Graph Neural Networks (GNNs)

Nguyễn Quản Bá Hồng*

Ngày 4 tháng 8 năm 2025

Tóm tắt nội dung

This text is a part of the series *Some Topics in Advanced STEM & Beyond*:
URL: https://nqbh.github.io/advanced_STEM/.
Latest version:

- .
  PDF: URL: .pdf.
  TEX: URL: .tex.
- .
  PDF: URL: .pdf.
  TEX: URL: .tex.

Mục lục

1 Introduction to Graph Neural Networks

1.1 Keita Broadwater, Namid Stillmann. Graph Neural Networks in Action

- Fig: Mental model of GNN project. The steps involved for a GNN project are similar to many conventional ML pipelines, but we need to use graph-specific tools to create them. Start with raw data, which is then transformed into a graph data model & that can be stored in a graph database or used in a graph processing system. From the graph processing system (& some graph database), we can do exploratory data analysis & visualization. Finally, for graph ML, we preprocess data into a format that can be submitted for training & then train our graph ML model, in our examples, these will be GNNs. $\boxed{\text{GNNs} \subset \text{Graph ML models}}$.

  – Hình: Mô hình tinh thần của dự án GNN. Các bước cần thực hiện cho một dự án GNN tương tự như nhiều quy trình ML thông thường, nhưng chúng ta cần sử dụng các công cụ dành riêng cho đồ thị để tạo ra chúng. Bắt đầu với dữ liệu thô, sau đó được chuyển đổi thành mô hình dữ liệu đồ thị & có thể được lưu trữ trong cơ sở dữ liệu đồ thị hoặc sử dụng trong hệ thống xử lý đồ thị. Từ hệ thống xử lý đồ thị (& một số cơ sở dữ liệu đồ thị), chúng ta có thể thực hiện phân tích dữ liệu thăm dò & trực quan hóa. Cuối cùng, đối với ML đồ thị, chúng ta xử lý trước dữ liệu thành một định dạng có thể được gửi để huấn luyện & sau đó huấn luyện mô hình ML đồ thị của chúng ta, trong các ví dụ của chúng ta, đây sẽ là các GNN. $\boxed{\text{GNN} \subset \text{Mô hình ML đồ thị}}$.

- Foreword. Our world is highly rich in structure, comprising objects, their relations, & hierarchies. Sentences can be represented as sequences of words, maps can be broken down into streets & intersections, www connects websites via hyperlinks, & chemical compounds can be described by a set of atoms & their interactions. Despite prevalence of graph structures in our world, both traditional & even modern ML methods struggle to properly handle such rich structural information: ML conventionally expects fixed-sized vectors as inputs & is thus only applicable to simpler structures e.g. sequences or grids. Consequently, graph ML has long relied on labor-intensive & error-prone handcrafted feature engineering techniques. Graph neural networks (GNNs) finally revolutionize this paradigm by breaking up with regularity restriction of conventional DL techniques. They unlock ability to learn representations from raw graph data with exceptional performance & allow us to view DL as a much broader technique that can seamlessly generalize to complex & rich topological structures.

*A scientist- & creative artist wannabe, a mathematics & computer science lecturer of Department of Artificial Intelligence & Data Science (AIDS), School of Technology (SOT), UMT Trường Đại học Quản lý & Công nghệ TP.HCM, Hồ Chí Minh City, Việt Nam.
E-mail: nguyenquanbahong@gmail.com & hong.nguyenquanba@umt.edu.vn. Website: https://nqbh.github.io/. GitHub: https://github.com/NQBH.

– Thế giới của chúng ta vô cùng phong phú về cấu trúc, bao gồm các đối tượng, mối quan hệ của chúng, & hệ thống phân cấp. Câu có thể được biểu diễn dưới dạng chuỗi từ, bản đồ có thể được chia nhỏ thành các con phố & giao lộ, www kết nối các trang web thông qua siêu liên kết, & hợp chất hóa học có thể được mô tả bằng một tập hợp các nguyên tử & tương tác của chúng. Mặc dù cấu trúc đồ thị rất phổ biến trong thế giới của chúng ta, cả các phương pháp ML truyền thống & thậm chí hiện đại đều gặp khó khăn trong việc xử lý đúng cách thông tin cấu trúc phong phú như vậy: ML thường mong đợi các vectơ có kích thước cố định làm đầu vào & do đó chỉ áp dụng cho các cấu trúc đơn giản hơn, ví dụ như chuỗi hoặc lưới. Do đó, ML đồ thị từ lâu đã dựa vào các kỹ thuật thiết kế đặc trưng thủ công tốn nhiều công sức & dễ xảy ra lỗi. Mạng nơ-ron đồ thị (GNN) cuối cùng đã cách mạng hóa mô hình này bằng cách phá vỡ sự hạn chế về quy tắc của các kỹ thuật DL thông thường. Chúng mở khóa khả năng học các biểu diễn từ dữ liệu đồ thị thô với hiệu suất vượt trội & cho phép chúng ta xem DL như một kỹ thuật rộng hơn nhiều, có thể khái quát hóa liền mạch thành các cấu trúc tôpô phức tạp & phong phú.

When Matthias Fey – creator of PyTorch Geometric & founding engineer Kumo.AI – begin to dive into field of graph ML, DL on graphs was still in its early stages. Over time, dozens to hundreds of different methods were developed, contributing incremental insights & refreshing ideas. Tools like our own PyTorch Geometric library have expanded significantly, offering cutting-edge graph-based building blocks, models, examples, & scalability solutions. Reflecting on this growth, it is clear how overwhelming it can be for newcomers to navigate essentials & best practices that have emerged over time, as valuable information is scattered across theoretical research papers or buried in implementations in GitHub repositories.

– Khi Matthias Fey – người sáng lập PyTorch Geometric & kỹ sư sáng lập Kumo.AI – bắt đầu dấn thân vào lĩnh vực học máy đồ thị, học máy trên đồ thị vẫn còn ở giai đoạn sơ khai. Theo thời gian, hàng chục đến hàng trăm phương pháp khác nhau đã được phát triển, đóng góp những hiểu biết sâu sắc & những ý tưởng mới mẻ. Các công cụ như thư viện PyTorch Geometric của chúng tôi đã mở rộng đáng kể, cung cấp các khối xây dựng, mô hình, ví dụ, & giải pháp khả năng mở rộng dựa trên đồ thị tiên tiến. Nhìn lại sự phát triển này, rõ ràng là những người mới bắt đầu có thể gặp khó khăn như thế nào khi tìm hiểu những điều cốt lõi & các phương pháp hay nhất đã xuất hiện theo thời gian, khi thông tin giá trị nằm rải rác trong các bài báo nghiên cứu lý thuyết hoặc bị chôn vùi trong các triển khai trên kho lưu trữ GitHub.

Now power of GNNs has been widely understood, this timely book provides a well-structured & easy-to-follow overview of field, providing answers to many pain points of graph ML practitioners. Hands-on approach, with practical code examples embedded directly within each chap, invaluably demystifies complexities, making concepts tangible & actionable. Despite success of GNNs across all kinds of domains in research, adoption in real-world applications remains limited to companies that have enough resources to acquire necessary knowledge for applying GNNs in practice. Confident: this book will serve as an invaluable resource to empower practitioners to over that gap & unlock full potentials of GNNs.

– Giờ đây, sức mạnh của GNN đã được hiểu rộng rãi, cuốn sách kịp thời này cung cấp một cái nhìn tổng quan được cấu trúc tốt & dễ hiểu về lĩnh vực này, giải đáp nhiều vấn đề khó khăn của các chuyên gia ML đồ thị. Phương pháp tiếp cận thực hành, với các ví dụ mã thực tế được nhúng trực tiếp trong mỗi chương, giúp làm sáng tỏ những điều phức tạp, biến các khái niệm thành hiện thực & khả thi. Mặc dù GNN đã thành công trong nhiều lĩnh vực nghiên cứu, việc áp dụng vào các ứng dụng thực tế vẫn chỉ giới hạn ở các công ty có đủ nguồn lực để có được kiến thức cần thiết cho việc áp dụng GNN vào thực tế. Tự tin: cuốn sách này sẽ là một nguồn tài nguyên vô giá giúp các chuyên gia vượt qua khoảng cách đó & khai phá toàn bộ tiềm năng của GNN.

- Preface. My journey into world of graphs began unexpectedly, during an interview at LinkedIn. As session wrapped up, shown a visualization of network – a mesmerizing structure that told stories without a single word. Organizations I had been part of appeared clustered, like constellations against a dark canvas. What surprised me most was that this structure was not built using metadata LinkedIn held about my connection; rather, it emerged organically from relationships between nodes & edges.

  – Hành trình khám phá thế giới đồ thị của tôi bắt đầu một cách bất ngờ, trong một buổi phỏng vấn tại LinkedIn. Khi buổi phỏng vấn kết thúc, một hình ảnh trực quan về mạng lưới được trình chiếu - một cấu trúc mê hoặc kể những câu chuyện mà không cần một lời nào. Các tổ chức mà tôi từng là thành viên hiện ra như những chòm sao trên nền vải tối. Điều khiến tôi ngạc nhiên nhất là cấu trúc này không được xây dựng bằng siêu dữ liệu mà LinkedIn nắm giữ về kết nối của tôi; thay vào đó, nó xuất hiện một cách tự nhiên từ các mối quan hệ giữa các nút & cạnh.

  Years later, driven by curiosity, I recreated that visualization. I marveled once again at how underlying connections along could map out an intricate picture of my professional life. This deepened my appreciation for power inherent in graphs – a fascination that only grew when I joined Cloudera & encountered graph neural networks (GNNs). Their potential for solving complex problems was captivating, but diving into them was like trying to navigate an uncharted forest without a map. There were no comprehensive resources tailored for nonacademics; progress was slow, often cobbled together from fragments & trial & error.

  – Nhiều năm sau, nhờ sự tò mò, tôi đã tái hiện lại hình ảnh đó. Tôi lại một lần nữa kinh ngạc trước cách các kết nối cơ bản có thể vẽ nên một bức tranh phức tạp về cuộc sống nghề nghiệp của mình. Điều này càng làm tôi trân trọng hơn sức mạnh tiềm ẩn của đồ thị – một niềm đam mê chỉ lớn dần khi tôi gia nhập Cloudera & gặp gỡ mạng nơ-ron đồ thị (GNN). Tiềm năng giải quyết các vấn đề phức tạp của chúng thật hấp dẫn, nhưng việc đào sâu vào chúng cũng giống như cố gắng khám phá một khu rừng chưa được khám phá mà không có bản đồ. Không có tài nguyên toàn diện nào được thiết kế riêng cho những người không chuyên; tiến độ rất chậm, thường được chắp vá từ những mảnh & thử & sai.

  This book is guide I wish I had during those early days. It aims to provide a clear & accessible path for practitioners, enthusiasts, & anyone looking to understand & apply GNNs without wading through endless academic papers or fragmented online searches. Hop: it serves as a 1-stop resource to learn fundamentals & paves way for deeper exploration. Whether you

are here out of professional necessity, sheer curiosity, or same kind of amazement that 1st drew me in, invite to embark on this journey, bring potential of GNNs to life.

– Cuốn sách này chính là cẩm nang mà tôi ước mình đã có trong những ngày đầu ấy. Nó hướng đến việc cung cấp một lộ trình rõ ràng & dễ tiếp cận cho các chuyên gia, người đam mê, & bất kỳ ai muốn hiểu & áp dụng GNN mà không cần phải lội qua vô số bài báo học thuật hay tìm kiếm trực tuyến rời rạc. Hop: nó là một nguồn tài nguyên tổng hợp để học các kiến thức cơ bản & mở đường cho những khám phá sâu hơn. Cho dù bạn đến đây vì nhu cầu công việc, sự tò mò đơn thuần, hay cùng một sự ngạc nhiên như đã thu hút tôi lần đầu, hãy tham gia vào hành trình này, khai phá tiềm năng của GNN.

- About this book. GNNs in Action is a book designed for people to jump quickly into this new field & start building applications. At same time, try to strike a balance by including just enough critical theory to make this book as standalone as possible. Also fill in implementation details that may not be obvious or are left unexplained in currently available online tutorials & documents. In particular, information about new & emerging topics is very likely to be fragmented. This fragmentation adds friction when implementing & testing new technologies.

  – GNNs in Action là một cuốn sách được thiết kế để mọi người có thể nhanh chóng bước vào lĩnh vực mới này & bắt đầu xây dựng ứng dụng. Đồng thời, hãy cố gắng cân bằng bằng cách đưa vào vừa đủ lý thuyết quan trọng để cuốn sách này trở nên độc lập nhất có thể. Đồng thời, hãy bổ sung những chi tiết triển khai có thể chưa rõ ràng hoặc chưa được giải thích trong các hướng dẫn trực tuyến hiện có & tài liệu. Đặc biệt, thông tin về các chủ đề mới & đang nổi lên rất có thể sẽ bị phân mảnh. Sự phân mảnh này gây khó khăn khi triển khai & thử nghiệm các công nghệ mới.

  With GNNs in Action, offer a book that can reduce that friction by filling in gaps & answering key questions whose answers are likely scattered over internet or not covered at all. Done so in a way that emphasizes approachability rather than high rigor.

  – Với GNNs in Action, hãy cung cấp một cuốn sách có thể giảm thiểu sự khó khăn đó bằng cách lấp đầy những khoảng trống & trả lời những câu hỏi quan trọng mà câu trả lời có thể nằm rải rác trên internet hoặc chưa được đề cập đến. Hãy làm điều này theo cách nhấn mạnh tính dễ tiếp cận hơn là tính nghiêm ngặt cao.

  - Who should read this book. This book is designed for ML engineers & data scientists familiar with neural networks but new to graph learning. If have experience in OOP, find concepts particularly accessible & applicable.

    – Cuốn sách này được thiết kế dành cho các kỹ sư ML & nhà khoa học dữ liệu đã quen thuộc với mạng nơ-ron nhưng chưa quen với học đồ thị. Nếu có kinh nghiệm về OOP, hãy tìm các khái niệm đặc biệt dễ hiểu & áp dụng.

  - How this book is organized: A road map. In Part 1 of this book, provide a motivation for exploring GNNs, as well as cover fundamental concepts of graphs & graph-based ML. In Chap. 1, introduce concepts of graphs & graph ML, providing guidelines for their use & applications. Chap. 2 covers graph representations up to & including node embeddings. This will be 1st programmic exposure to GNNs, which are used to create such embeddings.

    – Trong Phần 1 của cuốn sách này, chúng tôi sẽ cung cấp động lực để khám phá GNN, cũng như đề cập đến các khái niệm cơ bản về đồ thị & Học máy dựa trên đồ thị. Trong Chương 1, chúng tôi sẽ giới thiệu các khái niệm về đồ thị & Học máy dựa trên đồ thị, đồng thời cung cấp hướng dẫn sử dụng & ứng dụng của chúng. Chương 2 sẽ đề cập đến các biểu diễn đồ thị, bao gồm cả nhúng nút. Đây sẽ là lần đầu tiên chúng tôi tiếp xúc với GNN, được sử dụng để tạo ra các nhúng như vậy.

    In part 2, core of book, introduce major types of GNNs, including graph convolutional networks (GCNs) & GraphSAGE in Chap. 3, graph attention networks (GATs) in Chap. 4, & graph autoencoders (GAEs) in Chap. 5. These methods are bread & butter for most GNN applications & also cover a range of other DL concepts e.g. convolution, attention, & autoencoders.

    – Trong phần 2, cốt lõi của cuốn sách, giới thiệu các loại GNN chính, bao gồm mạng tích chập đồ thị (GCN) & GraphSAGE trong Chương 3, mạng chú ý đồ thị (GAT) trong Chương 4, & bộ mã hóa tự động đồ thị (GAE) trong Chương 5. Các phương pháp này là nền tảng cho hầu hết các ứng dụng GNN & cũng bao gồm một loạt các khái niệm DL khác, e.g., tích chập, chú ý, & bộ mã hóa tự động.

    In part 3, look at more advanced topics. Describe GNNs for dynamic graphs (spatio-temporal GNNs) in Chap. 6 & give methods to train GNNs at scale in Chap. 7. Finally, end with some consideration for project & system planning for graph learning projects in Chap. 8.

    – Trong phần 3, hãy xem xét các chủ đề nâng cao hơn. Mô tả GNN cho đồ thị động (GNN không gian-thời gian) trong Chương 6 & đưa ra các phương pháp huấn luyện GNN ở quy mô lớn trong Chương 7. Cuối cùng, kết thúc bằng một số cân nhắc về dự án & lập kế hoạch hệ thống cho các dự án học đồ thị trong Chương 8.

  - About code. Python is coding language of choice throughout this book. There are now several GNN libraries in Python ecosystem, including PyTorch Geometric (PyG), Deep Graph Library (DGL), GraphScope, & Jraph. Focus on PyG, which is 1 of most popular & easy-to-use frameworks, written on top of PyTorch. Want this book to be approachable by an audience with a wide set of hardware constraints, so with exception of some individual sects & Chap. 7 on scalability, distributed systems & GPU systems aren't required, although they can be used for some of coded examples.

    – Python là ngôn ngữ lập trình được lựa chọn trong suốt cuốn sách này. Hiện nay, hệ sinh thái Python đã có một số thư viện GNN, bao gồm PyTorch Geometric (PyG), Thư viện Đồ thị Sâu (DGL), GraphScope, Jraph. Tập trung vào PyG, một trong những framework phổ biến nhất, dễ sử dụng, được viết trên nền tảng PyTorch. Tôi muốn cuốn sách này dễ tiếp cận với những độc giả có nhiều hạn chế về phần cứng, vì vậy, ngoại trừ một số điểm riêng biệt trong Chương 7 về khả năng mở rộng, các hệ thống phân tán và GPU không bắt buộc, mặc dù chúng có thể được sử dụng cho một số ví dụ được mã hóa.

Book provides a survey of most relevant implementations of GNNs, including graph convolutional networks (GCNs), graph autoencoders (GAEs), graph attention networks (GATs), & graph long short-term memory (LSTM). Aim: cover GNN tasks mentioned earlier. In addition, touch on different types of graphs, including knowledge graphs.

– Sách cung cấp một bản tổng quan về các triển khai GNN phổ biến nhất, bao gồm mạng tích chập đồ thị (GCN), bộ mã hóa tự động đồ thị (GAE), mạng chú ý đồ thị (GAT), bộ nhớ dài hạn đồ thị (LSTM). Mục tiêu: bao quát các nhiệm vụ GNN đã đề cập trước đó. Ngoài ra, đề cập đến các loại đồ thị khác nhau, bao gồm đồ thị tri thức.

This book contains many examples of source code both in numbered listings & in line with normal text. In both case, source code is formatted in a `fixed-width font like this` to separate it from ordinary text. Sometimes code is also **in bold** to highlight code

PART 1: 1ST STEPS. Graphs are 1 of most versatile & powerful ways to represent complex, interconnected data. This 1st part introduces fundamental concepts of graph theory, explaining what graphs are, why they matter as a data type, & how their structure captures relationships that traditional data formats miss. Explore building blocks of graphs & different graph types.

– Đồ thị là một trong những phương pháp linh hoạt và mạnh mẽ nhất để biểu diễn dữ liệu phức tạp, có liên kết với nhau. Phần 1 này giới thiệu các khái niệm cơ bản của lý thuyết đồ thị, giải thích đồ thị là gì, tại sao chúng quan trọng như một kiểu dữ liệu, và cách cấu trúc của chúng nắm bắt các mối quan hệ mà các định dạng dữ liệu truyền thống bỏ sót. Khám phá các khối xây dựng của đồ thị và các kiểu đồ thị khác nhau.

Explore fundamental concepts about GNNs, beginning with what they are & how they differ from traditional neural networks. With this foundation, study graph embeddings, uncovering how to represent graphs in a way that makes them useful for ML. These concepts set stage for mastering GNNs & their transformative capabilities in later chaps. By end of this book, have a solid understanding of basics, preparing you to dive deeper into mechanics of GNNs.

– Khám phá các khái niệm cơ bản về GNN, bắt đầu với bản chất của chúng & sự khác biệt so với mạng nơ-ron truyền thống. Với nền tảng này, hãy nghiên cứu nhúng đồ thị, khám phá cách biểu diễn đồ thị sao cho hữu ích cho ML. Những khái niệm này đặt nền tảng cho việc nắm vững GNN & khả năng biến đổi của chúng trong các chương sau. Khi đọc xong cuốn sách này, bạn sẽ có được kiến thức cơ bản vững chắc, sẵn sàng cho việc tìm hiểu sâu hơn về cơ chế hoạt động của GNN.

- 1. Discovering graph neural networks. Covers:
  - Defining graphs & GNNs
  - Understanding why people are excited about GNNs
  - Recognizing when to use GNNs
  - Taking a big picture look at solving a problem with a GNN

For data practitioners, fields of ML & DS initially excite us because of potential to draw nonintuitive & useful insights from data. In particular, insights from ML & DL promise to enhance our understanding of world. For working engineer, these tools promise to deliver business value in unprecedented ways.

– Đối với các chuyên gia dữ liệu, lĩnh vực Học máy & Phân tích dữ liệu (ML & DS) ban đầu khiến chúng ta hào hứng vì tiềm năng rút ra những hiểu biết phi trực quan & hữu ích từ dữ liệu. Đặc biệt, những hiểu biết từ Học máy & Phân tích dữ liệu (ML & DL) hứa hẹn sẽ nâng cao hiểu biết của chúng ta về thế giới. Đối với các kỹ sư đang làm việc, những công cụ này hứa hẹn mang lại giá trị kinh doanh theo những cách chưa từng có.

Experience deviates from this ideal. Real-world data is usually messy, dirty, & biased. Furthermore, statistical methods & learning systems come with their own set of limitations. An essential role of practitoners: comprehend these limitations & bridge gap between real data & a feasible solution. E.g., may want to predict fraudulent activity in a bank, but 1st need to make sure that our training data has been correctly labeled. Even more importantly, need to check that our models won't incorrectly assign fraudulent activity to normal behaviors, possibly due to some hidden confounders in data.

– Kinh nghiệm thực tế thường khác xa lý tưởng này. Dữ liệu thực tế thường lộn xộn, bẩn thỉu, & thiên vị. Hơn nữa, các phương pháp thống kê & hệ thống học tập cũng có những hạn chế riêng. Một vai trò thiết yếu của người thực hành: hiểu rõ những hạn chế này & thu hẹp khoảng cách giữa dữ liệu thực tế & một giải pháp khả thi. Ví dụ: có thể muốn dự đoán hoạt động gian lận trong một ngân hàng, nhưng trước tiên cần đảm bảo rằng dữ liệu đào tạo của chúng ta đã được dán nhãn chính xác. Quan trọng hơn nữa, cần kiểm tra xem các mô hình của chúng ta có gán sai hoạt động gian lận cho các hành vi bình thường hay không, có thể do một số yếu tố gây nhiễu tiềm ẩn trong dữ liệu.

For graph data, until recently, bridging this gap has been particularly challenging. Graphs are a data structure that is rich with information & especially adept at capturing intricacies of data where relationships play a crucial role. Graphs are omnipresent, with relationship data appearing in different forms e.g. atoms in molecules (nature), social networks (society), & even models connection of web pages on internet (technology). Important to note: term *relational* here does not refer to *relational databases*, but rather to data where relationships are of significance.

– Đối với dữ liệu đồ thị, cho đến gần đây, việc thu hẹp khoảng cách này đặc biệt khó khăn. Đồ thị là một cấu trúc dữ liệu giàu thông tin & đặc biệt khéo léo trong việc nắm bắt những dữ liệu phức tạp, nơi các mối quan hệ đóng vai trò then chốt. Đồ thị hiện diện ở khắp mọi nơi, với dữ liệu mối quan hệ xuất hiện dưới nhiều dạng khác nhau, ví dụ: nguyên tử trong phân tử (tự nhiên), mạng xã hội (xã hội), & thậm chí cả mô hình kết nối các trang web trên internet (công nghệ). Điều quan trọng cần

lưu ý: thuật ngữ *relational* ở đây không đề cập đến *relational databases*, mà là dữ liệu trong đó các mối quan hệ có ý nghĩa quan trọng.

Previously, if you wanted to incorporate relational features from a graph into a DL model, it had to be done in an indirect way, with different models used to process, analyze, & then use graph data. These separate models often couldn't be easily scaled & had trouble taking into account all node & edge properties of graph data. To make best use of this rich & ubiquitous data type for ML, needed a specialized ML technique specifically designed for distinct qualities of graphs & relational data. This is gap that GNNs fill.

– Trước đây, nếu muốn tích hợp các đặc điểm quan hệ từ đồ thị vào mô hình DL, việc này phải được thực hiện gián tiếp, sử dụng các mô hình khác nhau để xử lý, phân tích, & sau đó sử dụng dữ liệu đồ thị. Các mô hình riêng biệt này thường không dễ dàng mở rộng & gặp khó khăn trong việc tính đến tất cả các thuộc tính nút & cạnh của dữ liệu đồ thị. Để tận dụng tối đa kiểu dữ liệu phong phú & phổ biến này cho ML, cần có một kỹ thuật ML chuyên biệt được thiết kế riêng cho các đặc tính riêng biệt của đồ thị & dữ liệu quan hệ. Đây chính là khoảng trống mà GNN lấp đầy.

DP field often contains a lot of hype around new technologies & methods. However, GNNs are widely recognized as a genuine leap forward for graph-based learning [2]. This does not mean: GNNs are a silver bullet. Careful comparisons should be done between predictive results derived from GNNs & other ML & DL methods.

– Lĩnh vực DP thường chứa đựng nhiều thông tin cường điệu về các công nghệ & phương pháp mới. Tuy nhiên, GNN được công nhận rộng rãi là một bước tiến thực sự cho học tập dựa trên đồ thị [2]. Điều này không có nghĩa là: GNN là giải pháp hoàn hảo. Cần so sánh cẩn thận giữa các kết quả dự đoán thu được từ GNN & các phương pháp ML & DL khác.

Key thing to remember: if your DS problem involves data that can be structured as a graph – i.e., data is connected or relational – then GNNs could offer a valuable approach, even if you weren't aware that sth was missing in your approach. GNNs can be designed to handle very large data, to scale, to adapt to graphs of different sizes & shapes. This can make working with relationship-centric data easier & more efficient, as well as yield richer results.

– Điều quan trọng cần nhớ: nếu bài toán DS của bạn liên quan đến dữ liệu có thể được cấu trúc dưới dạng đồ thị - tức là dữ liệu được kết nối hoặc quan hệ - thì GNN có thể cung cấp một phương pháp tiếp cận hữu ích, ngay cả khi bạn không nhận ra rằng phương pháp của mình còn thiếu điều gì đó. GNN có thể được thiết kế để xử lý dữ liệu rất lớn, có khả năng mở rộng, thích ứng với các đồ thị có kích thước & hình dạng khác nhau. Điều này có thể giúp việc xử lý dữ liệu tập trung vào mối quan hệ dễ dàng & hiệu quả hơn, cũng như mang lại kết quả phong phú hơn.

Standout advantages of GNNs are why data scientists & engineers are increasingly recognizing importance of mastering them. GNNs have the ability to unveil unique insights from relational dat – from identifying new drug candidates to optimizing ETA prediction accuracy in your Google Maps app – acting as a catalyst for discovery & innovation, & empowering professionals to push boundaries of conventional data analysis. Their diverse applicability spans various fields, offering professionals a versatile tool that is as relevant in e-commerce (e.g., recommendation engines) as it is in bioinformatics (e.g., drug toxicity prediction). Proficiency in GNNs equips data professionals with a multifaceted tool for enhanced, accurate, & innovative data analysis of graphs.

– Những lợi thế nổi bật của GNN là lý do tại sao các nhà khoa học dữ liệu & kỹ sư ngày càng nhận thức được tầm quan trọng của việc thành thạo chúng. GNN có khả năng khám phá những hiểu biết độc đáo từ dữ liệu quan hệ – từ việc xác định các ứng cử viên thuốc mới đến tối ưu hóa độ chính xác dự đoán ETA trong ứng dụng Google Maps – đóng vai trò là chất xúc tác cho khám phá & đổi mới, & trao quyền cho các chuyên gia vượt qua các giới hạn của phân tích dữ liệu thông thường. Khả năng ứng dụng đa dạng của chúng trải rộng trên nhiều lĩnh vực, mang đến cho các chuyên gia một công cụ đa năng, vừa phù hợp trong thương mại điện tử (ví dụ: công cụ đề xuất) vừa phù hợp trong tin sinh học (ví dụ: dự đoán độc tính của thuốc). Thành thạo GNN trang bị cho các chuyên gia dữ liệu một công cụ đa năng để phân tích dữ liệu đồ thị chính xác, & sáng tạo.

For all these reasons, GNNs are now popular choice for recommender engines, analyzing social networks, detecting fraud, understanding how biomolecules behave, & many other practical examples.

– Vì tất cả những lý do này, GNN hiện là lựa chọn phổ biến cho các công cụ đề xuất, phân tích mạng xã hội, phát hiện gian lận, hiểu cách các phân tử sinh học hoạt động, & nhiều ví dụ thực tế khác.

- 1.1. Goals of this book. GNNs in Action is aimed at practitioners who want to begin to deploy GNNs to solve real problems. This could be a ML engineer not familiar with graph data structures, a data scientist who hasn't yet tried GNNs, or even a software engineer who may be unfamiliar with either. Throughout this book, cover topics from basics of graphs all way to more complex GNN models. Build up architecture of a GNN, step-by-step. This includes overall architecture of a GNN & critical aspect of message passing. Then go on to add different features & extensions to these basic aspects, e.g. introducing convolution & sampling, attention mechanisms, a generative model, & operating on dynamic graphs. When building our GNNs, work with Python & use some standard libraries. GNNs libraries are either standalone or use TensorFlow or PyTorch as a backend. In this text, focus will be on PyTorch Geometric (PyG). Other popular libraries include Deep Graph Library (DGL, a standalone library) & SPektral (which uses Keras & TensorFlow as a backend). There is also Jraph for JAX users.

  – GNNs in Action hướng đến những người thực hành muốn bắt đầu triển khai GNN để giải quyết các vấn đề thực tế. Họ có thể là một kỹ sư ML chưa quen thuộc với cấu trúc dữ liệu đồ thị, một nhà khoa học dữ liệu chưa từng thử GNN, hoặc thậm chí là một kỹ sư phần mềm chưa quen thuộc với cả hai. Xuyên suốt cuốn sách này, chúng tôi sẽ đề cập đến các chủ đề từ kiến thức cơ bản về đồ thị cho đến các mô hình GNN phức tạp hơn. Xây dựng kiến trúc của GNN, từng bước một. Điều này bao gồm kiến trúc tổng thể của GNN & khía cạnh quan trọng của việc truyền thông điệp. Sau đó, tiếp tục thêm các tính năng khác nhau & phần mở rộng cho các khía cạnh cơ bản này, ví dụ: giới thiệu tích chập & lấy mẫu, cơ chế chú

ý, mô hình sinh, & hoạt động trên đồ thị động. Khi xây dựng GNN, hãy làm việc với Python & sử dụng một số thư viện chuẩn. Thư viện GNN có thể độc lập hoặc sử dụng TensorFlow hoặc PyTorch làm nền tảng. Trong văn bản này, trọng tâm sẽ là PyTorch Geometric (PyG). Các thư viện phổ biến khác bao gồm Thư viện Deep Graph (DGL, một thư viện độc lập) & SPektral (sử dụng Keras & TensorFlow làm nền tảng). Ngoài ra còn có Jraph dành cho người dùng JAX.

Our aim throughout this book is to enable you to:

1. access suitability of a GNN solution for your problem.
2. understand when traditional neural networks won't perform as well as a GNN for graph structured data & when GNNs may not be the best tool for tabular data.
3. design & implement a GNN architecture to solve problems specific to you.
4. make clear limitations of GNNs.

This book is weighted toward implementation using programming. Also devote some time on essential theory & concepts, so that techniques covered can be sufficiently understood. These are covered in an "Under Hood" sect at end of most chaps to separate technical reasons from actual implementation. There are many different models & packages that build on key concepts introduced in this book. So, this book should not be seen as a comprehensive review of all GNns methods & models, which could run to several thousands of pages, but rather starting point for curious & eager-to-learn practitioner.

– Mục tiêu của chúng tôi trong suốt cuốn sách này là giúp bạn:

1. đánh giá tính phù hợp của giải pháp GNN cho vấn đề của bạn.
2. hiểu khi nào mạng nơ-ron truyền thống không hoạt động tốt bằng GNN đối với dữ liệu có cấu trúc đồ thị & khi nào GNN có thể không phải là công cụ tốt nhất cho dữ liệu dạng bảng.
3. thiết kế & triển khai kiến trúc GNN để giải quyết các vấn đề cụ thể của bạn.
4. làm rõ những hạn chế của GNN.

Cuốn sách này thiên về việc triển khai bằng lập trình. Đồng thời dành thời gian cho các lý thuyết & khái niệm thiết yếu, để các kỹ thuật được đề cập có thể được hiểu đầy đủ. Những điều này được trình bày trong phần "Under Hood" ở cuối hầu hết các chương để phân biệt lý do kỹ thuật với việc triển khai thực tế. Có rất nhiều mô hình & gói khác nhau được xây dựng dựa trên các khái niệm chính được giới thiệu trong cuốn sách này. Vì vậy, cuốn sách này không nên được coi là một bài tổng quan toàn diện về tất cả các phương pháp & mô hình GNN, có thể dài tới hàng nghìn trang, mà nên là điểm khởi đầu cho những người thực hành & ham học hỏi.

Book is divided into 3 parts. Part 1 covers basics of GNNs, especially ways in which they differ from other neural networks, e.g. *message passing & embeddings*, which have specific meaning for GNNs. Part 2, heart of book, goes over models themselves, where we cover a handful of key model types. Then, in part 3, go into more detail with some of harder models & concepts, including how to scale graphs & deal with temporal data.

– Sách được chia thành 3 phần. Phần 1 trình bày những kiến thức cơ bản về GNN, đặc biệt là những điểm khác biệt giữa chúng với các mạng nơ-ron khác, ví dụ như truyền thông điệp & nhúng, vốn có ý nghĩa riêng đối với GNN. Phần 2, trọng tâm của sách, sẽ đề cập đến bản thân các mô hình, trong đó chúng ta sẽ tìm hiểu một số loại mô hình chính. Sau đó, trong phần 3, chúng ta sẽ đi sâu hơn vào một số mô hình & khái niệm khó hơn, bao gồm cách chia tỷ lệ đồ thị & xử lý dữ liệu thời gian.

GNNs in Action is designed for people to jump quickly into this new field & start building applications. Aim for this book: reduce friction of implementing new technologies by filling in gaps & answering key development questions whose answers may not be easy to find or may not be covered elsewhere at all. Each method is introduced through an example application so you can understand how GNNs are applied in practice.

– GNNs in Action được thiết kế để mọi người nhanh chóng tiếp cận lĩnh vực mới này & bắt đầu xây dựng ứng dụng. Mục tiêu của cuốn sách này: giảm thiểu sự cản trở khi triển khai các công nghệ mới bằng cách lấp đầy những khoảng trống & trả lời những câu hỏi phát triển quan trọng mà câu trả lời có thể không dễ tìm hoặc chưa được đề cập ở bất kỳ nơi nào khác. Mỗi phương pháp được giới thiệu thông qua một ứng dụng ví dụ để bạn có thể hiểu cách GNN được áp dụng trong thực tế.

∗ 1.1.1. Catching up on graph fundamentals. Do need to understand basics of graphs before you can understand GNNs. Goal for this book is to teach GNNs to DL practitioners & builders for traditional neural networks who may not know much about graphs. At same time, also recognize: readers of this book may vary enormously in their knowledge of graphs. How to address these differences & make sure everyone has what they need to make the most of this book? In this chap, provide an introduction to fundamental graph concepts that are most essential to understanding GNNs.

– Bạn cần nắm vững những kiến thức cơ bản về đồ thị trước khi có thể hiểu về GNN. Mục tiêu của cuốn sách này là hướng dẫn GNN cho những người thực hành DL & những người xây dựng mạng nơ-ron truyền thống, những người có thể chưa biết nhiều về đồ thị. Đồng thời, cũng cần lưu ý: kiến thức về đồ thị của độc giả có thể rất khác nhau. Làm thế nào để giải quyết những khác biệt này & đảm bảo mọi người đều có những kiến thức cần thiết để tận dụng tối đa cuốn sách này? Trong chương này, chúng tôi sẽ giới thiệu các khái niệm cơ bản về đồ thị, những khái niệm thiết yếu nhất để hiểu về GNN.

After refresher on key concepts in graphs & graph learning, look into some case studies in several fields where GNNs are being successfully applied. Then, break down those specific cases to see what makes a good case for using a GNN, as well as how to know if you have a GNN problem on your hands. At end of chap, introduce mechanics of GNNs, barebone skeleton that the rest of book will add to.

– Sau khi ôn lại các khái niệm chính về đồ thị & học đồ thị, hãy xem xét một số nghiên cứu điển hình trong một số lĩnh vực mà GNN đang được ứng dụng thành công. Sau đó, hãy phân tích các trường hợp cụ thể đó để xem đâu là lý do tốt để sử dụng GNN, cũng như cách nhận biết liệu bạn có đang gặp vấn đề về GNN hay không. Cuối chương, hãy giới thiệu cơ chế hoạt động của GNN, bộ khung xương cốt mà phần còn lại của cuốn sách sẽ bổ sung.

- 1.2. Graph-based learning. This section defines graphs, graph-based learning, & some fundamentals of GNNs, including basic structure of a graph & a taxonomy of different types of graphs. Then, review graph-based learning, putting GNNs in context with other learning methods. Finally, explain value of graphs, ending with an example of data derived from Titanic dataset.

  – Học tập dựa trên đồ thị. Phần này định nghĩa đồ thị, học tập dựa trên đồ thị, & một số kiến thức cơ bản về mạng nơ-ron nhân tạo (GNN), bao gồm cấu trúc cơ bản của đồ thị & phân loại các loại đồ thị khác nhau. Sau đó, xem xét lại học tập dựa trên đồ thị, đặt GNN vào bối cảnh của các phương pháp học tập khác. Cuối cùng, giải thích giá trị của đồ thị, kết thúc bằng một ví dụ về dữ liệu được lấy từ tập dữ liệu Titanic.

  - 1.2.1. What are graphs? Graphs are data structures with elements, expressed as *nodes or vertices*, & relationships between elements, expressed as *edges or links*. All nodes in graph will have additional *feature data*. This is node-specific data, relating to things e.g. names or ages of individuals in a social network. Links are key to power of relational data, as they allow us to learn more about system, give new tools for analyzing data, & predict new properties from it. This is in contrast to tabular data e.g. a database table, dataframe, or spreadsheet, where data is fixed in rows & columns.

    – Đồ thị là cấu trúc dữ liệu với các phần tử, được biểu diễn dưới dạng *nút hoặc đỉnh*, & mối quan hệ giữa các phần tử, được biểu diễn dưới dạng *cạnh hoặc liên kết*. Tất cả các nút trong đồ thị sẽ có thêm *dữ liệu đặc trưng*. Đây là dữ liệu cụ thể của từng nút, liên quan đến các thông tin như tên hoặc tuổi của các cá nhân trong mạng xã hội. Liên kết là chìa khóa cho sức mạnh của dữ liệu quan hệ, vì chúng cho phép chúng ta tìm hiểu thêm về hệ thống, cung cấp các công cụ mới để phân tích dữ liệu và & dự đoán các thuộc tính mới từ dữ liệu đó. Điều này trái ngược với dữ liệu dạng bảng, ví dụ: bảng cơ sở dữ liệu, khung dữ liệu hoặc bảng tính, trong đó dữ liệu được cố định theo hàng & cột.

    To describe & learn from edges between nodes, we need a way to write them down. This can be explicitly, quickly, can see describing things in this way becomes unwieldy & might be repeating redundant information. Luckily, there are many mathematical formalisms for describing relations in graphs. 1 of most common: describe *adjacency matrix*. Notice: adjacency matrix is symmetric across diagonal & all values are 1s or 0s. Adjacency matrix of a graph is an important concept that makes it easy to observe all connections of a graph in a single table. Here assumed: there is no directionability in our graphs, i.e., if 0 is connected to 1, then 1 is also connected to 0. This is known as an *undirected graph*. Undirected graphs can be easily inferred from an adjacency matrix because, in this case, matrix is symmetric across diagonal, upper-right triangle is reflected onto bottom-left.

    – Để mô tả & học từ các cạnh giữa các nút, chúng ta cần một cách để viết chúng ra. Điều này có thể rõ ràng, nhanh chóng, có thể thấy việc mô tả mọi thứ theo cách này trở nên cồng kềnh & có thể lặp lại thông tin dư thừa. May mắn thay, có nhiều công thức toán học để mô tả các mối quan hệ trong đồ thị. 1 trong những công thức phổ biến nhất: mô tả *adjacency matrix*. Lưu ý: ma trận kề là đối xứng qua đường chéo & tất cả các giá trị là 1 hoặc 0. Ma trận kề của đồ thị là một khái niệm quan trọng giúp dễ dàng quan sát tất cả các kết nối của đồ thị trong một bảng duy nhất. Ở đây giả sử: không có tính định hướng trong đồ thị của chúng ta, tức là nếu 0 được kết nối với 1, thì 1 cũng được kết nối với 0. Đây được gọi là *undirected graph*. Đồ thị vô hướng có thể dễ dàng suy ra từ ma trận kề vì, trong trường hợp này, ma trận đối xứng qua đường chéo, tam giác trên cùng bên phải được phản chiếu xuống dưới cùng bên trái.

    Also assume: all relations between nodes are identical. If we wanted relation of nodes B-E to mean more than relation of nodes B-A, then we could increase weight of this edge. This translates to increasing value in adjacency matrix, making entry for B-A edge equal to 10 instead of 1, e.g.

    – Cũng giả sử: tất cả các mối quan hệ giữa các nút đều giống hệt nhau. Nếu chúng ta muốn mối quan hệ giữa các nút B-E có ý nghĩa hơn mối quan hệ giữa các nút B-A, thì chúng ta có thể tăng trọng số của cạnh này. Điều này tương đương với việc tăng giá trị trong ma trận kề, ví dụ, nhập giá trị cho cạnh B-A bằng 10 thay vì 1.

    Graphs where all relations are of equal importance are known as *unweighted graphs* & can also be easily observed from adjacency matrix because all graph entries are either 1s or 0s. Graphs where edges have multiple values are known as *weighted*.

    – Đồ thị mà tất cả các mối quan hệ đều có tầm quan trọng như nhau được gọi là *unweighted graphs* & cũng có thể dễ dàng quan sát được từ ma trận kề vì tất cả các mục đồ thị đều là 1 hoặc 0. Đồ thị mà các cạnh có nhiều giá trị được gọi là *weighted*.

    If any of nodes in graph do not have an edge that connects to itself, then nodes will also have 0s at their own value in adjacency matrix (0s along diagonal), i.e., a graph does not have self-loops. A *self-loop* occurs when a node has an edge that connects to that same node. To add a self-loop, we just make value for that node nonzero at its position in diagonal.

    – Nếu bất kỳ nút nào trong đồ thị không có cạnh nối với chính nó, thì các nút cũng sẽ có giá trị 0 tại chính nó trong ma trận kề (các giá trị 0 dọc theo đường chéo), tức là đồ thị không có vòng lặp tự thân. Một vòng lặp tự thân xảy ra khi một nút có cạnh nối với chính nút đó. Để thêm một vòng lặp tự thân, chúng ta chỉ cần gán giá trị khác không cho nút đó tại vị trí của nó trên đường chéo.

    In practice, an adjacency matrix is only 1 of many ways to describe relations in a graph. Others include adjacency lists, edge lists, or an incidence matrix. Understanding these types of data structures well is vital to graph-based learning. If you are unfamiliar with these terms, or need a refresher, recommend looking through appendix A, which has additional details & explanations.

    – Trên thực tế, ma trận kề chỉ là một trong nhiều cách để mô tả các mối quan hệ trong đồ thị. Các cách khác bao gồm danh sách kề, danh sách cạnh, hoặc ma trận liên quan. Việc hiểu rõ các loại cấu trúc dữ liệu này rất quan trọng đối với

việc học tập dựa trên đồ thị. Nếu bạn chưa quen với các thuật ngữ này hoặc cần ôn tập lại, hãy xem Phụ lục A, trong đó có thêm chi tiết & giải thích.

∗ 1.2.2. Different types of graphs. Understanding many different types of graphs can help us work out what methods to use to analyze & transform graph, & what ML methods to apply. In following, give a very quick overview of some of most common properties for graphs to have.
– Hiểu biết về nhiều loại đồ thị khác nhau có thể giúp chúng ta tìm ra phương pháp nào cần sử dụng để phân tích & biến đổi đồ thị, & áp dụng phương pháp ML nào. Sau đây, chúng tôi sẽ giới thiệu sơ lược về một số thuộc tính phổ biến nhất của đồ thị.

· Homogeneous & Heterogeneous Graphs. Most basic graphs are *homogeneous graphs*, which are made up of 1 type of node & 1 type of edge. Consider a homogeneous graph that describes a recruitment network. In this type of graph, nodes would represent job candidates, & edges would represent relationships between candidates.
– Đồ thị Đồng nhất & Đồ thị Không Đồng nhất. Hầu hết các đồ thị cơ bản là *đồ thị đồng nhất*, được tạo thành từ 1 loại nút & 1 loại cạnh. Xem xét một đồ thị đồng nhất mô tả một mạng lưới tuyển dụng. Trong loại đồ thị này, các nút sẽ đại diện cho các ứng viên xin việc, & cạnh sẽ đại diện cho mối quan hệ giữa các ứng viên.
If want to expand power of our graph to describe our recruitment network, could give it more types of nodes & edges, making it a *heterogeneous graphs*. With this expansion, some nodes may be candidates & others may be companies. Edges could now consist of relationships between candidates & current or past employment of job candidates at companies. See Fig. 1.2: A homogeneous graph & a heterogeneous graph. Here, shade of a node or edge represents its type or class. For homogeneous graph, all nodes are of same type, & all edges are of same type. For heterogeneous graph, nodes & edges have multiple types for a comparison of a homogeneous graph (all nodes or edges have same shade) with a heterogeneous graph (nodes & edges have a variety of shades).
– Nếu muốn mở rộng sức mạnh của đồ thị để mô tả mạng lưới tuyển dụng, có thể cung cấp cho nó nhiều loại nút & cạnh hơn, biến nó thành *đồ thị không đồng nhất*. Với sự mở rộng này, một số nút có thể là ứng viên & những nút khác có thể là công ty. Các cạnh bây giờ có thể bao gồm các mối quan hệ giữa ứng viên & việc làm hiện tại hoặc trước đây của ứng viên tại các công ty. Xem Hình 1.2: Đồ thị đồng nhất & đồ thị không đồng nhất. Ở đây, sắc thái của một nút hoặc cạnh biểu thị loại hoặc lớp của nó. Đối với đồ thị đồng nhất, tất cả các nút đều cùng loại, & tất cả các cạnh đều cùng loại. Đối với đồ thị không đồng nhất, các nút & cạnh có nhiều loại để so sánh đồ thị đồng nhất (tất cả các nút hoặc cạnh có cùng sắc thái) với đồ thị không đồng nhất (các nút & cạnh có nhiều sắc thái khác nhau).

· Bipartite graphs. Similar to heterogeneous graphs, *bipartite graphs* also can be separated or partitioned into different subsets. However, bipartite graphs (Fig. 1.3: A bipartite graph. There are 2 types of nodes (2 shades of circles). In a bipartite graph, nodes cannot be connected to nodes of same type. This is also an example of a heterogeneous graph.) have a very specific network structure s.t. nodes in each subset connect to nodes outside of their subset & not inside. Later, discuss recommendation system & Pinterest graph. This graph is bipartite because 1 set of nodes (pins) connects another set of nodes (boards) but not to nodes within their set (pins).
– Đồ thị hai phần. Tương tự như đồ thị không đồng nhất, *Đồ thị 2 phần* cũng có thể được tách hoặc phân vùng thành các tập con khác nhau. Tuy nhiên, đồ thị hai phần (Hình 1.3: Đồ thị hai phần. Có 2 loại nút (2 sắc thái của hình tròn). Trong đồ thị hai phần, các nút không thể được kết nối với các nút cùng loại. Đây cũng là một ví dụ về đồ thị không đồng nhất.) có cấu trúc mạng rất cụ thể, ví dụ: các nút trong mỗi tập con kết nối với các nút bên ngoài tập con của chúng & không kết nối với các nút bên trong. Sau đó, hãy thảo luận về hệ thống đề xuất & đồ thị Pinterest. Đồ thị này là hai phần vì 1 tập hợp các nút (ghim) kết nối với một tập hợp các nút khác (bảng) nhưng không kết nối với các nút trong tập hợp của chúng (ghim).

· Cyclic graphs, acyclic graphs, & directed acyclic graphs. A graph is *cyclic* if it allows you to start at a node, travel along its edge, & return to starting node without retracing any steps, creating a circular path within graph. In contrast, in an *acyclic* graph, no matter which path you take from any starting node, you cannot return to starting point without backtracking. These graphs, as shown in Fig. 1.4: A cyclic graph, an acyclic graph, & a DAG. In cyclic graph, cycle is shown by arrows (directed edges) connecting nodes A-E-D-C-B-A. Note 2 nodes, F & G are part of graph, but not part of its defining cycle. Acyclic graph is composed of undirected edges, & no cycle is possible. In DAG, all directed edges flow in 1 direction, from A to F., often resemble tree-like structures or paths that do not loop back on themselves.
– Đồ thị tuần hoàn, đồ thị phi chu trình, & đồ thị phi chu trình có hướng. Một đồ thị là *tuần hoàn* nếu nó cho phép bạn bắt đầu tại một nút, di chuyển dọc theo cạnh của nó, & quay lại nút bắt đầu mà không cần quay lại bất kỳ bước nào, tạo ra một đường tròn trong đồ thị. Ngược lại, trong đồ thị *phi chu trình*, bất kể bạn đi theo đường nào từ bất kỳ nút bắt đầu nào, bạn không thể quay lại điểm bắt đầu mà không quay lại. Các đồ thị này, như được thể hiện trong Hình 1.4: Đồ thị tuần hoàn, đồ thị phi chu trình, & DAG. Trong đồ thị tuần hoàn, chu trình được biểu diễn bằng các mũi tên (cạnh có hướng) nối các nút A-E-D-C-B-A. Lưu ý 2 nút, F & G là một phần của đồ thị, nhưng không phải là một phần của chu trình xác định của nó. Đồ thị phi chu trình bao gồm các cạnh không có hướng, & không thể có chu trình. Trong DAG, tất cả các cạnh có hướng đều chảy theo 1 hướng, từ A đến F., thường giống các cấu trúc giống cây hoặc các đường dẫn không vòng lại với chính chúng.
While both cyclic & acyclic graphs can be either undirected or directed, a *directed acyclic graph (DAG)* is a specific type of acyclic graph that is exclusively directed. In a DAG, all edges have a direction, & no cycles are allowed. DAGs represent 1-way relationships where we can't follow arrows & end up back at starting point. This characteristic makes DAGs essential in causal analysis, as they reflect causal structures where causality is assumed to be undirectional. E.g., A can cause B, but B can't simultaneously cause A. This unidirectional nature aligns perfectly with structure of DAGs, making them ideal for modeling workflow processes, dependency chains, & causal relationships in various fields.

– Trong khi cả đồ thị tuần hoàn và đồ thị phi tuần hoàn đều có thể vô hướng hoặc có hướng, *đồ thị phi tuần hoàn có hướng (DAG)* là một loại đồ thị phi tuần hoàn cụ thể chỉ có hướng. Trong DAG, tất cả các cạnh đều có hướng, & không cho phép chu trình. DAG biểu diễn các mối quan hệ một chiều, trong đó chúng ta không thể theo mũi tên & kết thúc ở điểm xuất phát. Đặc điểm này làm cho DAG trở nên thiết yếu trong phân tích nhân quả, vì chúng phản ánh các cấu trúc nhân quả trong đó quan hệ nhân quả được coi là vô hướng. Ví dụ: A có thể gây ra B, nhưng B không thể đồng thời gây ra A. Bản chất đơn hướng này hoàn toàn phù hợp với cấu trúc của DAG, khiến chúng trở nên lý tưởng để mô hình hóa các quy trình công việc, chuỗi phụ thuộc, & các mối quan hệ nhân quả trong nhiều lĩnh vực khác nhau.

· Knowledge graphs. A *knowledge graph* is a specialized type of heterogeneous graph that represents data with enriched semantic meaning, capturing not only relationships between different entities but also context & nature of these relationships. Unlike conventional graphs, which primarily emphasize structure & connectivity, a knowledge graph incorporates metadata & follow specific schemas to provide deeper contextual information. This allows for advanced reasoning & querying capabilities, e.g. identifying patterns, uncovering specific types of connections, or inferring new relationships.
– Đồ thị tri thức. Đồ thị tri thức là một loại đồ thị không đồng nhất chuyên biệt, biểu diễn dữ liệu với ý nghĩa ngữ nghĩa phong phú, không chỉ nắm bắt mối quan hệ giữa các thực thể khác nhau mà còn cả bối cảnh & bản chất của các mối quan hệ này. Không giống như đồ thị thông thường, chủ yếu nhấn mạnh vào cấu trúc & kết nối, đồ thị tri thức kết hợp siêu dữ liệu & tuân theo các lược đồ cụ thể để cung cấp thông tin ngữ cảnh sâu hơn. Điều này cho phép khả năng suy luận & truy vấn nâng cao, ví dụ: xác định các mẫu, khám phá các loại kết nối cụ thể hoặc suy ra các mối quan hệ mới.
In example of an academic research network at a university, a knowledge graph might represent various entities e.g. Professors, Students, Papers, & Research Topics, & explicitly define relationships between them. E.g., Professors & Students could be associated with Papers through an Authorship relationship, while Professors might also Supervise Students. Furthermore, graph would reflect hierarchical structures, e.g., Professors & Students being categorized under Departments. Can see this knowledge graph depicted in Fig. 1.5: A knowledge graph representing an academic research network within a university's physics department. Graph illustrates both hierarchical relationships, e.g., professors & students as members of department, & behavioral relationships, e.g., professors supervising students & authoring papers. Entities e.g. Professors, Students, Papers, & Topics are connected through semantically meaningful relationships (Supervises, Wrote, Inspires). Entities also have detailed features (Name, Department, Type) providing further context. Semantic connections & features enable advanced querying & analysis of complex academic interactions.
– Trong ví dụ về mạng lưới nghiên cứu học thuật tại một trường đại học, biểu đồ kiến thức có thể biểu diễn nhiều thực thể khác nhau, ví dụ: Giáo sư, Sinh viên, Bài báo, & Chủ đề nghiên cứu, & định nghĩa rõ ràng mối quan hệ giữa chúng. Ví dụ: Giáo sư & Sinh viên có thể được liên kết với Bài báo thông qua mối quan hệ Tác giả, trong khi Giáo sư cũng có thể Giám sát Sinh viên. Hơn nữa, biểu đồ sẽ phản ánh các cấu trúc phân cấp, ví dụ: Giáo sư & Sinh viên được phân loại theo Khoa. Có thể xem biểu đồ kiến thức này được mô tả trong Hình 1.5: Biểu đồ kiến thức biểu diễn mạng lưới nghiên cứu học thuật trong khoa vật lý của một trường đại học. Biểu đồ minh họa cả các mối quan hệ phân cấp, ví dụ: giáo sư & sinh viên là thành viên của khoa, & các mối quan hệ hành vi, ví dụ: giáo sư hướng dẫn sinh viên & biên soạn bài báo. Các thực thể, ví dụ: Giáo sư, Sinh viên, Bài báo, & Chủ đề được kết nối thông qua các mối quan hệ có ý nghĩa ngữ nghĩa (Giám sát, Viết, Truyền cảm hứng). Các thực thể cũng có các tính năng chi tiết (Tên, Khoa, Loại) cung cấp thêm ngữ cảnh. Kết nối ngữ nghĩa & các tính năng cho phép truy vấn nâng cao & phân tích các tương tác học thuật phức tạp.

Note 1. *Should use multigraphs to further represent knowledge graphs.*

A key feature of knowledge graphs is their ability to provide explicit context. Unlike conventional heterogeneous graphs, which display different types of entities & their basic connections without detailed semantic meaning, knowledge graphs go further by defining specific types & meanings of relationships. E.g., while a traditional graph might show that Professors are connected to Departments or that Students are linked to Papers, a knowledge graph would specify that Professors supervise Students or that Students & Professors Wrote Papers. This added layer of meaning enables more powerful querying & analysis, making knowledge graphs particularly valuable in fields e.g. NLP, recommendation systems, & academic research analysis.
– 1 đặc điểm quan trọng của đồ thị tri thức là khả năng cung cấp ngữ cảnh rõ ràng. Không giống như các đồ thị không đồng nhất thông thường, vốn hiển thị các loại thực thể khác nhau & các kết nối cơ bản của chúng mà không có ý nghĩa ngữ nghĩa chi tiết, đồ thị tri thức tiến xa hơn bằng cách xác định các loại & ý nghĩa cụ thể của các mối quan hệ. Ví dụ: trong khi đồ thị truyền thống có thể hiển thị Giáo sư được kết nối với Khoa hoặc Sinh viên được liên kết với Bài báo, đồ thị tri thức sẽ chỉ rõ Giáo sư hướng dẫn Sinh viên hoặc Sinh viên & Giáo sư viết Bài báo. Lớp ý nghĩa bổ sung này cho phép truy vấn & phân tích mạnh mẽ hơn, khiến đồ thị tri thức đặc biệt có giá trị trong các lĩnh vực như NLP, hệ thống đề xuất, & phân tích nghiên cứu học thuật.

· Hypergraphs. 1 of more complex & difficult graphs to work with is hypergraph. *Hypergraphs* are those where a single edge can be connected to multiple different nodes. For graphs that are not hypergraphs, edges are used to connected exactly 2 nodes (or a node to itself for self-loops). As shown in Fig. 1.6: 1 undirected hypergraph, illustrated in 2 ways. On left, have a graph whose edges are represented by shaded areas, marked by letters, & whose vertices are dots, marked by numbers. On right, have a graph whose edge lines (marked by letters) connect up to 3 nodes (circles marked by numbers)., edges in a hypergraph can connect between any number of nodes. Complexity of a hypergraph is reflected in its adjacency data. For typical graphs, network connectivity is represented by a 2D adjacency matrix. For hypergraphs, adjacency matrix extends to a higher dimensional tensor, referred to as an *incidence tensor*. This tensor is N-dimensional, where N is maximum number of nodes connected by a single edge. An example of a hypergraph might be a communication platform that allows for group chats as well as single person conversations. In an ordinary graph, edges would only connect 2

people. In a hypergraph, 1 hyperedge could connect multiple people, representing a group chat.

– Siêu đồ thị. 1 trong những đồ thị phức tạp hơn & khó làm việc hơn là siêu đồ thị. Siêu đồ thị là những đồ thị mà một cạnh đơn có thể được kết nối với nhiều nút khác nhau. Đối với những đồ thị không phải là siêu đồ thị, các cạnh được sử dụng để kết nối chính xác 2 nút (hoặc một nút với chính nó đối với các vòng lặp tự thân). Như thể hiện trong Hình 1.6: 1 siêu đồ thị vô hướng, được minh họa theo 2 cách. Bên trái, có một đồ thị mà các cạnh được biểu diễn bằng các vùng tô bóng, được đánh dấu bằng các chữ cái, & có các đỉnh là các dấu chấm, được đánh dấu bằng các số. Bên phải, có một đồ thị mà các đường cạnh (được đánh dấu bằng các chữ cái) kết nối tối đa 3 nút (các vòng tròn được đánh dấu bằng các số)., các cạnh trong siêu đồ thị có thể kết nối giữa bất kỳ số lượng nút nào. Độ phức tạp của siêu đồ thị được phản ánh trong dữ liệu kề của nó. Đối với các đồ thị thông thường, kết nối mạng được biểu diễn bằng ma trận kề 2D. Đối với siêu đồ thị, ma trận kề mở rộng đến một tenxơ có chiều cao hơn, được gọi là tenxơ *incidence tenxơ*. Tenxơ này là tenxơ N chiều, trong đó N là số lượng nút tối đa được kết nối bởi một cạnh duy nhất. Một ví dụ về siêu đồ thị có thể là một nền tảng giao tiếp cho phép trò chuyện nhóm cũng như trò chuyện cá nhân. Trong một đồ thị thông thường, các cạnh chỉ kết nối 2 người. Trong một siêu đồ thị, 1 siêu cạnh có thể kết nối nhiều người, đại diện cho một cuộc trò chuyện nhóm.

* 1.2.3. Graph-based learning. Graphs are ubiquitous in our everyday life. *Graph-based learning* takes graphs as input data to build models that give insight into questions about this data. Later in this chap, look at different examples of graph data as well as at sort of questions & tasks we can use graph-based learning to answer.

– Học tập dựa trên đồ thị. Đồ thị hiện diện khắp nơi trong cuộc sống hàng ngày của chúng ta. *Học tập dựa trên đồ thị* sử dụng đồ thị làm dữ liệu đầu vào để xây dựng các mô hình cung cấp thông tin chi tiết về các câu hỏi liên quan đến dữ liệu này. Ở phần sau của chương này, hãy xem xét các ví dụ khác nhau về dữ liệu đồ thị cũng như các loại câu hỏi & bài tập mà chúng ta có thể sử dụng học tập dựa trên đồ thị để trả lời.

Graph-based learning uses a variety of ML methods to build *representations* of graphs. These representations are then used for downstream tasks e.g. node or link prediction or graph classification. In Chap. 2, learn about 1 of essential tools in graph-based learning, building embeddings. Briefly, embeddings are *low-dimensional* vector representations. Can build an embedding of different nodes, edges, or entire graphs, & there are a number of different ways to do this e.g. Node2Vec (N2V) or DeepWalk algorithms.

– Học tập dựa trên đồ thị sử dụng nhiều phương pháp học máy khác nhau để xây dựng *biểu diễn* của đồ thị. Các biểu diễn này sau đó được sử dụng cho các tác vụ hạ nguồn, ví dụ như dự đoán nút hoặc liên kết hoặc phân loại đồ thị. Trong Chương 2, hãy tìm hiểu về một trong những công cụ thiết yếu trong học tập dựa trên đồ thị, đó là xây dựng nhúng. Nói một cách ngắn gọn, nhúng là biểu diễn vectơ *chiều thấp*. Có thể xây dựng nhúng của các nút, cạnh hoặc toàn bộ đồ thị khác nhau, và có một số cách khác nhau để thực hiện việc này, ví dụ như thuật toán Node2Vec (N2V) hoặc DeepWalk.

Methods for analysis on graph data have been around for a long time, at least as early as 1950s when *clique methods* used certain features of a graph to identify subsets or communities in graph data [4].

– Các phương pháp phân tích dữ liệu đồ thị đã có từ rất lâu, ít nhất là từ những năm 1950 khi *clique methods* sử dụng một số tính năng nhất định của đồ thị để xác định các tập hợp con hoặc cộng đồng trong dữ liệu đồ thị.

1 of most famous graph-based algorithms is PageRank, which was developed by Larry Page & Sergey Brin in 1996 & formed basis for Google's search algorithms. Some believe: this algorithm was a key element in company's meteoric rise in following years. This highlights that a successful graph-based learning algorithm can have a huge effect.

– 1 trong những thuật toán dựa trên đồ thị nổi tiếng nhất là PageRank, được phát triển bởi Larry Page và Sergey Brin vào năm 1996 và tạo nền tảng cho các thuật toán tìm kiếm của Google. Một số người tin rằng: thuật toán này là yếu tố then chốt cho sự phát triển vượt bậc của công ty trong những năm tiếp theo. Điều này nhấn mạnh rằng một thuật toán học dựa trên đồ thị thành công có thể mang lại hiệu quả to lớn.

These methods are only a small subset of graph-based learning & analysis techniques. Others include belief propagation [5], graph kernel methods [6], label propagation [7], & isomaps [8]. However, in this book, focus on 1 of newest & most exciting additions to family of graph-based learning techniques: GNNs.

– Những phương pháp này chỉ là một tập hợp con nhỏ của các kỹ thuật học tập dựa trên đồ thị & phân tích. Các phương pháp khác bao gồm truyền bá niềm tin [5], phương pháp hạt nhân đồ thị [6], truyền bá nhãn [7], & isomaps [8]. Tuy nhiên, trong cuốn sách này, chúng tôi tập trung vào một trong những bổ sung mới nhất & thú vị nhất cho nhóm kỹ thuật học tập dựa trên đồ thị: GNN.

* 1.2.4. What is a GNN? GNNs combine graph-based learning with DL, i.e., neural networks are used to build embeddings & process relational data. An overview of inner workings of a GNN is shown in Fig. 1.7: An overview of how GNNs work. An input graph is passed to a GNN. GNN then uses neural networks to transform graph features e.g. nodes or edges into nonlinear embeddings through a process known as message passing. These embeddings are then tuned to specific unknown properties using training data. After GNN is trained, it can predict unknown features of a graph.

– GNN kết hợp học tập dựa trên đồ thị với DL, tức là mạng nơ-ron được sử dụng để xây dựng các nhúng & xử lý dữ liệu quan hệ. Tổng quan về hoạt động bên trong của GNN được thể hiện trong Hình 1.7: Tổng quan về cách thức hoạt động của GNN. Một đồ thị đầu vào được truyền đến GNN. Sau đó, GNN sử dụng mạng nơ-ron để chuyển đổi các đặc trưng đồ thị, ví dụ như các nút hoặc cạnh, thành các nhúng phi tuyến tính thông qua một quá trình được gọi là truyền thông điệp. Các nhúng này sau đó được điều chỉnh theo các thuộc tính cụ thể chưa biết bằng cách sử dụng dữ liệu huấn luyện. Sau khi GNN được huấn luyện, nó có thể dự đoán các đặc trưng chưa biết của đồ thị.

GNNs allows you to represent & learn from graphs, including their constituent nodes, edges, & features. In particular, many methods of GNNs are built specifically to scale effectively with size & complexity of a graph, i.e., GNNs can operate on huge graphs. In this sense, GNNs provide analogous advantages to relational data as convolutional neural networks have given for image-based data & computer vision.

– GNN cho phép bạn biểu diễn & học từ các đồ thị, bao gồm các nút, cạnh, & đặc trưng cấu thành của chúng. Đặc biệt, nhiều phương pháp GNN được xây dựng chuyên biệt để mở rộng hiệu quả theo kích thước & độ phức tạp của đồ thị, tức là GNN có thể hoạt động trên các đồ thị rất lớn. Theo nghĩa này, GNN mang lại những lợi thế tương tự cho dữ liệu quan hệ như mạng nơ-ron tích chập đã mang lại cho dữ liệu dựa trên hình ảnh & thị giác máy tính.

Historically, applying traditional ML methods to graph data structures has been challenging because graph data, when represented in grid-like formats & data structures, can lead to massive repetitions of data. To address this, graph-based learning focuses on approaches that are *permutation invariant*, i.e., ML method is uninfluenced by ordering of graph representation. In concrete terms, it means: we can shuffle rows & columns of adjacency matrix without affecting our algorithm's performance. Whenever we are working with data that contains relational data, i.e., has an adjacency matrix, then we want to use a ML method that is permutation invariant to make our method more general & efficient. Although GNNs can be applied to all graph data, GNNs are especially useful because they can deal with huge graph datasets & typically perform better than other ML methods.

– Theo truyền thống, việc áp dụng các phương pháp ML truyền thống vào cấu trúc dữ liệu đồ thị là một thách thức vì dữ liệu đồ thị, khi được biểu diễn ở định dạng dạng lưới & cấu trúc dữ liệu, có thể dẫn đến sự lặp lại dữ liệu rất lớn. Để giải quyết vấn đề này, học dựa trên đồ thị tập trung vào các phương pháp *permutation invariant*, tức là phương pháp ML không bị ảnh hưởng bởi thứ tự biểu diễn đồ thị. Nói một cách cụ thể, điều này có nghĩa là: chúng ta có thể xáo trộn các hàng & cột của ma trận kề mà không ảnh hưởng đến hiệu suất của thuật toán. Bất cứ khi nào chúng ta làm việc với dữ liệu có chứa dữ liệu quan hệ, tức là có ma trận kề, thì chúng ta muốn sử dụng phương pháp ML không thay đổi hoán vị để làm cho phương pháp của chúng ta tổng quát hơn & hiệu quả hơn. Mặc dù GNN có thể được áp dụng cho tất cả dữ liệu đồ thị, nhưng GNN đặc biệt hữu ích vì chúng có thể xử lý các tập dữ liệu đồ thị khổng lồ & thường hoạt động tốt hơn các phương pháp ML khác.

Permutation invariances are a type of *inductive bias*, or an algorithm's learning bias, & are powerful tools for designing ML algorithms [1]. Need for permutation-invariant approaches is 1 of central reasons that graph-based learning has increased in popularity in recent years.

– Bất biến hoán vị là một loại *thiên vị quy nạp*, hay thiên vị học tập của thuật toán, & là những công cụ mạnh mẽ để thiết kế các thuật toán ML [1]. Nhu cầu về các phương pháp bất biến hoán vị là một trong những lý do chính khiến học tập dựa trên đồ thị ngày càng phổ biến trong những năm gần đây.

Insights. Being designed for permutation-invariant data comes with some drawbacks along with its advantages. GNNs are not as

While this might seem obvious, images & tables are not permutation invariant & therefore not a good fit for GNNs. If we shuffle rows & columns of an image, then we scramble input. Instead, ML algorithms for images seek *translational invariance*, i.e., we can translate (shift) object in an image, & it won't affect performance of algorithm. Other neural networks, e.g. convolutional neural networks (CNNs) typically perform much better on images.

– Việc được thiết kế cho dữ liệu bất biến hoán vị đi kèm với một số nhược điểm bên cạnh những ưu điểm của nó.

GNN không phù hợp lắm với các dữ liệu khác, ví dụ như hình ảnh hoặc bảng. Mặc dù điều này có vẻ hiển nhiên, nhưng hình ảnh & bảng không bất biến hoán vị & do đó không phù hợp với GNN. Nếu chúng ta xáo trộn các hàng & cột của một hình ảnh, thì chúng ta sẽ xáo trộn dữ liệu đầu vào. Thay vào đó, các thuật toán ML cho hình ảnh tìm kiếm *translational invariance*, tức là chúng ta có thể dịch chuyển (shift) đối tượng trong một hình ảnh, & điều này sẽ không ảnh hưởng đến hiệu suất của thuật toán. Các mạng nơ-ron khác, ví dụ như mạng nơ-ron tích chập (CNN) thường hoạt động tốt hơn nhiều trên hình ảnh.

∗ 1.2.5. Differences between tabular & graph data. Graph data includes all data with some relational content, making it a powerful way to represent complex connections. While graph data might initially seem distinct from traditional tabular data, many datasets that are typically represented in tables can be recreated as graphs with some data engineering & imagination. Take a closer look at Titanic dataset, a classic example in ML, & explore how it can be transformed from a table format to a graph format.

– Sự khác biệt giữa dữ liệu dạng bảng & dữ liệu đồ thị. Dữ liệu đồ thị bao gồm tất cả dữ liệu có nội dung quan hệ, khiến nó trở thành một phương pháp mạnh mẽ để biểu diễn các kết nối phức tạp. Mặc dù ban đầu dữ liệu đồ thị có vẻ khác biệt so với dữ liệu dạng bảng truyền thống, nhưng nhiều tập dữ liệu thường được biểu diễn dưới dạng bảng có thể được tái tạo dưới dạng đồ thị với một chút kỹ thuật dữ liệu & trí tưởng tượng. Hãy xem xét kỹ hơn tập dữ liệu Titanic, một ví dụ kinh điển trong ML, & khám phá cách nó có thể được chuyển đổi từ định dạng bảng sang định dạng đồ thị.

Titanic dataset describes passengers on Titanic, a ship that famously met an untimely end when it collided with an iceberg. Historically, this datasets has been analyzed in tabular format, containing rows for each passenger with columns representing features e.g. age, gender, fare, class, & survival status. However, dataset also contains rich, unexplored relationships that are not immediately visible in a table format Fig. 1.8: Titanic Dataset is usually displayed & analyzed using a table format.

– Bộ dữ liệu Titanic mô tả hành khách trên Titanic, một con tàu nổi tiếng đã gặp phải cái chết bất ngờ khi va chạm với một tảng băng trôi. Trước đây, bộ dữ liệu này được phân tích theo định dạng bảng, bao gồm các hàng cho mỗi hành khách và các cột biểu diễn các đặc điểm như tuổi, giới tính, giá vé, hạng ghế, & tình trạng sống sót. Tuy nhiên, bộ dữ liệu cũng chứa các mối quan hệ phong phú, chưa được khám phá mà không thể hiển thị ngay lập tức ở định dạng bảng Hình 1.8: Bộ dữ liệu Titanic thường được hiển thị & phân tích bằng định dạng bảng.

· Recasting Titanic dataset as a graph. To transform Titanic dataset into a graph, need to consider how to represent underlying relationships between passengers as nodes & edges:

1. Nodes: In graph, each passenger can be represented as a node. Can also introduce nodes for other entities, e.g. cabins, families, or even groups e.g. "3rd-class passengers".

2. Edges represent relationships or connections between these nodes, e.g.: Passengers who are family members (siblings, spouses, parents, or children) based on available data; Passengers who share a cabin or were traveling together; Social or business relationships that might be inferred from shared ticket numbers, last names, or other identifying features.

– Tái cấu trúc tập dữ liệu Titanic dưới dạng đồ thị. Để chuyển đổi tập dữ liệu Titanic thành đồ thị, cần xem xét cách biểu diễn các mối quan hệ cơ bản giữa hành khách dưới dạng các nút & cạnh:

1. Nút: Trong đồ thị, mỗi hành khách có thể được biểu diễn dưới dạng một nút. Cũng có thể giới thiệu các nút cho các thực thể khác, ví dụ: cabin, gia đình hoặc thậm chí các nhóm, ví dụ: "hành khách hạng 3".
2. Cạnh biểu diễn các mối quan hệ hoặc kết nối giữa các nút này, ví dụ: Hành khách là thành viên gia đình (anh chị em ruột, vợ/chồng, cha mẹ hoặc con cái) dựa trên dữ liệu có sẵn; Hành khách ở chung cabin hoặc đi cùng nhau; Các mối quan hệ xã hội hoặc kinh doanh có thể được suy ra từ số vé, họ hoặc các đặc điểm nhận dạng chung khác.

To construct this graph, need to use existing information in table & potentially enrich it with secondary data sources or assumptions (e.g., linking last names to create family groups). This process converts tabular data into a graph-based structure, shown in Fig. 1.9: Titanic dataset, showing family relationships of people on Titanic visualized as a graph. Here, can see that there was a rich social network as well as many passengers with unknown family ties., where each edge & node encapsulates meaningful relational data.

– Để xây dựng biểu đồ này, cần sử dụng thông tin hiện có trong bảng & có thể làm giàu nó bằng các nguồn dữ liệu thứ cấp hoặc giả định (ví dụ: liên kết họ để tạo nhóm gia đình). Quá trình này chuyển đổi dữ liệu dạng bảng thành cấu trúc dạng biểu đồ, được hiển thị trong Hình 1.9: Bộ dữ liệu Titanic, thể hiện mối quan hệ gia đình của những người trên Titanic được trực quan hóa dưới dạng biểu đồ. Ở đây, có thể thấy rằng có một mạng lưới xã hội phong phú cũng như nhiều hành khách có mối quan hệ gia đình chưa rõ ràng., trong đó mỗi cạnh & nút đóng gói dữ liệu quan hệ có ý nghĩa.

· How graph data adds depth & meaning. Once dataset is represented as a graph, it provides a much deeper view of social & familial connections between passengers, e.g.,:

1. *Family relationships*: Graph clearly shows how certain passengers were related (e.g., as parents, children, or siblings). This could help us understand survival patterns, as family members might have behaved differently in a crisis than individuals traveling alone.
2. *Social networks*: Beyond families, graph could reveal broader social networks (e.g., friendships or business connections), which could be important factors in analyzing behavior & outcomes.
3. *Community insights*: Graph structure also allows for community detection algorithms to identify clusters of related or connected passengers, which may reveal new insights into survival rates, rescue patterns, or other behaviors.

Graph representations add depth by specifying connections that might not be obvious in a tabular format. E.g., understanding who traveled together, who shared a cabin, or who had social or family ties can provide more context on survival rates & passenger behavior. This is crucial for tasks e.g. node prediction, where we want to predict attributes or outcomes based on relationships represented in graph.

– Cách dữ liệu đồ thị tăng thêm chiều sâu & ý nghĩa. Khi tập dữ liệu được biểu diễn dưới dạng đồ thị, nó cung cấp cái nhìn sâu sắc hơn nhiều về các mối quan hệ xã hội & gia đình giữa các hành khách, ví dụ:

1. *Mối quan hệ gia đình*: Đồ thị cho thấy rõ mối quan hệ của một số hành khách nhất định (ví dụ: cha mẹ, con cái hoặc anh chị em ruột). Điều này có thể giúp chúng ta hiểu được các mô hình sinh tồn, vì các thành viên trong gia đình có thể đã hành xử khác nhau trong khủng hoảng so với những cá nhân đi du lịch một mình.
2. *Mạng xã hội*: Ngoài gia đình, đồ thị có thể tiết lộ các mạng xã hội rộng hơn (ví dụ: tình bạn hoặc kết nối kinh doanh), đây có thể là những yếu tố quan trọng trong việc phân tích hành vi & kết quả.
3. *Thông tin chi tiết về cộng đồng*: Cấu trúc đồ thị cũng cho phép các thuật toán phát hiện cộng đồng xác định các cụm hành khách có liên quan hoặc kết nối, điều này có thể tiết lộ những hiểu biết mới về tỷ lệ sống sót, mô hình cứu hộ hoặc các hành vi khác.

Biểu diễn đồ thị tăng thêm chiều sâu bằng cách chỉ định các kết nối có thể không rõ ràng ở định dạng bảng. Ví dụ, việc hiểu rõ ai đi cùng nhau, ai ở chung cabin, hoặc ai có mối quan hệ xã hội hoặc gia đình có thể cung cấp thêm bối cảnh về tỷ lệ sống sót & hành vi của hành khách. Điều này rất quan trọng đối với các tác vụ như dự đoán nút, trong đó chúng ta muốn dự đoán các thuộc tính hoặc kết quả dựa trên các mối quan hệ được biểu diễn trên đồ thị.

By creating an adjacency matrix or defining graph edges & nodes based on relationships in dataset, can transition from simple data analysis to more sophisticated graph-based learning methods.

– Bằng cách tạo ma trận kề hoặc xác định các cạnh và nút đồ thị dựa trên các mối quan hệ trong tập dữ liệu, có thể chuyển đổi từ phân tích dữ liệu đơn giản sang các phương pháp học dựa trên đồ thị phức tạp hơn.

◦ 1.3. GNN applications: Case studies. GNNs are neural networks designed to work on relational data. They give new ways for relational data to be transformed & manipulated, by being easier to scale & more accurate than previous graph-based learning methods. In following, discuss some exciting applications of GNNs, to see, at a high level, how this class of models are solving real-world problems.

– Ứng dụng GNN: Nghiên cứu điển hình. GNN là mạng nơ-ron được thiết kế để hoạt động trên dữ liệu quan hệ. Chúng cung cấp những cách thức mới để chuyển đổi & thao tác dữ liệu quan hệ, nhờ khả năng mở rộng dễ dàng & chính xác hơn so với các phương pháp học dựa trên đồ thị trước đây. Tiếp theo, hãy thảo luận về một số ứng dụng thú vị của GNN, để xem xét ở cấp độ tổng quan, cách lớp mô hình này đang giải quyết các vấn đề thực tế.

∗ 1.3.1. Recommendation engines. Enterprise graphs can exceed billions of nodes & many billions of edges. On other hand, many GNNs are benchmarked on datasets that consist of fewer than a million nodes. When applying GNNs to large

graphs, adjustments of training & inference algorithms & storage techniques all have to be made. (Can learn more about specifics of scaling GNNs in Chap. 7.)

– Công cụ đề xuất. Đồ thị doanh nghiệp có thể vượt quá hàng tỷ nút & hàng tỷ cạnh. Mặt khác, nhiều GNN được đánh giá chuẩn trên các tập dữ liệu có ít hơn một triệu nút. Khi áp dụng GNN cho đồ thị lớn, cần phải điều chỉnh các thuật toán huấn luyện & suy luận & kỹ thuật lưu trữ. (Bạn có thể tìm hiểu thêm về các chi tiết cụ thể về việc mở rộng GNN trong Chương 7.)

1 of most well-known industry examples of GNNs is their use as recommendation engines. E.g., Pinterest is a social media platform for finding & sharing images & ideas. there are 2 major concepts to Pinterest's users: collections or categories of ideas, called *boards* (like a bulletin board); & objects a user wants to bookmark called *pins*. Pins include images, videos, & websites URLs. A user board focused on dogs might then include pins of pet photos, puppy videos, or dog-related website links. A board's pins aren't exclusive to it; a pet drawing that was pinned to Dogs board could also be pinned to a Puppies board, as shown in Fig. 1.10: A bipartite graph that is like Pinterest graph. Nodes in this case are pins & boards.

– 1 trong những ví dụ nổi tiếng nhất về GNN trong ngành là việc sử dụng chúng làm công cụ đề xuất. E.g., Pinterest là một nền tảng truyền thông xã hội để tìm kiếm & chia sẻ hình ảnh & ý tưởng. Có 2 khái niệm chính đối với người dùng Pinterest: bộ sưu tập hoặc danh mục ý tưởng, được gọi là *boards* (giống như bảng tin); & đối tượng mà người dùng muốn đánh dấu được gọi là *pins*. Ghim bao gồm hình ảnh, video, & URL trang web. Một bảng người dùng tập trung vào chó có thể bao gồm các ghim ảnh thú cưng, video về chó con hoặc các liên kết trang web liên quan đến chó. Ghim của một bảng không chỉ giới hạn ở đó; một bức vẽ thú cưng được ghim vào bảng Chó cũng có thể được ghim vào bảng Chó con, như thể hiện trong Hình 1.10: Đồ thị hai phần giống như đồ thị Pinterest. Các nút trong trường hợp này là ghim & boards.

1 way to interpret relationships between pins & boards is as a *biparite graph*. For Pinterest graph, all pins are connected to boards, but no pin is connected to another pin, & no board is connected to another board. Pins & boards are 2 classes of nodes. Members of these classes can be linked to members of other class, but not to member of same class. Pinterest graph was reported to have 3 billion nodes & 18 billion edges.

– 1 cách để diễn giải mối quan hệ giữa các chân & bảng là sử dụng đồ thị lưỡng đối. Đối với đồ thị Pinterest, tất cả các chân được kết nối với các bảng, nhưng không có chân nào được kết nối với chân khác, & không có bảng nào được kết nối với bảng khác. Chân & bảng là 2 lớp nút. Các thành viên của các lớp này có thể được liên kết với các thành viên của lớp khác, nhưng không thể liên kết với các thành viên của cùng lớp. Đồ thị Pinterest được báo cáo có 3 tỷ nút & 18 tỷ cạnh.

PinSage, a graph convolutional network (GCN), was 1 of 1st documented highly scaled GNNs used in an enterprise system [9]. This was used in Pinterest's recommendation systems to overcome past challenges of applying graph-learning models to massive graphs. Compared to baseline methods, tests on this system showed it improved user engagement by 30%. Specifically, PinSage was used to predict which objects should be recommended to be included in a user's graph. However, GNNs can also be used to predict what an object is, e.g. whether it contains a dog or mountain, based on the rest of nodes in graph & how they are connected. Do a deep dive on GCNs, of which PinSage is an extension, in Chap. 3.

– PinSage, một mạng tích chập đồ thị (GCN), là 1 trong những GNN có quy mô lớn đầu tiên được ghi nhận sử dụng trong hệ thống doanh nghiệp [9]. Mạng này được sử dụng trong các hệ thống đề xuất của Pinterest để vượt qua những thách thức trước đây khi áp dụng các mô hình học đồ thị vào các đồ thị lớn. So với các phương pháp cơ sở, các thử nghiệm trên hệ thống này cho thấy nó đã cải thiện mức độ tương tác của người dùng lên 30%. Cụ thể, PinSage được sử dụng để dự đoán đối tượng nào nên được đề xuất đưa vào đồ thị của người dùng. Tuy nhiên, GNN cũng có thể được sử dụng để dự đoán một đối tượng là gì, ví dụ: nó chứa một con chó hay một ngọn núi, dựa trên các nút còn lại trong đồ thị & cách chúng được kết nối. Hãy tìm hiểu sâu hơn về GCN, trong đó PinSage là một phần mở rộng, trong Chương 3.

∗ 1.3.2. Drug discovery & molecular science. In chemistry & molecular sciences, a prominent problem has been representing molecules in a general, application-agnostic way, & inferring possible interfaces between molecules, e.g. proteins. For molecule representation, can see: drawings of molecules that are common in high school chemistry classes bear resemblance to a graph structure, consisting of nodes (atoms) & edges (atomic bonds), as shown in Fig. 1.11: In this molecule, can see individual atoms as nodes & atomic bonds as edges.

– Khám phá thuốc & khoa học phân tử. Trong hóa học & khoa học phân tử, một vấn đề nổi cộm là biểu diễn phân tử theo cách tổng quát, không phụ thuộc vào ứng dụng, & suy ra các giao diện khả dĩ giữa các phân tử, ví dụ như protein. Về biểu diễn phân tử, có thể thấy: các hình vẽ phân tử thường thấy trong các lớp hóa học trung học phổ thông có cấu trúc đồ thị, bao gồm các nút (nguyên tử) & các cạnh (liên kết nguyên tử), như thể hiện trong Hình 1.11: Trong phân tử này, có thể thấy các nguyên tử riêng lẻ là các nút & các liên kết nguyên tử là các cạnh.

Applying GNNs to these structures can, in certain circumstances, outperform traditional "fingerprint" methods for determining properties of a molecule. These traditional methods involve creation of features by domain experts to capture a molecule's properties, e.g. interpreting presence or absence of certain molecules or atoms [10]. GNNs learn new data-driven features that can be used to group certain molecules together in new & unexpected ways or even to propose new molecules for synthesis. This is extremely important for predicting whether a chemical is toxic or safe for use or whether it has some downstream effects that can affect disease progression. Therefore, GNNs have shown themselves to be incredibly useful in field of drug discovery.

– Việc áp dụng GNN vào các cấu trúc này, trong một số trường hợp, có thể vượt trội hơn các phương pháp "dấu vân tay" truyền thống để xác định các đặc tính của một phân tử. Các phương pháp truyền thống này liên quan đến việc tạo ra các đặc điểm bởi các chuyên gia trong lĩnh vực để nắm bắt các đặc tính của một phân tử, ví dụ: diễn giải sự hiện diện hoặc vắng mặt của một số phân tử hoặc nguyên tử nhất định [10]. GNN học các đặc điểm mới dựa trên dữ liệu, có thể được sử dụng để nhóm các phân tử nhất định lại với nhau theo những cách mới & bất ngờ, hoặc thậm chí đề xuất các phân tử mới để tổng hợp. Điều này cực kỳ quan trọng để dự đoán liệu một hóa chất có độc hại hay an toàn để sử dụng

hay liệu nó có một số tác động hạ lưu có thể ảnh hưởng đến sự tiến triển của bệnh hay không. Do đó, GNN đã chứng tỏ mình cực kỳ hữu ích trong lĩnh vực khám phá thuốc.

    - 1.3.3. Mechanical reasoning.
  - 1.4. When to use a GNN?
- 2. Graph embeddings.

  PART 3: GRAPH NEURAL NETWORKS GNNs
- 3. Graph convolutional networks & GraphSAGE.
- 4. Graph attention networks.
- 5. Graph autoencoders.

  PART 3: ADVANCED TOPICS.
- 6. Dynamic graphs: Spatiotemporal GNNs.
- 7. Learning & inference at scale.
- 8. Considerations for GNN projects.
- A. Discovering graphs.
- B. Installing & configuring PyTorch Geometric.

2 Miscellaneous