will be announced via piazza
- **Piazza** (listed at Canvas)
- **Recorded Lectures**
  - Not provided in general, to encourage in-person classes
  - Exceptions (if you catch COVID) can be arranged via the TA

Outline

✓Administrative Stuff

- Course Contents Overview
  - with motivating examples

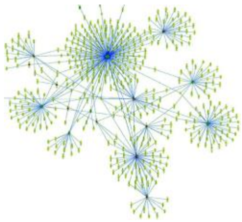
ICLR 2023



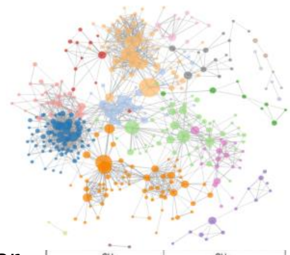
Why graphs?

A common language for representing various types of data

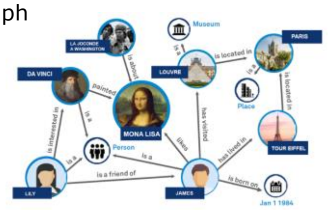
Web Site Hyperlinks



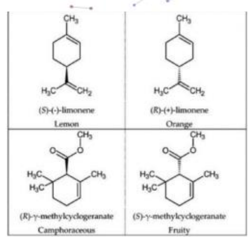
Citation Network



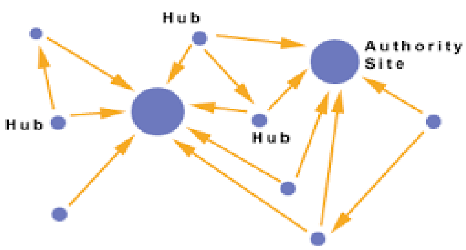
Knowledge Graph



Molecular Structures



Ex 1: Web Page Popularity Analysis

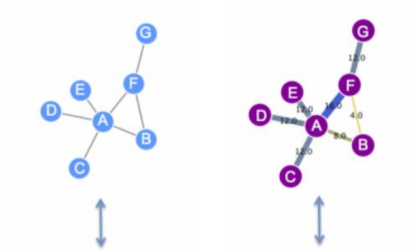


- Which web pages are more trustworthy (good authorities)?
- Which web pages are more resourceful (good hubs)?
- Naïve answer: Count the in/out degrees of each node.
- Better answer
  - A node is authoritive if it is pointed by many good hubs.
  - A node is a good hub if it points to many authoritive nodes
- We end up with a chicken-egg problem!

Representing the Graph Structure

- **Graph** $\boldsymbol{G = (V, E)}$
  - with vertices (V) and edges (E)
- **Adjacency Matrix** $\boldsymbol{A_{n \times n}}$
  - Edge weights, which can be binary (left) or weighted (right)
- **Graph Laplacian** $\boldsymbol{L \overset{\text{def}}{=} D - A}$
  - D is a diagonal matrix with

$$D_{ii} = \sum_{j:j \neq i} A_{ij}.$$



	A	B	C	D	E	F	G
A	0	1	1	1	1	1	0
B	1	0	0	0	0	1	0
C	1	0	0	0	0	0	0
D	1	0	0	0	0	0	0
E	1	0	0	0	0	0	0
F	1	1	0	0	0	0	1
G	0	0	0	0	0	1	0

	A	B	C	D	E	F	G
A	0	8	12	12	12	16	12
B	8	0	0	0	0	4	0
C	12	0	0	0	0	0	0
D	12	0	0	0	0	0	0
E	12	0	0	0	0	0	0
F	16	4	0	0	0	0	12
G	12	0	0	0	0	12	0

Adjacency matrices

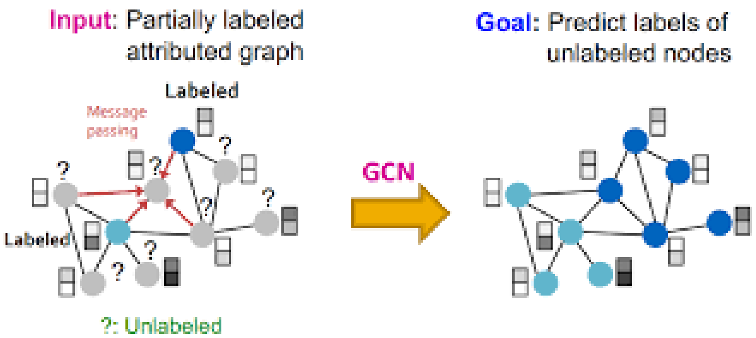
15

How to deal with the chicken-egg problem?

- HITS (J. Kleinberg, 1998)
  - Calculate $u_1 :=$1st eigenvector of $(A^T A)$
    $v_1 :=$1st eigenvector of (AA^T)
  - u_1 gives the **authority** scores of nodes, and v_1 gives the **hub** scores of nodes.
- PageRank (S. Brin and L. Page, 1998)
  - Define probiotic transition matrix $M_{n \times n}$ with $M[i,j] = P(j|i)$;
  - Calculate $r_1 :=$1st eigenvector of a smoothed version of matrix M, which gives the **PageRank** scores of nodes.
- Both methods utilize the eigendecompostion of those matrices, based on **random walk** over a graph with infinite steps

16

Ex 2. Graph Convolution Network (GCN)
for Semi-supervised Classification [ICLR 2017]



Key Idea: Propagating node features (embeddings) over the graph, yielding the connected nodes with smooth labeling.

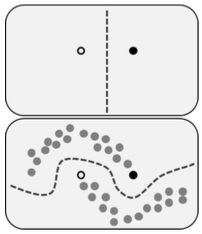
Fundamental Question
(since internet became popular in mid 1990's)

What is a document, anyway?

1. A piece of text (a human point of view)
2. A bag of words (a traditional IR point of view)
3. A sequence of tokens (a neural language model point of view)
4. A bag of links (from the graph connectivity point of view)
5. A bag of linked pages (each link reaching out a web page)
6. A node in a connected graph (each node have its own words).

Key Idea in SSL for Node Classification

Upper: Decision boundary based on labeled data only



Lower: Decision boundary based on labeled + unlabeled data

Key Question: How do we represent the manifold in data?

Controlling the Smoothness of Node Embedding

- Graph Laplacian $L \stackrel{\text{def}}{=} D - A$
  - A is the adjacency matrix
  - D is a diagonal matrix, with $D_{ii} \sum_{j:j \neq i} A_{ij}$
- Using (a subset) the eigenvectors of L we can control the smoothness of node embedding over the graph.

Various Graph Neural Networks

- Graph **Convolution** Network (GCN) [ICLR 2017]
  - node-level classification
- Graph **Attention** Network (GAT) [ICLR 2018)
  - node-level classification by leveraging masked self-attentional layers
- Graph **Isomorphism** Network(GIN) [ICLR 2019]
  - graph-level classification with multi-layer perceptron
- Graphormer [NeurIPS 2021]
  - graph-level classification with **graph-adapted Transformer**
- SignNet [ICLR 2023]
  - graph-level regression with Laplacian Eigenvectors for graph positional encoding

21

Part I. Fundamentals & Building Blocks

- Word2vec Embedding Methods (1 lecture)
- Recurrent Neural Networks (RNN) (1 lecture)
- Convolution Neural Networks (CNN) (1 lecture)

  HW1. CNN & RNN classifiers
- Attention Models (1 lecture)
- LM Architectures (1 lecture)
- Classification Fundamentals (4 lectures)

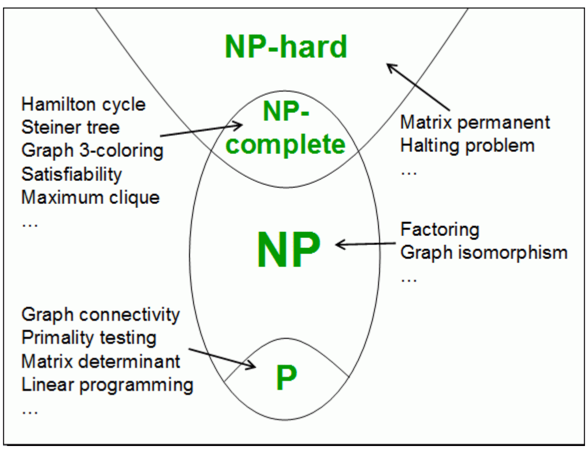
  HW2, Soft-max & SGD

22

Part II. Graph-based Learning Topics

- ✓ **Social Popularity Analysis (3 lectures)**
  - HW3. PageRank models
- Node Embedding (1 lecture)
- ✓ **Graph Neural Networks for Classification (2 lectures)**
  - HW4. GCN models
- Knowledge Graph Embedding (2 lectures)
  - HW5. Node Embedding with TransE
- Neural Solvers for Combinatorial Optimization (3-4 lectures)
- Reasoning w/ Heterogeneous Graphs (2 lectures)
- Invited Talks (2; industrial applications & insights)

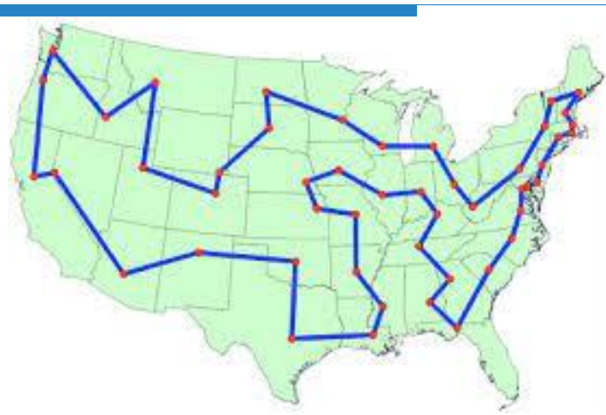
Computational Problems



NP-Completeness (NPC)

- NPC is "the hardest problems in NP"
- If some NPC problem has a polynomial time algorithm, all problems in NP do.
- Our focus is to on recent neutral network solvers
  - For large NPC problems that traditional solvers cannot handle
  - With graph-based learning and approximation techniques

Ex 3. Traveling Salesman Problem (TSP)



Task: Given a graph with n nodes, find the shortest tour where each node is visited once and only once expect the starting node (as the ending node).

Relating TSP to LM

ChatGPT Input

- Prompt: a sequence of words as the instruction

ChatGPT output

- Response: the generated sequence of words per the instruction

TSP Input

- A set of nodes with 2D coordinates in a graph

TSP Output

- The predicted sequence of nodes where each node appears once and only once, and the last node is the starting node.

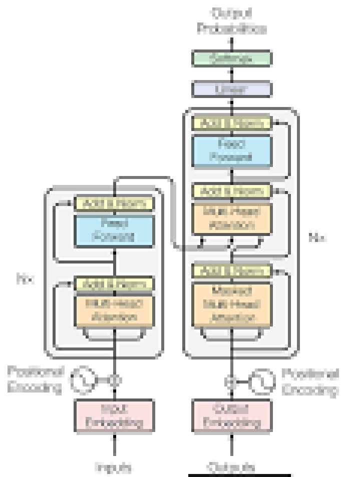
Existing TSP Solvers

- **Exhaustive search is not tractable for large n**
  - $n!$ feasible solutions
  - Dynamic Programming (Held-Karp) takes $O(n^2 2^n)$ time
- **Hand-crafted Heuristic Solvers** (traditional in OR)
  - Not generalizable across problems
- **Deep Reinforcement Learning (DRL,** recent in ML)
  - Not relying on hand-crafted heuristics
  - Learning from a large training-set of graphs for smart search
  - Model applicable to new graphs beyond training examples
  - No need to know the optimal solution(s) for each training graph

Encoder-Decoder Architecture in NLP

BERT

Encoder



GPT

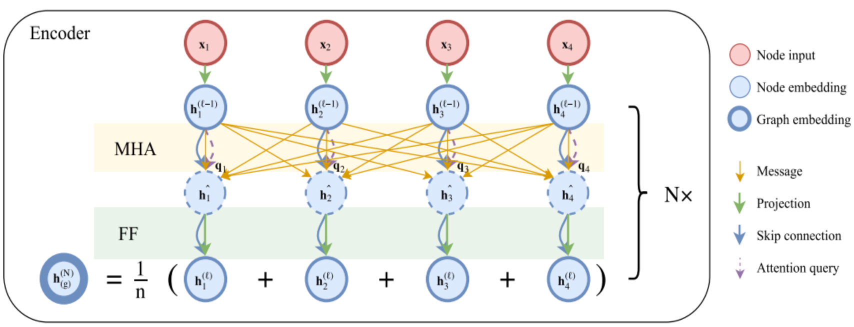
Decoder

1/16/2024 @Yiming Yang, 11-741 S24 Introduction[redacted]

29

Learn to Solve NPC with DRL

(Kool et al., ICLR 2019 - Transformer Encoder)

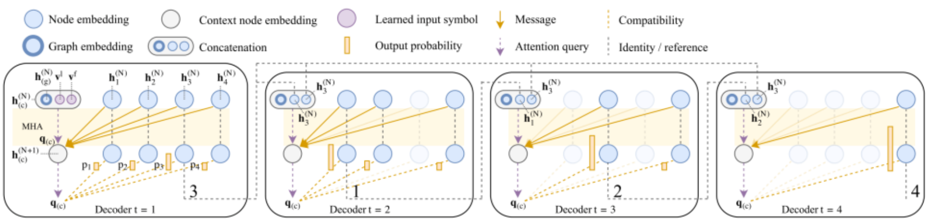


Kool, Wouter, Herke van Hoof, and Max Welling. "Attention, Learn to Solve Routing Problems!." International Conference on Learning Representations. 2019.

30

30

Learn to Solve NPC with DRL
(Kool et al., ICLR 2019 - Transformer Encoder)



Kool, Wouter, Herke van Hoof, and Max Welling. "Attention, Learn to Solve Routing Problems!." International Conference on Learning Representations. 2019.

Recent Developments

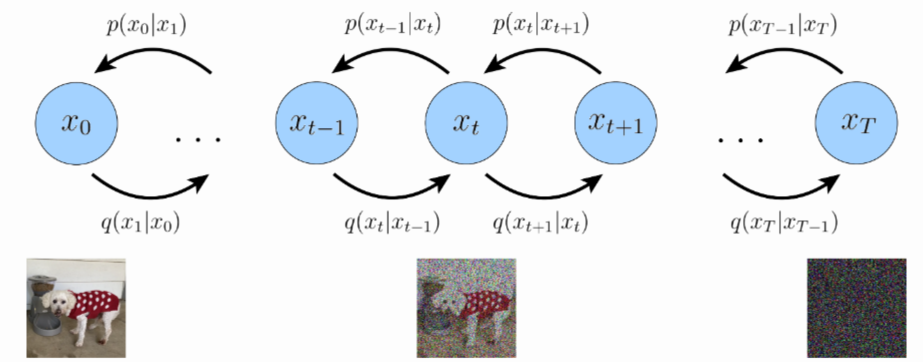
- **Good News in 2019**
  - DRL solvers for TSP found near-optimal solutions on evaluation benchmarks
- **Bad News before 2022**
  - Neural-net solvers for TSP **only scale to graphs with $\leq$ 100 nodes**
  - Neural network architectures for TSP may not generalize to other NPC problems (e.g., Maximum Independent Set)
- **Recent Advances**
  - DIMES (Qiu*, Sun*, Yang: NeurIPS 2022): neural DRL solver scale to graphs with 10,000 nodes and improved SOTA results
  - DIFUSCO (Sun, Yang: NeurIPS 2023): neural diffusion model outperforming DIMES in both accuracy and inference time complexity
  - Both offer a generic framework for NPC problems, including TSP, MIS, etc.

High-Level Ideas

- **DIMES (Qiu*, Sun*, Yang, NearIPS 2022)**
  - Removing the decoder part (costly) and modeling with encoder-only
  - Introducing meta learning to make up the (lost) accuracy
- **DIFUSCO (Sun, Yang, NearIPS 2023)**
  - Borrowing the success of “diffusion” from computer vision (Decomposing a tough task into a sequency of easier tasks)
  - Highly efficient due to parallelable training
  - The first graph-based diffusion framework (as apposed to image-based)
  - New SOTA results in TSP and MIS

33

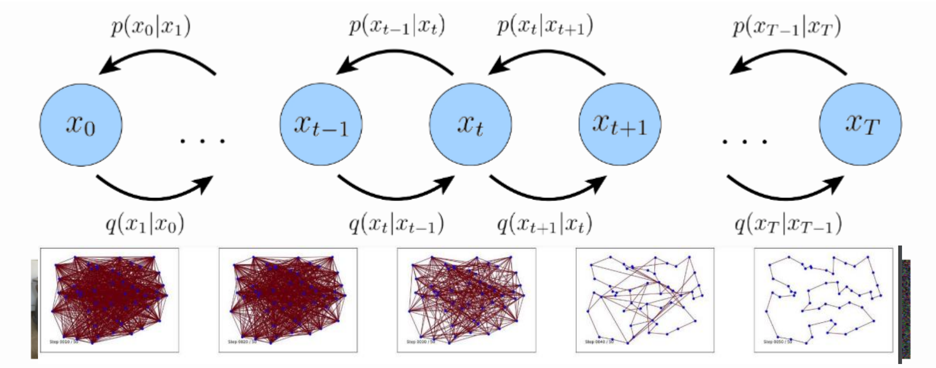
A Diffusion Model in Computer Vision



Visual representation of a Variational Diffusion Model. An input is steadily noised over time until it becomes identical to Gaussian noise; a diffusion model learns to reverse this process.

34

Our Diffusion Model for TSP



Visual representation of a Variational Diffusion Model. An input is steadily noised over time until it becomes identical to Gaussian noise; a diffusion model learns to reverse this process.

Concluding Remarks

- **Why graphs?**
  - A common language for representing many types of entities, relations and human knowledge, supporting graph-based reasoning beyond bag/sequence of tokens
  - Broad applications (social impact analysis, classification/regression tasks, solving NPC problems, and more)
- **Connections to recent deep learning**
  - Novel adaptation/enrichment of popular LLMs, Deep Reinforcement Learning (DRL) and Diffusion Models