Graph Neural Networks (GNNs)

Nguyễn Quản Bá Hồng*

Ngày 5 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 & 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 & 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, & 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 & 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 & & 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 & 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, & 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 & Sergey Brin vào năm 1996 & 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 & 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 & 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.
Drug discovery, especially for GNNs, can be understood as a graph prediction problem. *Graph prediction* tasks are those that require learning & predicting properties about entire graph. For drug discovery, aim: predict properties e.g. toxicity or treatment effectiveness (discriminative) or to suggest entirely new graphs that should be synthesized & tested (generative). To suggest these new graphs, drug discovery methods often combine GNNs with other generative models e.g. variational graph autoencoders (VGAEs), as shown, e.g., in Fig. 1.12: A GNN system used to predict new molecules [11]. Workflow here starts on left with a representation of a molecule as a graph. In middle parts of figure, this graph representation is transformed via a GNN into a latent representation. Latent representation is then transformed back to molecule to ensure: latent space can be decoded (right). Describe VGAEs in more detail in Chap. 5 & show how we can use these to predict molecules.
– Khám phá thuốc, đặc biệt đối với GNN, có thể được hiểu là một bài toán dự đoán đồ thị. Nhiệm vụ *Dự đoán đồ thị* là những nhiệm vụ yêu cầu học & dự đoán các thuộc tính về toàn bộ đồ thị. Đối với khám phá thuốc, mục tiêu: dự đoán các thuộc tính, ví dụ như độc tính hoặc hiệu quả điều trị (phân biệt) hoặc đề xuất các đồ thị hoàn toàn mới cần được tổng hợp & thử nghiệm (sinh). Để đề xuất các đồ thị mới này, các phương pháp khám phá thuốc thường kết hợp GNN với các mô hình sinh khác, ví dụ như bộ mã hóa tự động đồ thị biến phân (VGAE), như được hiển thị, ví dụ, trong Hình 1.12: Hệ thống GNN được sử dụng để dự đoán các phân tử mới [11]. Quy trình làm việc ở đây bắt đầu ở bên trái với biểu diễn của một phân tử dưới dạng đồ thị. Ở phần giữa của hình, biểu diễn đồ thị này được chuyển đổi thông qua GNN thành biểu diễn tiềm ẩn. Biểu diễn tiềm ẩn sau đó được chuyển đổi trở lại thành phân tử để đảm bảo: không gian tiềm ẩn có thể được giải mã (bên phải). Mô tả VGAE chi tiết hơn trong Chương 5 & chỉ ra cách chúng ta có thể sử dụng chúng để dự đoán các phân tử.

∗ 1.3.3. Mechanical reasoning. Develop rudimentary intuition about mechanics & physics of world around us at a remarkably young age & without any formal training in subject. Do not need to write down a set of equations to know how to catch a bouncing ball. Do not even have to be in presence of a physical ball. Given a series of snapshots of a bouncing ball, can predict reasonably well where ball is going to end up.
– Lý luận cơ học. Phát triển trực giác cơ bản về cơ học & vật lý của thế giới xung quanh chúng ta ngay từ khi còn rất nhỏ & mà không cần bất kỳ sự đào tạo chính thức nào về môn học này. Không cần phải viết ra một loạt phương trình để biết cách bắt một quả bóng đang nảy. Thậm chí không cần phải ở gần một quả bóng thực tế. Chỉ cần một loạt ảnh chụp nhanh về một quả bóng đang nảy, bạn có thể dự đoán khá chính xác vị trí quả bóng sẽ rơi.
While these problems might seem trivial for us, they are critical for many physical industries, including manufacturing & autonomous driving. E.g., autonomous driving systems need to anticipate what will happen in a traffic scene consisting of many moving objects. Until recently, this task was typically treated as a problem of computer vision. However, more recent approaches have begun to use GNNs [12]. These GNN-based methods demonstrate: including relational information, e.g. how limbs are connected, can enable algorithms to develop physical intuition about how a person or animal moves with higher accuracy & less data.
– Mặc dù những vấn đề này có vẻ tầm thường đối với chúng ta, nhưng chúng lại rất quan trọng đối với nhiều ngành công nghiệp vật lý, bao gồm sản xuất & lái xe tự động. Ví dụ, hệ thống lái xe tự động cần dự đoán những gì sẽ xảy ra trong một cảnh giao thông gồm nhiều vật thể chuyển động. Cho đến gần đây, nhiệm vụ này thường được coi là một vấn đề của thị giác máy tính. Tuy nhiên, các phương pháp tiếp cận gần đây hơn đã bắt đầu sử dụng GNN [12]. Các phương pháp dựa trên GNN này chứng minh: việc bao gồm thông tin quan hệ, ví dụ như cách các chi được kết nối, có thể cho phép các thuật toán phát triển trực giác vật lý về cách một người hoặc động vật di chuyển với độ chính xác cao hơn & ít dữ liệu hơn.
In Fig. 1.13: A graph representation of a mechanical body, taken from Sanchez-Gonzalez [13]. Body's segments are represented as nodes, & mechanical forces binding them are edges., give an example of how a body can be thought of as a "mechanical" graph. Input graphs for these physical reasoning systems have elements that reflect problem. E.g., when reasoning about a human or animal body, a graph could consist of nodes that represent points on body where limbs connect. For systems of free bodies, nodes of a graph could be individual objects e.g. bouncing balls. Edges of graph then represent physical relationship (e.g., gravitational forces, elastic springs, or rigid connections) between nodes. Given these inputs, GNNs learn to predict future states of a set of objects without explicitly calling on physical/mechanical laws [13]. These methods are a form of *edge prediction*, i.e., they predict how nodes connect over time. Furthermore, these models have to be dynamic to account for temporal evolution of system. Consider these problems in detail in Chap. 6.
– Trong Hình 1.13: Biểu diễn đồ thị của một vật thể cơ học, lấy từ Sanchez-Gonzalez [13]. Các đoạn của vật thể được biểu diễn là các nút, & lực cơ học liên kết chúng là các cạnh., đưa ra một ví dụ về cách một vật thể có thể được coi là một đồ thị "cơ học". Đồ thị đầu vào cho các hệ thống suy luận vật lý này có các thành phần phản ánh vấn đề. Ví dụ: khi suy luận về cơ thể người hoặc động vật, đồ thị có thể bao gồm các nút biểu diễn các điểm trên cơ thể nơi các chi kết nối. Đối với các hệ thống vật thể tự do, các nút của đồ thị có thể là các vật thể riêng lẻ, ví dụ: quả bóng nảy. Các cạnh của đồ thị sau đó biểu diễn mối quan hệ vật lý (ví dụ: lực hấp dẫn, lò xo đàn hồi hoặc kết nối cứng) giữa các nút. Với các đầu vào này, GNN học cách dự đoán trạng thái tương lai của một tập hợp các vật thể mà không cần gọi rõ ràng các định luật cơ học vật lý [13]. Các phương pháp này là một dạng *dự đoán cạnh*, tức là chúng dự đoán cách các nút kết nối theo thời gian. Hơn nữa, các mô hình này phải mang tính động để tính đến sự tiến hóa theo thời gian của hệ thống. Hãy xem xét chi tiết những vấn đề này trong Chương 6.

◦ 1.4. When to use a GNN? Have explored real-world applications of GNNs, identify some underlying characteristics that make problems suitable for graph-based solutions. While cases of previous section clearly involved data that was naturally modeled as a graph, crucial to recognize that GNNs can also be effectively applied to problems where graph-like nature may not be

immediately obvious.

– Khi nào nên sử dụng GNN? Đã khám phá các ứng dụng thực tế của GNN, hãy xác định một số đặc điểm cơ bản giúp bài toán phù hợp với các giải pháp dựa trên đồ thị. Mặc dù các trường hợp trong phần trước rõ ràng liên quan đến dữ liệu được mô hình hóa tự nhiên dưới dạng đồ thị, nhưng điều quan trọng là phải nhận ra rằng GNN cũng có thể được áp dụng hiệu quả cho các bài toán mà bản chất giống đồ thị có thể không rõ ràng ngay lập tức.

So, instead of simply stating GNNs are useful for graph problems, this section will help you recognize patterns & relationships within your data that could benefit from graph-based modeling, even if those relationships aren't immediately apparent. Essentially, there are 3 types of criteria for identifying GNN problems: implicit relationships & interdependencies; high dimensionality & sparsity; & complex nonlocal interactions.

– Vì vậy, thay vì chỉ đơn thuần nêu rằng GNN hữu ích cho các bài toán đồ thị, phần này sẽ giúp bạn nhận ra các mẫu & mối quan hệ trong dữ liệu của mình mà mô hình dựa trên đồ thị có thể mang lại lợi ích, ngay cả khi những mối quan hệ đó không rõ ràng ngay lập tức. Về cơ bản, có 3 loại tiêu chí để xác định các bài toán GNN: mối quan hệ ngầm & phụ thuộc lẫn nhau; tính đa chiều cao & thưa thớt; & tương tác phi cục bộ phức tạp.

∗ 1.4.1. Implicit relationships & interdependencies. Graphs are versatile data structures that can model a wide range of relationships. Even when a problem doesn't initially appear to be graph-like, even if your dataset is tabular, it is beneficial to explore whether implicit relationships or interdependencies might exist that could be represented explicitly. Implicit relationships are connections that are not immediately documented or obvious within data but can still play a significant role in understanding underlying patterns & behaviors.

– Mối quan hệ ngầm & sự phụ thuộc lẫn nhau. Đồ thị là cấu trúc dữ liệu đa năng có thể mô hình hóa một loạt các mối quan hệ. Ngay cả khi một vấn đề ban đầu có vẻ không giống đồ thị, ngay cả khi tập dữ liệu của bạn ở dạng bảng, việc khám phá xem liệu có tồn tại các mối quan hệ ngầm hoặc sự phụ thuộc lẫn nhau có thể được biểu diễn một cách rõ ràng hay không vẫn rất hữu ích. Mối quan hệ ngầm là những kết nối không được ghi chép ngay lập tức hoặc hiển nhiên trong dữ liệu nhưng vẫn có thể đóng một vai trò quan trọng trong việc hiểu các mô hình & hành vi cơ bản.

Key indicators. To determine if your problem might benefit from modeling implicit relationships with graphs, consider whether there are hidden or indirect connections between entities in your dataset. E.g., in customer behavior analysis, customers may appear as independent entities in a tabular dataset containing their purchases, demographics, & other details. However, they could be connected through social media influence, peer recommendations, or shared purchasing patterns, forming an underlying network of interactions.

– **Các chỉ số chính.** Để xác định xem vấn đề của bạn có thể được hưởng lợi từ việc mô hình hóa các mối quan hệ ngầm định bằng biểu đồ hay không, hãy xem xét liệu có các kết nối ẩn hoặc gián tiếp giữa các thực thể trong tập dữ liệu của bạn hay không. Ví dụ: trong phân tích hành vi khách hàng, khách hàng có thể xuất hiện dưới dạng các thực thể độc lập trong một tập dữ liệu dạng bảng chứa thông tin mua hàng, nhân khẩu học & các chi tiết khác của họ. Tuy nhiên, họ có thể được kết nối thông qua ảnh hưởng trên mạng xã hội, khuyến nghị của đồng nghiệp hoặc các mô hình mua sắm chung, tạo thành một mạng lưới tương tác cơ bản.

Another indicator is presence of entities that share common attributes or activities without a direct or documented relationship. In case of investors, e.g., 2 or more investors may not have any formal connection but might frequently co-invest in same companies under similar conditions. Such patterns of co-investment could indicate a shared strategy or influence. In this scenario, a graph representation can be created where nodes represent individual investors, & edges are formed between nodes when 2 or more investors co-invest in the same company. Additional attributes, e.g., investment size, timing, or types of companies invested in can be added to nodes or edges, allowing GNNs to identify patterns, trends, or even potential collaboration opportunities.

– Một chỉ báo khác là sự hiện diện của các thực thể có chung thuộc tính hoặc hoạt động mà không có mối quan hệ trực tiếp hoặc được ghi chép lại. Ví dụ, trong trường hợp nhà đầu tư, 2 hoặc nhiều nhà đầu tư có thể không có bất kỳ mối liên hệ chính thức nào nhưng thường xuyên cùng đầu tư vào cùng một công ty trong các điều kiện tương tự. Các mô hình đồng đầu tư như vậy có thể chỉ ra một chiến lược hoặc ảnh hưởng chung. Trong trường hợp này, có thể tạo biểu đồ biểu diễn, trong đó các nút đại diện cho các nhà đầu tư cá nhân, & các cạnh được hình thành giữa các nút khi 2 hoặc nhiều nhà đầu tư cùng đầu tư vào cùng một công ty. Các thuộc tính bổ sung, ví dụ: quy mô đầu tư, thời điểm đầu tư hoặc loại hình công ty được đầu tư, có thể được thêm vào các nút hoặc cạnh, cho phép GNN xác định các mô hình, xu hướng hoặc thậm chí các cơ hội hợp tác tiềm năng.

Additionally, consider whether data involves entities that are interconnected through shared references or co-occurrence patterns. Document & text data may not immediately suggest a graph structure, but if documents cite each other or share common topics or authors, they can be represented as nodes in a graph, with edges reflecting these relationships. Similarly, terms within documents can form co-occurrence networks, which are useful for tasks e.g. keyword extraction, document classification, or topic modeling.

– Ngoài ra, hãy cân nhắc xem dữ liệu có bao gồm các thực thể được kết nối với nhau thông qua các tham chiếu chung hay các mẫu đồng hiện diện hay không. Dữ liệu tài liệu & văn bản có thể không gợi ý ngay lập tức một cấu trúc đồ thị, nhưng nếu các tài liệu trích dẫn lẫn nhau hoặc có chung chủ đề hoặc tác giả, chúng có thể được biểu diễn dưới dạng các nút trong đồ thị, với các cạnh phản ánh các mối quan hệ này. Tương tự, các thuật ngữ trong tài liệu có thể tạo thành các mạng đồng hiện diện, hữu ích cho các tác vụ như trích xuất từ khóa, phân loại tài liệu hoặc mô hình hóa chủ đề.

By identifying these key indicators in your data, you can uncover hidden or implicit relationships that can be represented explicitly through graphs. Such representations allow for more advanced analyses using GNNs, which can effectively capture & model these relationships, leading to more accurate predictions & deeper insights into data.

– Bằng cách xác định các chỉ số chính này trong dữ liệu, bạn có thể khám phá các mối quan hệ ẩn hoặc ngầm định có thể

được biểu diễn rõ ràng thông qua biểu đồ. Các biểu diễn như vậy cho phép phân tích nâng cao hơn bằng cách sử dụng GNN, có thể nắm bắt hiệu quả & mô hình hóa các mối quan hệ này, dẫn đến dự đoán chính xác hơn & hiểu biết sâu sắc hơn về dữ liệu.

∗ 1.4.2. High dimensionality & sparsity. Graph-based models are particularly effective in handling high-dimensional data where many features may be sparse or missing. These models excel in situations where there are underlying structure connecting sparse entities, allowing for more meaningful analysis & improved performance.

– Các mô hình dựa trên đồ thị đặc biệt hiệu quả trong việc xử lý dữ liệu đa chiều, trong đó nhiều đặc điểm có thể thưa thớt hoặc bị thiếu. Các mô hình này đặc biệt hiệu quả trong các tình huống có cấu trúc cơ bản kết nối các thực thể thưa thớt, cho phép phân tích có ý nghĩa hơn & cải thiện hiệu suất.

Key indicators. To determine if your problem involves high-dimensional & sparse data suitable for GNNs, consider whether your dataset contains numerous entities with limited direct interactions or relationships. E.g., in recommender systems, user-item interaction data may appear tabular, but it is inherently sparse – most users only interact with a small subset of available items. By representing users & items as nodes & representing their interactions (e.g., purchases or clicks) as edges, GNNs can exploit network effects to make more accurate recommendations. These models can also address cold-start problem by uncovering both explicit & implicit relationships, leading to better performance in recommending new items to users or engaging new users with existing items.

– **Các chỉ số chính.** Để xác định xem vấn đề của bạn có liên quan đến dữ liệu đa chiều & thưa thớt phù hợp với GNN hay không, hãy xem xét liệu tập dữ liệu của bạn có chứa nhiều thực thể với các tương tác hoặc mối quan hệ trực tiếp hạn chế hay không. Ví dụ: trong các hệ thống đề xuất, dữ liệu tương tác giữa người dùng & sản phẩm có thể xuất hiện dưới dạng bảng, nhưng bản chất của nó là thưa thớt – hầu hết người dùng chỉ tương tác với một tập hợp con nhỏ các sản phẩm khả dụng. Bằng cách biểu diễn người dùng & sản phẩm dưới dạng các nút & biểu diễn các tương tác của họ (ví dụ: mua hàng hoặc nhấp chuột) dưới dạng các cạnh, GNN có thể khai thác hiệu ứng mạng để đưa ra các đề xuất chính xác hơn. Các mô hình này cũng có thể giải quyết vấn đề khởi động nguội bằng cách khám phá cả các mối quan hệ rõ ràng & ngầm định, dẫn đến hiệu suất tốt hơn trong việc đề xuất sản phẩm mới cho người dùng hoặc thu hút người dùng mới sử dụng các sản phẩm hiện có.

Another indicator that your problem may be suitable for graph-based models is when data represents entities that are sparsely connected but share significant characteristics. In drug discovery, e.g., molecules are represented as graphs, with atoms as nodes & chemical bonds as edges. This representation captures inherent sparsity of molecular structures, where most atoms form only a few bonds, & large portions of molecule may be distant from each other in graph. Traditional ML methods often struggle to predict properties of new molecules due to this sparsity, as they don't account for full structural context.

– Một dấu hiệu khác cho thấy vấn đề của bạn có thể phù hợp với các mô hình dựa trên đồ thị là khi dữ liệu biểu diễn các thực thể được kết nối thưa thớt nhưng có chung các đặc điểm quan trọng. Ví dụ, trong khám phá thuốc, các phân tử được biểu diễn dưới dạng đồ thị, với các nguyên tử là nút & các liên kết hóa học là cạnh. Cách biểu diễn này nắm bắt được tính thưa thớt vốn có của các cấu trúc phân tử, trong đó hầu hết các nguyên tử chỉ tạo thành một vài liên kết, & các phần lớn phân tử có thể nằm cách xa nhau trên đồ thị. Các phương pháp ML truyền thống thường gặp khó khăn trong việc dự đoán các đặc tính của các phân tử mới do tính thưa thớt này, vì chúng không tính đến toàn bộ bối cảnh cấu trúc.

Graph-based models, particularly GNNs, overcome these challenges by capturing both local atomic environments & global molecular structures. GNNs learn hierarchical features from fine-grained atomic interactions to broader molecular properties, & their ability to remain invariant to ordering of atoms ensures consistent predictions. By using graph structure of molecules, GNNs make accurate predictions from sparse, connected data, thereby accelerating drug discovery process.

– Các mô hình dựa trên đồ thị, đặc biệt là GNN, khắc phục những thách thức này bằng cách nắm bắt cả môi trường nguyên tử cục bộ & cấu trúc phân tử toàn cục. GNN học các đặc điểm phân cấp từ các tương tác nguyên tử chi tiết đến các đặc tính phân tử rộng hơn, & khả năng duy trì tính bất biến theo thứ tự nguyên tử đảm bảo các dự đoán nhất quán. Bằng cách sử dụng cấu trúc đồ thị của phân tử, GNN đưa ra các dự đoán chính xác từ dữ liệu thưa thớt và kết nối, do đó đẩy nhanh quá trình khám phá thuốc.

By recognizing these key indicators in your data, you can identify situations where graph-based models can effectively handle high-dimensional & sparse datasets. Representing such data as graphs allows GNNs to capture & use underlying structures, resulting in more accurate predictions & deeper insights across various applications.

– Bằng cách nhận diện các chỉ số chính này trong dữ liệu, bạn có thể xác định các tình huống mà mô hình dựa trên đồ thị có thể xử lý hiệu quả các tập dữ liệu đa chiều & thưa thớt. Việc biểu diễn dữ liệu dưới dạng đồ thị cho phép GNN nắm bắt & sử dụng các cấu trúc cơ bản, mang lại dự đoán chính xác hơn & hiểu biết sâu sắc hơn trên nhiều ứng dụng khác nhau.

∗ 1.4.3. Complex, nonlocal interactions. Certain problems require underlying how distant elements in a dataset influence each other. In these cases, GNNs provide a framework to capture these complex interactions, where predicted value or label of a particular data point depends not just on features of its immediate neighbors but also on those of other related data points. This capability is especially useful when relationships extend beyond direct connections to involve multiple levels or degrees of separation.

– Một số vấn đề đòi hỏi phải hiểu rõ cách các phần tử ở xa trong một tập dữ liệu ảnh hưởng lẫn nhau. Trong những trường hợp này, mạng nơ-ron nhân tạo (GNN) cung cấp một khuôn khổ để nắm bắt những tương tác phức tạp này, trong đó giá trị dự đoán hoặc nhãn của một điểm dữ liệu cụ thể không chỉ phụ thuộc vào các đặc điểm của các điểm lân cận mà còn phụ thuộc vào các đặc điểm của các điểm dữ liệu liên quan khác. Khả năng này đặc biệt hữu ích khi các mối quan

hệ vượt ra ngoài các kết nối trực tiếp, bao gồm nhiều cấp độ hoặc mức độ phân tách.

However, some standard GNNs, which rely primarily on local message passing, may struggle to capture long-range dependencies effectively. Advanced architectures or modifications, e.g. those incorporating global attention, nonlocal aggregation, or hierarchical message-passing, can be better address these challenges [14].

Key indicators. To determine if your problem involves complex, nonlocal interactions suitable for GNNs, consider whether outcome or behavior of 1 entity depends on attributes or actions of identities that are not directly connected to it but may be indirectly connected through other entities. E.g., in supply chain optimization, a delay in 1 supplier may not only affect its immediate downstream customers but could cascade through multiple levels of network, influencing distributors & final consumers.

– **Các chỉ số chính.** Để xác định xem vấn đề của bạn có liên quan đến các tương tác phức tạp, phi cục bộ phù hợp với GNN hay không, hãy xem xét liệu kết quả hoặc hành vi của 1 thực thể có phụ thuộc vào các thuộc tính hoặc hành động của các danh tính không được kết nối trực tiếp với nó nhưng có thể được kết nối gián tiếp thông qua các thực thể khác hay không. Ví dụ, trong tối ưu hóa chuỗi cung ứng, sự chậm trễ của 1 nhà cung cấp không chỉ ảnh hưởng đến các khách hàng hạ nguồn trực tiếp mà còn có thể lan truyền qua nhiều cấp độ mạng lưới, ảnh hưởng đến các nhà phân phối & người tiêu dùng cuối cùng.

Another indicator is whether problem involves scenarios where information, influence, or effects propagate through a network over time. In healthcare & epidemiology, e.g., a disease outbreak might spread from a small cluster of patients through their interactions with shared healthcare providers, common environments, or overlapping social networks. Such propagation requires an approach that captures indirect transmission pathways of information or effects.

– Một chỉ báo khác là liệu vấn đề có liên quan đến các kịch bản mà thông tin, ảnh hưởng hoặc tác động lan truyền qua mạng lưới theo thời gian hay không. Trong chăm sóc sức khỏe & dịch tễ học, ví dụ, một đợt bùng phát dịch bệnh có thể lây lan từ một nhóm nhỏ bệnh nhân thông qua tương tác của họ với các nhà cung cấp dịch vụ chăm sóc sức khỏe chung, môi trường chung hoặc các mạng lưới xã hội chồng chéo. Sự lan truyền như vậy đòi hỏi một phương pháp tiếp cận nắm bắt các con đường truyền thông tin hoặc tác động gián tiếp.

To close this section, in determining whether your problem is a good candidate for a GNN, ask yourself these questions:

1. Are there implicit relationships or interdependencies in my data that I could model?
2. Do interactions between entities exhibit complex, nonlocal dependencies that go beyond immediate connections?
3. Is data high-dimensional & sparse, with a need to capture underlying relational structures?

If answer to any of these questions is yes, consider framing your problem as a graph & applying GNNs to unlock new insights & predictive capabilities.

– Để kết thúc phần này, khi xác định xem vấn đề của bạn có phù hợp để áp dụng mô hình mạng nơ-ron nhân tạo (GNN) hay không, hãy tự hỏi mình những câu hỏi sau:

1. Liệu có mối quan hệ ngầm định hoặc phụ thuộc lẫn nhau nào trong dữ liệu của tôi mà tôi có thể mô hình hóa không?
2. Liệu các tương tác giữa các thực thể có biểu hiện sự phụ thuộc phức tạp, phi cục bộ, vượt ra ngoài các kết nối tức thời không?
3. Liệu dữ liệu có đa chiều & thưa thớt, cần phải nắm bắt các cấu trúc quan hệ cơ bản không?

Nếu câu trả lời cho bất kỳ câu hỏi nào trong số này là có, hãy cân nhắc việc định hình vấn đề của bạn dưới dạng biểu đồ & áp dụng GNN để khám phá những hiểu biết mới & khả năng dự đoán.

- 1.5. Understanding how GNNs operate. In this section, explore how GNNs work, starting from initial collection of raw data to final deployment of trained models. Examine each step, highlighting processes of data handling, model building, & unique message-passing technique that sets GNNs apart from traditional DL models.
  - 1.5.1. Mental model for training a GNN. Our mental model covers data sourcing, graph representation, preprocessing, & model development workflow. Start with raw data & end up with a trained GNN model & its outputs. Fig. 1.14: Mental model of GNN project. Start with raw data, which is transformed into a graph data model that can be stored in a graph database or used in a graph processing system. From graph processing system (& some graph databases), exploratory data analysis & visualization can be done. Finally, for graph ML, data is preprocessed into a form that can be submitted for training illustrates & visualizes topics related to these stages, annotated with chaps in which these topics appear.

  – Mô hình tinh thần để huấn luyện GNN. Mô hình tinh thần của chúng tôi bao gồm việc tìm nguồn dữ liệu, biểu diễn đồ thị, tiền xử lý và quy trình phát triển mô hình. Bắt đầu với dữ liệu thô và kết thúc bằng một mô hình GNN đã được huấn luyện và kết quả của nó. Hình 1.14: Mô hình tinh thần của dự án GNN. Bắt đầu với dữ liệu thô, được chuyển đổi thành mô hình dữ liệu đồ thị có thể được lưu trữ trong cơ sở dữ liệu đồ thị hoặc được sử dụng trong hệ thống xử lý đồ thị. Từ hệ thống xử lý đồ thị (và một số cơ sở dữ liệu đồ thị), có thể thực hiện phân tích dữ liệu thăm dò và trực quan hóa. Cuối cùng, đối với học máy đồ thị, dữ liệu được tiền xử lý thành một biểu mẫu có thể được gửi để huấn luyện minh họa và trực quan hóa các chủ đề liên quan đến các giai đoạn này, được chú thích bằng các chương trong đó các chủ đề này xuất hiện.

  While not all workflows include every step or stage of this process, most will incorporate at least some elements. At different stages of a model development project, different parts of this process will typically be used. E.g., when *training* a model, data analysis & visualization may be needed to make design decisions, but when *deploying* a model, it may only be necessary to stream raw data & quickly preprocess it for ingestion into a model. Though this book touches on earlier stages in this mental model, bulk of book is focused on how to train different types of GNNs. When other topics are discussed, they serve to support this main focus.

  – Mặc dù không phải tất cả quy trình làm việc đều bao gồm mọi bước hoặc giai đoạn của quy trình này, nhưng hầu hết đều sẽ kết hợp ít nhất một số yếu tố. Ở các giai đoạn khác nhau của một dự án phát triển mô hình, các phần khác nhau

của quy trình này thường sẽ được sử dụng. Ví dụ: khi *training* một mô hình, phân tích dữ liệu & trực quan hóa có thể cần thiết để đưa ra quyết định thiết kế, nhưng khi *deployment* một mô hình, có thể chỉ cần truyền dữ liệu thô & xử lý nhanh dữ liệu trước để đưa vào mô hình. Mặc dù cuốn sách này đề cập đến các giai đoạn trước đó của mô hình tư duy này, phần lớn nội dung sách tập trung vào cách huấn luyện các loại GNN khác nhau. Khi các chủ đề khác được thảo luận, chúng sẽ hỗ trợ cho trọng tâm chính này.

Mental model shows core tasks of applying GNNs to ML problems, & we return to this process repeatedly through rest of book. Examine this diagram from end to end.

– Mô hình tinh thần cho thấy các nhiệm vụ cốt lõi của việc áp dụng mạng nơ-ron nhân tạo (GNN) vào các bài toán ML, & chúng ta sẽ quay lại quá trình này nhiều lần trong suốt phần còn lại của cuốn sách. Hãy xem xét sơ đồ này từ đầu đến cuối.

1st step in training a GNN in structuring this raw data into a graph format, if it is not already. This requires deciding which entities in data to represent as nodes & edges, as well as determining features to assign to them. Decision must also be made about data storage – whether to use a graph database, processing system, or other formats.

– Bước đầu tiên trong quá trình huấn luyện GNN là cấu trúc dữ liệu thô này thành định dạng đồ thị, nếu dữ liệu chưa được định dạng. Điều này đòi hỏi phải quyết định những thực thể nào trong dữ liệu sẽ được biểu diễn dưới dạng nút & cạnh, cũng như xác định các đặc điểm cần gán cho chúng. Quyết định cũng cần được đưa ra về lưu trữ dữ liệu – sử dụng cơ sở dữ liệu đồ thị, hệ thống xử lý hay các định dạng khác.

For ML, data must be preprocessed for training & inference, involving tasks e.g. sampling, batching, & splitting data into training, validation, & test sets. Throughout this book, use PyTorch Geometric (PyG), which offers specialized classes for preprocessing & data splitting while preserving graph's structure. Preprocessing is covered in most chaps, with more-in-depth explanations available in Appendix B.

– Đối với ML, dữ liệu phải được xử lý trước để huấn luyện & suy luận, bao gồm các tác vụ như lấy mẫu, xử lý hàng loạt, & chia dữ liệu thành các tập huấn luyện, xác thực, & kiểm tra. Trong suốt cuốn sách này, hãy sử dụng PyTorch Geometric (PyG), cung cấp các lớp chuyên biệt để xử lý trước & chia tách dữ liệu trong khi vẫn bảo toàn cấu trúc đồ thị. Phần lớn các chương đều đề cập đến tiền xử lý, với các giải thích chi tiết hơn có sẵn trong Phụ lục B.

After processing data, can then move on to model training. In this book, cover several architectures & training types:

· Chaps. 2–3 discuss convolutional GNNs, where 1st use a GCN layer to produce graph embeddings (Chap. 2) & then train a full GCN & GraphSAGE models (Chap. 3).

· Chap. 4 explains graph attention networks (GATs), which adds attention to our GNNs.

· Chap. 5 introduces GNNs for unsupervised & generative problems, where we train & use a variational graph autoencoder (VGAE).

· Chap. 6 then explores advanced concept of spatiotemporal GNNs, based on graphs that evolve over time. Train a neural relational inference (NRI) model, which combines an autoencoder structure with a RNN.

  Most of examples provided for GNNs mentioned so far are illustrated with code examples which use small-scale graphs that can fit into memory on a laptop or desktop computer.

· In Chap. 7, delve into strategies for handling data that exceeds processing capacity of a single machine.

· In Chap. 8, close with some considerations for graph & GNN projects, e.g. practical aspects of working with graph data, as well as how to convert nongraph data into a graph format.

– Sau khi xử lý dữ liệu, có thể chuyển sang huấn luyện mô hình. Trong cuốn sách này, chúng tôi sẽ đề cập đến một số kiến trúc & loại huấn luyện:

· Chương 2-3 thảo luận về mạng GNN tích chập, trong đó đầu tiên sử dụng một lớp GCN để tạo nhúng đồ thị (Chương 2) & sau đó huấn luyện một mô hình GCN đầy đủ & GraphSAGE (Chương 3).

· Chương 4 giải thích về mạng chú ý đồ thị (GAT), giúp tăng cường sự chú ý cho các GNN của chúng tôi.

· Chương 5 giới thiệu GNN cho các bài toán không giám sát & sinh, trong đó chúng tôi huấn luyện & sử dụng bộ mã hóa tự động đồ thị biến phân (VGAE).

· Chương 6 sau đó khám phá khái niệm nâng cao về mạng GNN không gian - thời gian, dựa trên các đồ thị phát triển theo thời gian. Huấn luyện một mô hình suy luận quan hệ thần kinh (NRI), kết hợp cấu trúc bộ mã hóa tự động với RNN.

  Hầu hết các ví dụ được cung cấp cho GNN đã đề cập cho đến nay đều được minh họa bằng các ví dụ mã sử dụng đồ thị quy mô nhỏ, có thể chứa trong bộ nhớ của máy tính xách tay hoặc máy tính để bàn.

· Trong Chương 7, hãy đi sâu vào các chiến lược xử lý dữ liệu vượt quá khả năng xử lý của một máy tính.

· Trong Chương 8, hãy kết thúc bằng một số cân nhắc cho các dự án đồ thị & GNN, ví dụ: các khía cạnh thực tế khi làm việc với dữ liệu đồ thị, cũng như cách chuyển đổi dữ liệu không phải đồ thị sang định dạng đồ thị.

∗ 1.5.2. Unique mechanisms of a GNN model. Although there are a variety of GNN architectures at this point, they all tackle same problem of dealing with graph data in a way that is permutation invariant. They do this via encoding & exchanging information across graph structure during learning process.

– Cơ chế độc đáo của mô hình GNN. Mặc dù hiện tại có nhiều kiến trúc GNN khác nhau, tất cả đều giải quyết cùng một vấn đề là xử lý dữ liệu đồ thị theo cách bất biến hoán vị. Chúng thực hiện điều này thông qua việc mã hóa & trao đổi thông tin trên toàn bộ cấu trúc đồ thị trong quá trình học.

In a conventional neural network CNN, 1st need to initialize a set of parameters & functions. These include number of layers, size of layers, learning rate, loss function, batch size, & other hyperparameters. (These are all treated in detail in other books on DL, so assume familiar with those terms). Once defined these features, then train our network by

iteratively updating weights of network, as shown in Fig. 1.15: Process for training a GNN, which is similar to training most other DL models.

– Trong một mạng nơ-ron nhân tạo thông thường (CNN), trước tiên cần khởi tạo một tập hợp các tham số & hàm. Chúng bao gồm số lớp, kích thước lớp, tốc độ học, hàm mất mát, kích thước lô, & các siêu tham số khác. (Tất cả những điều này đều được trình bày chi tiết trong các sách khác về DL, vì vậy hãy coi như bạn đã quen thuộc với các thuật ngữ đó). Sau khi xác định các đặc điểm này, hãy huấn luyện mạng bằng cách cập nhật trọng số của mạng theo từng bước, như thể hiện trong Hình 1.15: Quy trình huấn luyện GNN, tương tự như huấn luyện hầu hết các mô hình DL khác.

Explicitly, perform following steps:

1. Input our data.
2. Pass data through neural network layers that transform data according to parameters of layer & an activation rule.
3. Output a representation from final layer of network.
4. Backpropagation error, & adjust parameters accordingly.
5. Repeat these steps a fixed number of *epochs* (process by which data is passed forward & backward to train a neural network).

For tabular data, these steps are exactly as listed, as shown in Fig. 1.16: Comparison of (simple) non-GNN & GNN. GNNs have a layer that distributes data among its vertices. For graph-based or relational data, these steps are similar except that each epoch relates to 1 iteration of message passing.

– Thực hiện các bước sau một cách rõ ràng:

1. Nhập dữ liệu.
2. Truyền dữ liệu qua các lớp mạng nơ-ron để biến đổi dữ liệu theo các tham số của lớp & một quy tắc kích hoạt.
3. Xuất ra một biểu diễn từ lớp cuối cùng của mạng.
4. Lỗi lan truyền ngược, & điều chỉnh các tham số cho phù hợp.
5. Lặp lại các bước này với số lượng cố định *epoch* (quy trình mà dữ liệu được truyền tới & lùi để huấn luyện mạng nơ-ron).

Đối với dữ liệu dạng bảng, các bước này chính xác như được liệt kê trong Hình 1.16: So sánh (đơn giản) không phải GNN & GNN. GNN có một lớp phân phối dữ liệu giữa các đỉnh của nó. Đối với dữ liệu dựa trên đồ thị hoặc dữ liệu quan hệ, các bước này tương tự nhau, ngoại trừ việc mỗi epoch liên quan đến 1 lần lặp truyền thông điệp.

∗ 1.5.3. Message passing. *Message passing*, which is touched on throughout book, is a central mechanism in GNNs that enables nodes to communicate & share information across a graph [15]. This process allows GNNs to learn rich, informative representations of graph-structured data, which is essential for tasks e.g. node classification, link prediction, & graph-level prediction. Fig. 1.17: Elements of our message passing layer. Each message passing layer consists of an aggregation, a transformation, & an update step: 1. Input initial graph with node, edges, & features. 2. Collect all features from neighboring nodes, known as messages, for each node. 3. Aggregate messages using invariant functions e.g. sum, max, or mean. 4. Transform messages using a neural network to create new node features. 5. Update all features in graph with new node features. illustrates steps involved in typical message-passing layer.

– Truyền tin nhắn. *Truyền tin nhắn*, được đề cập trong toàn bộ cuốn sách, là một cơ chế trung tâm trong GNN cho phép các nút giao tiếp & chia sẻ thông tin trên một đồ thị [15]. Quá trình này cho phép GNN học các biểu diễn phong phú, nhiều thông tin về dữ liệu có cấu trúc đồ thị, điều này rất cần thiết cho các tác vụ ví dụ như phân loại nút, dự đoán liên kết, & dự đoán cấp đồ thị. Hình 1.17: Các thành phần của lớp truyền tin nhắn của chúng tôi. Mỗi lớp truyền tin nhắn bao gồm một phép tổng hợp, một phép biến đổi, & một bước cập nhật: 1. Đầu vào đồ thị ban đầu với nút, cạnh, & các đặc điểm. 2. Thu thập tất cả các đặc điểm từ các nút lân cận, được gọi là các thông điệp, cho mỗi nút. 3. Tổng hợp các thông điệp bằng các hàm bất biến ví dụ như tổng, cực đại hoặc trung bình. 4. Biến đổi các thông điệp bằng mạng nơ-ron để tạo các đặc điểm nút mới. 5. Cập nhật tất cả các đặc điểm trong đồ thị bằng các đặc điểm nút mới. minh họa các bước liên quan đến lớp truyền tin nhắn thông thường.

Message-passing process begins with Input (step 1) of initial graph, where every node & edge have their own features. In Collect step (step 2), each node gathers information from its immediate neighbors – these pieces of information are referred to as "messages". This step ensures that each node has access to features of its neighbors, which are crucial for understanding local graph structure. Next, in Aggregate step (step 3), collected messages from neighboring nodes are combined using an invariant function, e.g. sum, mean, or max. This aggregation consolidates information from a node's neighborhood into a single vector, capturing most relevant details about its local environment.

– Quá trình truyền thông điệp bắt đầu với Đầu vào (bước 1) của đồ thị khởi tạo, trong đó mỗi nút & cạnh đều có các đặc trưng riêng. Trong bước Thu thập (bước 2), mỗi nút thu thập thông tin từ các nút lân cận trực tiếp của nó – những thông tin này được gọi là "thông điệp". Bước này đảm bảo rằng mỗi nút có quyền truy cập vào các đặc trưng của các nút lân cận, điều này rất quan trọng để hiểu cấu trúc đồ thị cục bộ. Tiếp theo, trong bước Tổng hợp (bước 3), các thông điệp được thu thập từ các nút lân cận được kết hợp bằng một hàm bất biến, ví dụ: tổng, trung bình hoặc tối đa. Quá trình tổng hợp này hợp nhất thông tin từ vùng lân cận của một nút thành một vectơ duy nhất, nắm bắt các chi tiết quan trọng nhất về môi trường cục bộ của nó.

In Transform step (step 4), aggregated messages are processed by a neural network to produce a new representation for each node. This transformation allows GNN to learn complex interactions & patterns within graph by applying nonlinear functions to aggregated information.

– Trong bước Biến đổi (bước 4), các thông điệp tổng hợp được xử lý bởi mạng nơ-ron để tạo ra một biểu diễn mới cho mỗi nút. Phép biến đổi này cho phép GNN học các tương tác phức tạp & các mẫu trong đồ thị bằng cách áp dụng các

hàm phi tuyến tính vào thông tin tổng hợp.
Finally, during Update step (step 5), features of each node in graph are replaced or updated with these new representations. This completes 1 round of message passing, incorporating information from neighboring nodes to refine each node's features.
– Cuối cùng, trong bước Cập nhật (bước 5), các đặc trưng của mỗi nút trong đồ thị được thay thế hoặc cập nhật bằng các biểu diễn mới này. Điều này hoàn tất 1 vòng truyền thông điệp, kết hợp thông tin từ các nút lân cận để tinh chỉnh các đặc trưng của từng nút.
Each message-passing layer in a GNN allows nodes to gather information from nodes that are further away, or more "hops" away, in graph. Repeating these steps over multiple layers enables GNN to capture more complex dependencies & long-range interactions within graph.
– Mỗi lớp truyền thông điệp trong GNN cho phép các nút thu thập thông tin từ các nút ở xa hơn, hoặc cách xa hơn "các bước nhảy", trong đồ thị. Việc lặp lại các bước này trên nhiều lớp cho phép GNN nắm bắt các mối quan hệ phụ thuộc phức tạp hơn & các tương tác tầm xa trong đồ thị.
By using message passing, GNNs efficiently encode graph structure & data into useful representations for a variety of downstream tasks. Advanced architectures, e.g. those incorporating global attention or hierarchical message passing, further enhance model's ability to capture long-range dependencies across graph, enabling more robust performance on diverse applications.
– Bằng cách sử dụng kỹ thuật truyền thông điệp, GNN mã hóa hiệu quả cấu trúc đồ thị & dữ liệu thành các biểu diễn hữu ích cho nhiều tác vụ hạ nguồn. Các kiến trúc tiên tiến, ví dụ như kiến trúc tích hợp sự chú ý toàn cục hoặc truyền thông điệp phân cấp, sẽ nâng cao hơn nữa khả năng của mô hình trong việc nắm bắt các phụ thuộc tầm xa trên toàn bộ đồ thị, cho phép hiệu suất mạnh mẽ hơn trên nhiều ứng dụng khác nhau.

- Summary.
  - GNNs are specialized tools for handling relational, or relationship-centric, data, particularly in scenarios where traditional neural networks struggle due to complexity & diversity of graph structures.
    – GNN là công cụ chuyên dụng để xử lý dữ liệu quan hệ hoặc dữ liệu tập trung vào mối quan hệ, đặc biệt là trong các tình huống mà mạng nơ-ron truyền thống gặp khó khăn do tính phức tạp & đa dạng của cấu trúc đồ thị.
  - GNNs have found significant applications in areas e.g. recommendation engines, drug discovery, & mechanical reasoning, showcasing their versatility in handling large & complex relational data for enhanced insights & predictions.
    – GNN đã tìm thấy những ứng dụng quan trọng trong các lĩnh vực như công cụ đề xuất, khám phá thuốc, & suy luận cơ học, thể hiện tính linh hoạt của chúng trong việc xử lý dữ liệu quan hệ phức tạp & lớn để có được những hiểu biết sâu sắc & dự đoán tốt hơn.
  - Specific GNN tasks include node prediction, edge prediction, graph prediction, & graph representation through embedding techniques.
    – Các nhiệm vụ cụ thể của GNN bao gồm dự đoán nút, dự đoán cạnh, dự đoán đồ thị và biểu diễn đồ thị thông qua các kỹ thuật nhúng.
  - Specific GNN tasks include node prediction, edge prediction, graph prediction, & graph representation through embedding techniques.
    – Các nhiệm vụ cụ thể của GNN bao gồm dự đoán nút, dự đoán cạnh, dự đoán đồ thị và biểu diễn đồ thị thông qua các kỹ thuật nhúng.
  - GNNs are best used when data is represented as a graph, indicating a strong emphasis on relationships & connections between data points. They are not ideal for individual, standalone data entries where relational information is insignificant.
    – GNN được sử dụng tốt nhất khi dữ liệu được biểu diễn dưới dạng đồ thị, thể hiện sự nhấn mạnh vào mối quan hệ & kết nối giữa các điểm dữ liệu. Chúng không lý tưởng cho các mục dữ liệu riêng lẻ, độc lập, nơi thông tin quan hệ không đáng kể.
  - When deciding if a GNN solution is a good fit for your problem, consider cases that have characteristics e.g. implicit relationships, high-dimensionality, sparsity, & complex nonlocal interactions. By understanding these fundamentals, practitioners can evaluate suitability of GNNs for their specific problems, implement them effectively, & recognize their tradeoffs & limitations in real-world applications.
    – Khi quyết định xem giải pháp GNN có phù hợp với vấn đề của bạn hay không, hãy xem xét các trường hợp có đặc điểm như mối quan hệ ngầm định, đa chiều, thưa thớt, & tương tác phi cục bộ phức tạp. Bằng cách hiểu những nguyên tắc cơ bản này, người thực hành có thể đánh giá tính phù hợp của GNN cho các vấn đề cụ thể, triển khai chúng một cách hiệu quả, & nhận ra những đánh đổi & hạn chế của chúng trong các ứng dụng thực tế.
  - Messages passing is a core mechanism of GNN,s which enables them to encode & exchange information across a graph's structure, allowing for meaningful node, edge, & graph-level predictions. Each layer of a GNN represents 1 step of message passing, with various aggregation functions to combine messages effectively, providing insights & representations useful for ML tasks.
    – Truyền thông điệp là một cơ chế cốt lõi của GNN, cho phép chúng mã hóa & trao đổi thông tin trên toàn bộ cấu trúc đồ thị, cho phép đưa ra các dự đoán có ý nghĩa ở cấp độ nút, cạnh và đồ thị. Mỗi lớp của GNN đại diện cho 1 bước truyền thông điệp, với nhiều hàm tổng hợp khác nhau để kết hợp các thông điệp một cách hiệu quả, cung cấp thông tin chi tiết & biểu diễn hữu ích cho các tác vụ ML.

- 2. Graph embeddings. Covers:

1. Exploring graph embeddings & their importance
2. Creating node embeddings using non-GNN & GNN methods
3. Comparing node embeddings on a semi-supervised problem
4. Taking a deeper dive into embedding methods

Graph embeddings are essential tools in graph-based ML. They transform intricate structure of graphs – be it the entire graph, individual nodes, or edges – into a more manageable, lower-dimensional space. Do this to compress a complex dataset into a form that is easier to work with, without losing its inherent patterns & relationships, information to which we will apply a GNN or other ML method.

– Bao gồm:

1. Khám phá nhúng đồ thị & tầm quan trọng của chúng
2. Tạo nhúng nút bằng phương pháp không phải GNN & GNN
3. So sánh nhúng nút trong bài toán bán giám sát
4. Đi sâu hơn vào các phương pháp nhúng

Nhúng đồ thị là công cụ thiết yếu trong học máy dựa trên đồ thị. Chúng chuyển đổi cấu trúc phức tạp của đồ thị – có thể là toàn bộ đồ thị, từng nút hoặc cạnh – thành một không gian ít chiều hơn, dễ quản lý hơn. Thực hiện điều này để nén một tập dữ liệu phức tạp thành một dạng dễ làm việc hơn, mà không làm mất các mẫu & mối quan hệ vốn có của nó, thông tin mà chúng ta sẽ áp dụng GNN hoặc phương pháp học máy khác.

Graphs encapsulate relationships & interactions within networks, whether they are social networks, biological networks, or any system where entities are interconnected. Embeddings capture these real-life relationships in a compact from, facilitating tasks e.g. visualization, clustering, or predictive modeling.

– Đồ thị gói gọn các mối quan hệ & tương tác trong các mạng lưới, dù là mạng xã hội, mạng sinh học hay bất kỳ hệ thống nào mà các thực thể được kết nối với nhau. Nhúng nắm bắt các mối quan hệ thực tế này một cách cô đọng, tạo điều kiện thuận lợi cho các tác vụ như trực quan hóa, phân cụm hoặc mô hình hóa dự đoán.

There are numerous strategies to derive these embeddings, each with its unique approach & application: from classical graph algorithms that use network's topology, to linear algebra techniques that decompose matrices representing graph, & more advanced methods e.g. GNNs [1]. GNNs stand out because they can integrate embedding process directly into learning algorithm itself.

– Có nhiều chiến lược để tạo ra các nhúng này, mỗi chiến lược có cách tiếp cận & ứng dụng riêng: từ các thuật toán đồ thị cổ điển sử dụng cấu trúc mạng, đến các kỹ thuật đại số tuyến tính phân tích các ma trận biểu diễn đồ thị, & các phương pháp tiên tiến hơn, ví dụ như GNN [1]. GNN nổi bật vì chúng có thể tích hợp quá trình nhúng trực tiếp vào chính thuật toán học.

In traditional ML workflows, embeddings are generated as a separate step, serving as a dimensionality-reduction technique in tasks e.g. regression or classification. However, GNNs merge embedding generation with model's learning process. As network processes inputs through its layers, embeddings are refined & updated, making learning phase & embedding phase inseparable. I.e., GNNs learn most informative representative of graph data during training time.

– Trong quy trình làm việc ML truyền thống, nhúng được tạo ra như một bước riêng biệt, đóng vai trò là kỹ thuật giảm chiều trong các tác vụ như hồi quy hoặc phân loại. Tuy nhiên, GNN kết hợp việc tạo nhúng với quy trình học của mô hình. Khi mạng xử lý dữ liệu đầu vào qua các lớp của nó, các nhúng được tinh chỉnh & cập nhật, khiến giai đoạn học & giai đoạn nhúng trở nên không thể tách rời. Tức là, GNN học dữ liệu biểu diễn đồ thị mang tính thông tin nhất trong thời gian đào tạo.

Using graph embeddings can significantly enhance your DS & ML projects, especially when dealing with complex networked data. By capturing essence of graph in a lower-dimensional space, embeddings make it feasible to apply a variety of other ML techniques to graph data, opening up a world of possibilities for analysis & model building.

– Việc sử dụng nhúng đồ thị có thể cải thiện đáng kể các dự án DS & ML của bạn, đặc biệt là khi xử lý dữ liệu mạng phức tạp. Bằng cách nắm bắt bản chất của đồ thị trong không gian ít chiều hơn, nhúng cho phép áp dụng nhiều kỹ thuật ML khác vào dữ liệu đồ thị, mở ra một thế giới khả thi cho việc phân tích & xây dựng mô hình.

In this chap, begin with an introduction to graph embeddings & a case study on a graph of political book purchases. Start with Node2Vec (N2V) to establish a baseline with a non-GNN approach, guiding you through its practical application. In Sect. 2.2, shift to GNNs, offering a hands-on introduction to GNN-based embeddings, including setup, preprocessing, & visualization. Sect. 2.3 provides a comparative analysis of N2V & GNN embeddings, highlighting their applications. Chap then rounds off with a discussion of theoretical aspects of these embedding methods, with a special focus on principles behind N2V & message-passing mechanism in GNNs. Process we take in this chap is illustrated in Fig. 2.1: Summary of process & objectives in Chap. 2: 1. Preprocess Political Books dataset for embedding. 2. Use N2V & GCN to create embeddings from preprocessed data. 3. Prepare N2V embeddings & GCN embeddings for semi-supervised classification. 4. Embeddings are used as features in a random forest classifier (tabular features) &a GCN classifier (node features).

– Trong chương này, hãy bắt đầu bằng phần giới thiệu về nhúng đồ thị & nghiên cứu điển hình về đồ thị mua sách chính trị. Bắt đầu với Node2Vec (N2V) để thiết lập đường cơ sở với phương pháp không phải GNN, hướng dẫn bạn ứng dụng thực tế. Trong Phần 2.2, chuyển sang GNN, cung cấp phần giới thiệu thực hành về nhúng dựa trên GNN, bao gồm thiết lập, tiền xử

lý, & trực quan hóa. Phần 2.3 cung cấp phân tích so sánh về nhúng N2V & GNN, làm nổi bật các ứng dụng của chúng. Sau đó, chương kết thúc bằng phần thảo luận về các khía cạnh lý thuyết của các phương pháp nhúng này, đặc biệt tập trung vào các nguyên tắc đằng sau cơ chế truyền tin nhắn N2V & trong GNN. Quy trình chúng tôi thực hiện trong chương này được minh họa trong Hình 2.1: Tóm tắt các mục tiêu của quy trình & trong Chương 2: 1. Tiền xử lý tập dữ liệu Sách chính trị để nhúng. 2. Sử dụng N2V & GCN để tạo nhúng từ dữ liệu đã được xử lý trước. 3. Chuẩn bị nhúng N2V & nhúng GCN cho phân loại bán giám sát. 4. Nhúng được sử dụng làm các đặc trưng trong bộ phân loại rừng ngẫu nhiên (các đặc trưng dạng bảng) & bộ phân loại GCN (các đặc trưng dạng nút).

- 2.1. Creating embeddings with Node2Vec. Understanding relationships within a network is a core task in many fields, from social network analysis to biology & recommendation systems. In this section, explor how to create node embeddings using `Node2Vec (N2V)`, a technique inspired by `Word2Vec` from NLP [2]. N2V captures context of nodes within a graph by simulating random walks, allowing us to understand neighborhood relationships between nodes in a low-dimensional space. This approach is effective for identifying patterns, clustering similar nodes, & preparing data for ML tasks.

  – Tạo nhúng với Node2Vec. Hiểu các mối quan hệ trong mạng là một nhiệm vụ cốt lõi trong nhiều lĩnh vực, từ phân tích mạng xã hội đến sinh học & hệ thống khuyến nghị. Trong phần này, hãy khám phá cách tạo nhúng nút bằng `Node2Vec (N2V)`, một kỹ thuật lấy cảm hứng từ `Word2Vec` trong NLP [2]. N2V nắm bắt ngữ cảnh của các nút trong đồ thị bằng cách mô phỏng các bước ngẫu nhiên, cho phép chúng ta hiểu các mối quan hệ lân cận giữa các nút trong không gian ít chiều. Phương pháp này hiệu quả trong việc xác định các mẫu, phân cụm các nút tương tự và chuẩn bị dữ liệu cho các tác vụ ML.

  To make this process accessible, use `Node2Vec` Python library, which is beginner-friendly, although it may be slower on larger graphs. N2V helps create embeddings that capture structural relationships between nodes, which we can then visualize to uncover insights about graph's structure. Our workflow involves several steps:

  1. *Load data & set N2V parameters.* Start by loading our graph data & initializing N2V with specific parameters to control random walks, e.g. walk length & number of walks per node.
  2. *Create embeddings.* N2V generates node embeddings by performing random walks on graph, effective summarizing each node's local neighborhood into a vector format.
  3. *Transform embeddings.* Resulting embeddings are saved & then transformed into a format suitable for visualization.
  4. *Visualize embeddings in 2D.* Use UMAP, a dimensionality reduction technique, to project these embeddings into 2D, making it easier to visualize & interpret results.

  – Để quá trình này dễ hiểu hơn, hãy sử dụng thư viện Python `Node2Vec`, thư viện này thân thiện với người mới bắt đầu, mặc dù có thể chậm hơn trên các đồ thị lớn hơn. N2V giúp tạo các nhúng nắm bắt mối quan hệ cấu trúc giữa các nút, sau đó chúng ta có thể trực quan hóa để khám phá những hiểu biết sâu sắc về cấu trúc của đồ thị. Quy trình làm việc của chúng tôi bao gồm một số bước:

  1. *Tải dữ liệu & đặt tham số N2V.* Bắt đầu bằng cách tải dữ liệu đồ thị & khởi tạo N2V với các tham số cụ thể để kiểm soát các bước ngẫu nhiên, ví dụ: độ dài bước & số lần bước trên mỗi nút.
  2. *Tạo nhúng.* N2V tạo nhúng nút bằng cách thực hiện các bước ngẫu nhiên trên đồ thị, tóm tắt hiệu quả vùng lân cận cục bộ của mỗi nút thành định dạng vector.
  3. *Biến đổi nhúng.* Các nhúng kết quả được lưu & sau đó được chuyển đổi thành định dạng phù hợp để trực quan hóa.
  4. *Hình dung các nhúng trong 2D.* Sử dụng UMAP, một kỹ thuật giảm chiều, để chiếu các nhúng này vào 2D, giúp hình dung & diễn giải kết quả dễ dàng hơn.

  Our data is Political Books dataset, which comprises books (nodes) connected by frequent co-purchases on Amazon.com during 2004 US election period (edges) [3]. Using this dataset provides a compelling example of how N2V can reveal underlying patterns in co-purchasing behavior, potentially reflecting broader ideological groupings among book buyers [4]. Table 2.1: Overview of Political Books dataset. provides key information about Political Books graph. The Political Books dataset contains the following:

  1. Nodes: represent books about US politics sold by Amazon.com.
  2. Edges: indicate frequent co-purchasing by same buyers, as suggested by Amazon's "customers who bought this book also bought these other books" feature.

  In Fig. 2.2: Graph visualization of Political Books dataset. Right-leaning books (nodes) are in a lighter shade & are clustered in top half of figure, left-leaning are darker shaded circles & clustered in lower half of figure, & neural political stance are dark squares & appear in middle. When 2 nodes are connected, it indicates that they have been purchased together frequently on Amazon.com., books are shaded based on their political alignment – darker shade for liberal, lighter shade for conservative, & striped for neural. Categories were assigned by Mark Newman through a qualitative analysis of book descriptions & reviews posted on Amazon.

  – Dữ liệu của chúng tôi là tập dữ liệu Sách Chính trị, bao gồm các cuốn sách (nút) được kết nối bởi các giao dịch mua chung thường xuyên trên Amazon.com trong giai đoạn bầu cử Hoa Kỳ năm 2004 (cạnh) [3]. Việc sử dụng tập dữ liệu này cung cấp một ví dụ thuyết phục về cách N2V có thể tiết lộ các mô hình cơ bản trong hành vi mua chung, có khả năng phản ánh các nhóm ý thức hệ rộng hơn giữa những người mua sách [4]. Bảng 2.1: Tổng quan về tập dữ liệu Sách Chính trị. cung cấp thông tin chính về biểu đồ Sách Chính trị. Tập dữ liệu Sách Chính trị chứa các thông tin sau:

  1. Nút: biểu diễn các cuốn sách về chính trị Hoa Kỳ do Amazon.com bán.
  2. Cạnh: biểu thị việc mua chung thường xuyên của cùng một người mua, như được gợi ý bởi tính năng "khách hàng đã mua cuốn sách này cũng đã mua những cuốn sách khác này" của Amazon.

Trong Hình 2.2: Biểu đồ trực quan của tập dữ liệu Sách Chính trị. Sách thiên hữu (nút) được tô sáng hơn & tập trung ở nửa trên của hình, sách thiên tả là các vòng tròn tô đậm hơn & tập trung ở nửa dưới của hình, & lập trường chính trị thần kinh là các ô vuông đậm & xuất hiện ở giữa. Khi 2 nút được kết nối, điều đó cho thấy chúng đã được mua cùng nhau thường xuyên trên Amazon.com. Sách được tô màu dựa trên khuynh hướng chính trị của chúng – tô đậm hơn cho khuynh hướng tự do, tô nhạt hơn cho khuynh hướng bảo thủ, & có sọc cho khuynh hướng thần kinh. Các danh mục được phân loại bởi Mark Newman thông qua phân tích định tính các mô tả sách & bài đánh giá được đăng trên Amazon.

This dataset, compiled by Valdis Krebs & available through GNN in Action repository or Carnegie Mellon University website, contains 105 books (nodes) & 441 edges (co-purchases). If want to learn more about background of this dataset, Krebs has written an article with this information [4].

– Bộ dữ liệu này, do Valdis Krebs biên soạn & có sẵn trên kho lưu trữ GNN in Action hoặc trang web của Đại học Carnegie Mellon, chứa 105 sách (nút) & 441 cạnh (mua chung). Nếu muốn tìm hiểu thêm về bối cảnh của bộ dữ liệu này, Krebs đã viết một bài báo với thông tin này [4].

Using N2V, aim to explore structure of this collection of books, uncovering insights based on political learnings & potential associations between different book categories. By visualizing embeddings created by N2V, can gain a better understanding of how books are grouped & which ones might share a common audience, providing valuable insights into consumer behavior during a politically changed period.

– Sử dụng N2V, chúng tôi hướng đến việc khám phá cấu trúc của bộ sưu tập sách này, tìm ra những hiểu biết sâu sắc dựa trên các bài học chính trị & những mối liên hệ tiềm năng giữa các thể loại sách khác nhau. Bằng cách trực quan hóa các phần nhúng do N2V tạo ra, chúng tôi có thể hiểu rõ hơn về cách sách được nhóm lại & những cuốn nào có thể có chung đối tượng độc giả, từ đó cung cấp những hiểu biết giá trị về hành vi người tiêu dùng trong giai đoạn biến động chính trị.

From visualization, data is already clustered in a logical way. This is thanks to *Kamada-Kawai algorithm* graph algorithm, which exploits topological data only without metadata & is useful for visualizing graph. This graph visualization technique positions nodes in a way that reflects their connections, aiming for an arrangement where closely connected nodes are near each other but less connected nodes are farther apart. It achieves this by treating nodes like points connected by springs, iteratively adjusting their positions until "tension" in springs is minimized. This results in a layout that naturally reveals clusters & relationships within graph based purely on its structure.

– Từ trực quan hóa, dữ liệu đã được phân cụm theo một cách logic. Điều này là nhờ thuật toán đồ thị *Kamada-Kawai*, thuật toán này chỉ khai thác dữ liệu tôpô mà không cần siêu dữ liệu & rất hữu ích cho việc trực quan hóa đồ thị. Kỹ thuật trực quan hóa đồ thị này định vị các nút theo cách phản ánh kết nối của chúng, hướng đến một sự sắp xếp trong đó các nút có kết nối chặt chẽ sẽ ở gần nhau nhưng các nút ít kết nối hơn sẽ ở xa nhau hơn. Nó đạt được điều này bằng cách coi các nút như các điểm được kết nối bởi lò xo, điều chỉnh vị trí của chúng theo từng bước cho đến khi "lực căng" trong lò xo được giảm thiểu. Điều này dẫn đến một bố cục tự nhiên thể hiện các cụm & mối quan hệ trong đồ thị chỉ dựa trên cấu trúc của nó.

For Political Books dataset, Kamada-Kawai algorithm helps us visualize books (nodes) based on how often they are co-purchased on Amazon, without using any external information e.g. political alignment or book titles. This gives us an initial view of how books are grouped together by buying behavior. In next steps, use methods e.g. N2V to create embeddings that capture more detailed patterns & further distinguish different book groups.

– Đối với tập dữ liệu Sách Chính trị, thuật toán Kamada-Kawai giúp chúng tôi trực quan hóa các cuốn sách (nút) dựa trên tần suất chúng được mua chung trên Amazon, mà không cần sử dụng bất kỳ thông tin bên ngoài nào, ví dụ như liên kết chính trị hoặc tiêu đề sách. Điều này cung cấp cho chúng tôi cái nhìn ban đầu về cách sách được nhóm lại với nhau theo hành vi mua. Trong các bước tiếp theo, hãy sử dụng các phương pháp, ví dụ như N2V, để tạo các nhúng nắm bắt các mẫu chi tiết hơn & phân biệt rõ hơn các nhóm sách khác nhau.

∗ 2.1.1. Loading data, setting parameters, & creating embeddings. Use `Node2Vec, NetworkX` libraries for our 1st hands-on encounter with graph embeddings. After installing these packages using `pip`, load our dataset's graph data, which is stored in `.gml` format (Graph Modeling Language, GML), using `NetworkX` library & generate embeddings with `Node2Vec` library.
– Đang tải dữ liệu, thiết lập tham số, & tạo nhúng. Sử dụng thư viện `Node2Vec, NetworkX` cho lần thực hành đầu tiên với nhúng đồ thị. Sau khi cài đặt các gói này bằng `pip`, hãy tải dữ liệu đồ thị của tập dữ liệu, được lưu trữ ở định dạng `.gml` (Ngôn ngữ Mô hình Đồ thị, GML), bằng thư viện `NetworkX` & tạo nhúng bằng thư viện `Node2Vec`.
GML is a simple, human-readable plain text file format used to represent graph structures. It stores information about nodes, edges, & their attributes in a structured way, making it easy to read & write graph data. E.g., a `.gml` file might contain a list of nodes (e.g., books in our dataset) & edges (connections representing co-purchases) along with additional properties e.g. labels or weights. This format is widely used for exchanging graph data between different software & tools. By loading `.gml` file with `NetworkX`, can easily manipulate & analyze graph in Python.
– GML là một định dạng tệp văn bản thuần túy đơn giản, dễ đọc, được sử dụng để biểu diễn các cấu trúc đồ thị. Nó lưu trữ thông tin về các nút, cạnh và thuộc tính của chúng theo một cấu trúc nhất định, giúp việc đọc và ghi dữ liệu đồ thị trở nên dễ dàng. Ví dụ: tệp `.gml` có thể chứa danh sách các nút (ví dụ: sách trong tập dữ liệu của chúng tôi) và các cạnh (các kết nối biểu diễn các giao dịch mua chung) cùng với các thuộc tính bổ sung, ví dụ: nhãn hoặc trọng số. Định dạng này được sử dụng rộng rãi để trao đổi dữ liệu đồ thị giữa các phần mềm và công cụ khác nhau. Bằng cách tải tệp `.gml` với `NetworkX`, bạn có thể dễ dàng thao tác và phân tích đồ thị trong Python.
In `Node2Vec` library's `Node2Vec` function, can use following parameters to specify calculations done & properties of output embedding:

· *Size of embedding* (`dimensions`): Think of this as how detailed each node's profile is, as in how many different traits

you are noting down. Standard detail level is 128 traits, but you can tweak this based on how complex you want each node's profile to be.

- *Length of each walk* (`Walk Length`): This is about how far each random walk through your graph goes, with 80 steps being usual journey. If want to see more of neighborhood around a node, increase this number.
- *Number of walks per node* (`Num Walks`): This tells us how many times we will take a walk starting from each node. Starting with 10 walks gives a good overview, but if you want a fuller picture of a node's surroundings, consider going on more walks.
- *Backtracking control (Return Parameter p)*: This setting helps decide if our walk should circle back to where it has been. Setting it at 1 keeps things balanced, but adjusting it can make your walks more or less exploratory.
- *Exploration Depth (In-Out Parameter q)*: This one is about choosing between taking in broader neighborhood scene (e.g.,a BFS with $q > 1$) or diving deep into specific paths (e.g., a DFS with $q < 1$), with 1 being a mix of both.

Adjust these settings based on what you are looking to understand about your nodes & their connections. Want more depth? Tweak exploration depth. Looking for broader context? Adjust walk length & number of walks. In addition, keep in mind: size of your embeddings should match level of detail you need. In general, it is a good idea to try different combinations of these parameters to see effect on embeddings.

– Trong hàm `Node2Vec` của thư viện `Node2Vec`, bạn có thể sử dụng các tham số sau để chỉ định các phép tính được thực hiện & các thuộc tính của nhúng đầu ra:

- `Kích thước nhúng` (`kích thước`): Hãy coi đây là mức độ chi tiết của hồ sơ từng nút, tức là số lượng đặc điểm khác nhau mà bạn đang ghi lại. Mức độ chi tiết tiêu chuẩn là 128 đặc điểm, nhưng bạn có thể điều chỉnh tùy theo độ phức tạp mà bạn muốn hồ sơ của mỗi nút đạt được.
- `Độ dài mỗi lần đi` (`Độ dài lần đi`): Đây là khoảng cách mà mỗi lần đi ngẫu nhiên qua đồ thị của bạn đi được, với 80 bước là hành trình thông thường. Nếu muốn xem thêm về vùng lân cận xung quanh một nút, hãy tăng con số này.
- `Số lần đi trên mỗi nút` (`Số lần đi`): Con số này cho chúng ta biết chúng ta sẽ đi bao nhiêu lần bắt đầu từ mỗi nút. Bắt đầu với 10 lần đi bộ sẽ cho một cái nhìn tổng quan tốt, nhưng nếu bạn muốn có cái nhìn đầy đủ hơn về môi trường xung quanh của một nút, hãy cân nhắc đi bộ nhiều lần hơn.
- *Điều khiển quay lui (Tham số trả về p)*: Thiết lập này giúp quyết định xem việc đi bộ của chúng ta có nên quay lại vị trí ban đầu hay không. Đặt giá trị này ở mức 1 sẽ giữ mọi thứ cân bằng, nhưng việc điều chỉnh nó có thể khiến việc đi bộ của bạn mang tính khám phá nhiều hơn hoặc ít hơn.
- *Độ sâu khám phá (Tham số vào-ra q)*: Thiết lập này liên quan đến việc lựa chọn giữa việc xem xét toàn cảnh khu vực lân cận rộng hơn (ví dụ: BFS có $q > 1$) hoặc đi sâu vào các đường dẫn cụ thể (ví dụ: DFS có $q < 1$), với 1 là sự kết hợp của cả hai.

Điều chỉnh các thiết lập này dựa trên những gì bạn muốn hiểu về các nút & kết nối của chúng. Muốn có thêm chiều sâu? Hãy tinh chỉnh độ sâu khám phá. Muốn có bối cảnh rộng hơn? Hãy điều chỉnh độ dài & số lần đi bộ. Ngoài ra, hãy nhớ rằng: kích thước nhúng phải phù hợp với mức độ chi tiết bạn cần. Nhìn chung, bạn nên thử kết hợp các thông số này theo nhiều cách khác nhau để xem hiệu quả của việc nhúng.

For this exercise, use 1st 4 parameters. Deeper details on these parameters are found in Sect. 2.4. Code in Listing 2.1: Generating N2V embeddings begins by loading graph into a variable called `book_graph`, using `read_gml` method from `NetworkX` library. Next, a N2V `model` is initialized with loaded graph. This model is set up with specific parameters: it will create 64-dimensional embeddings for each node, use walks of 30 steps along, perform 200 walks starting from each node to gather context, & run these operations in parallel across 4 workers to speed up process.

– Với bài tập này, hãy sử dụng 4 tham số đầu tiên. Chi tiết sâu hơn về các tham số này được tìm thấy trong Mục 2.4. Mã trong Liệt kê 2.1: Tạo nhúng N2V bắt đầu bằng cách tải đồ thị vào một biến có tên là `book_graph`, sử dụng phương thức `read_gml` từ thư viện `NetworkX`. Tiếp theo, một mô hình N2V được khởi tạo với đồ thị đã tải. Mô hình này được thiết lập với các tham số cụ thể: nó sẽ tạo nhúng 64 chiều cho mỗi nút, sử dụng các bước đi 30 bước dọc theo, thực hiện 200 bước đi bắt đầu từ mỗi nút để thu thập ngữ cảnh, & chạy các thao tác này song song trên 4 worker để tăng tốc quá trình.

N2V model is then trained with additional parameters defined in `fit` method. This involves setting a context window size of 10 nodes around each target node to learn embeddings, considering all nodes at least once `min_count = 1`, & processing 4 words (nodes, in this context) each time during training.

– Mô hình N2V sau đó được huấn luyện với các tham số bổ sung được xác định trong phương thức `fit`. Phương thức này bao gồm việc thiết lập kích thước cửa sổ ngữ cảnh là 10 nút xung quanh mỗi nút mục tiêu để học các phép nhúng, xem xét tất cả các nút ít nhất một lần `min_count = 1`, & xử lý 4 từ (nút, trong ngữ cảnh này) mỗi lần trong quá trình huấn luyện.

Once trained, access node embeddings using `model`'s `mv` method (reflecting its NLP heritage, `wv` stands for word vectors). For our downstream tasks, we map each node to its embedding using a dictionary comprehension.

– Sau khi được đào tạo, hãy truy cập các nhúng nút bằng phương pháp `mv` của `model` (phản ánh di sản NLP của nó, `wv` là viết tắt của các vectơ từ). Đối với các tác vụ hạ nguồn, chúng tôi ánh xạ từng nút vào nhúng của nó bằng cách sử dụng một thuật toán hiểu từ điển.

```
import NetworkX as nx
from Node2Vec import Node2Vec
book_graph = nx.read_gml('polbooks.gml')
node2vec = Node2Vec(book_graph, dimensions = 64, walk_length = 30, num_walks = 200, workers = 4)
```

```
model = node2vec.fit(window = 10, min_count = 1, batch_words = 4)
embeddings = {str(node) : model.wv[str(node)] for node in gml_graph.nodes()}
```

1. Loads graph data from a GML file into a NetworkX graph object
2. Initializes N2V model with specified parameters for input graph
3. Trains N2V model
4. Extracts & stores node embeddings generated by N2V model in a dictionary.

∗ 2.1.2. Demystifying embeddings. Explore what these embeddings are & why they are valuable. An *embedding* is a dense numerical vector that represents identity of a node, edge, or graph in way that captures essential information about its structure & relationships. In our context, an embedding created by N2V captures a node's position & neighborhood within graph using topological information, i.e., it summarizes how node is connected to others, effectively capturing its role & importance in network. Later, when use GNNs to create embeddings, they will also encapsulate node's features, providing an even richer representation that includes both structure & attributes. Get deeper into theoretical aspects of embedding in Sect. 2.4.

– Giải mã các phép nhúng. Khám phá các phép nhúng này là gì & tại sao chúng có giá trị. Một phép nhúng là một vectơ số dày đặc biểu diễn danh tính của một nút, cạnh hoặc đồ thị theo cách nắm bắt thông tin thiết yếu về cấu trúc & các mối quan hệ của nó. Trong ngữ cảnh của chúng ta, một phép nhúng do N2V tạo ra nắm bắt vị trí & lân cận của một nút trong đồ thị bằng thông tin tôpô, tức là nó tóm tắt cách nút được kết nối với các nút khác, từ đó nắm bắt hiệu quả vai trò & tầm quan trọng của nút đó trong mạng. Sau này, khi sử dụng GNN để tạo các phép nhúng, chúng cũng sẽ đóng gói các đặc trưng của nút, cung cấp một biểu diễn phong phú hơn, bao gồm cả cấu trúc & các thuộc tính. Tìm hiểu sâu hơn về các khía cạnh lý thuyết của phép nhúng trong Phần 2.4.

These embeddings are powerful because they transform complex, high-dimensional graph data into a fixed-size vector format that can be easily in various analyses & ML tasks. E.g., they allow us to perform exploratory data analysis by revealing patterns, clusters, & relationships within graph. Beyond this, embeddings can be directly used as features in ML models, where each dimension of vector represents a distinct feature. This is particularly useful in applications where understanding structure & connections between data points, e.g. in social networks or recommendation systems, can significantly improve model performance.

– Các phép nhúng này rất mạnh mẽ vì chúng chuyển đổi dữ liệu đồ thị phức tạp, nhiều chiều thành định dạng vector có kích thước cố định, có thể dễ dàng thực hiện trong nhiều phân tích & tác vụ ML. Ví dụ: chúng cho phép chúng ta thực hiện phân tích dữ liệu thăm dò bằng cách khám phá các mẫu, cụm, & mối quan hệ trong đồ thị. Hơn thế nữa, các phép nhúng có thể được sử dụng trực tiếp làm đặc trưng trong các mô hình ML, trong đó mỗi chiều của vector đại diện cho một đặc trưng riêng biệt. Điều này đặc biệt hữu ích trong các ứng dụng mà việc hiểu cấu trúc & kết nối giữa các điểm dữ liệu, ví dụ như trong mạng xã hội hoặc hệ thống khuyến nghị, có thể cải thiện đáng kể hiệu suất mô hình.

To illustrate, consider node representing book *Losing Bin Laden* in our Political Books dataset. Using command `model.wv['Losing Bin Laden']`, we retrieve its dense vector embedding. This vector, shown in Fig. 2.3: Extracting embedding for nodes associated with political book *Losing Bin Laden*. Output is a dense vector represented as a Python list., captures various aspects of book's role within network of co-purchased books, providing a compact, informative representation that can be used for further analysis or as input to other models.

– Để minh họa, hãy xem xét nút biểu diễn cuốn sách `"Losing Bin Laden"` trong tập dữ liệu Sách Chính trị của chúng tôi. Sử dụng lệnh `model.wv['Losing Bin Laden']`, chúng tôi lấy được nhúng vector dày đặc của nó. Vector này, được hiển thị trong Hình 2.3: Trích xuất nhúng cho các nút liên kết với cuốn sách chính trị `"Losing Bin Laden"`. Đầu ra là một vector dày đặc được biểu diễn dưới dạng danh sách Python., nắm bắt các khía cạnh khác nhau về vai trò của cuốn sách trong mạng lưới các cuốn sách được mua chung, cung cấp một biểu diễn cô đọng, giàu thông tin, có thể được sử dụng để phân tích sâu hơn hoặc làm đầu vào cho các mô hình khác.

These embeddings can be used for exploratory data analysis to see patterns & relationships in a graph. However, their usage extends further. 1 common application is to use these vectors as features in a ML problem that uses tabular data. In that case, each element in our embedding array will become a distinct feature column in tabular data. This can add a rich representation of each node to complement other attributes in model training. In next sect, look at how to visualize these embeddings to gain deeper insights into patterns & relationships they represent.

– Các nhúng này có thể được sử dụng để phân tích dữ liệu thăm dò nhằm xem xét các mẫu & mối quan hệ trong biểu đồ. Tuy nhiên, ứng dụng của chúng còn mở rộng hơn nữa. Một ứng dụng phổ biến là sử dụng các vectơ này làm đặc trưng trong bài toán học máy sử dụng dữ liệu bảng. Trong trường hợp đó, mỗi phần tử trong mảng nhúng của chúng ta sẽ trở thành một cột đặc trưng riêng biệt trong dữ liệu bảng. Điều này có thể bổ sung thêm một biểu diễn phong phú cho mỗi nút để bổ sung cho các thuộc tính khác trong quá trình huấn luyện mô hình. Trong phần tiếp theo, hãy xem cách trực quan hóa các nhúng này để hiểu sâu hơn về các mẫu & mối quan hệ mà chúng biểu diễn.

∗ 2.1.3. Transforming & visualizing embeddings. Visualization methods e.g. Uniform Manifold Approximation & Projection (UMAP) are powerful tools for reducing high-dimensional datasets into lower-dimensional space [5]. UMAP is particularly effective for identifying inherent clusters & visualizing complex structures that are difficult to perceive in high-dimensional data. Compared to other methods, e.g. t-SNE, UMAP excels in preserving both local & global structures, making it ideal for revealing patterns & relationships across different scales in data.

– Biến đổi & trực quan hóa nhúng. Các phương pháp trực quan hóa, ví dụ như Xấp xỉ Đa tạp Đồng nhất & Chiếu (UMAP) là những công cụ mạnh mẽ để thu gọn các tập dữ liệu nhiều chiều thành không gian ít chiều hơn [5]. UMAP đặc biệt hiệu quả trong việc xác định các cụm cố hữu & trực quan hóa các cấu trúc phức tạp khó nhận biết trong dữ liệu nhiều chiều. So với các phương pháp khác, ví dụ như t-SNE, UMAP vượt trội trong việc bảo toàn cả cấu trúc cục bộ & toàn

cục, khiến nó trở nên lý tưởng để khám phá các mẫu & mối quan hệ trên các quy mô khác nhau trong dữ liệu.
While N2V generates embeddings by capturing network structure of our data, UMAP takes these high-dimensional embeddings & maps them onto a lower-dimensional space (typically 2D or 3D). This mapping aims to keep similar nodes close together while also preserving broader structural relationships, providing a more comprehensive visualization of graph's topology. After obtaining our N2V embeddings & converting them into a numerical array, we initialize UMAP model with 2 components to project our data onto a 2D plane. By carefully selecting parameters e.g. number of neighbors & minimum distance. UMAP can balance between revealing fine-grained local relationships & maintaining global distances between clusters.
– Trong khi N2V tạo ra các nhúng bằng cách nắm bắt cấu trúc mạng của dữ liệu, UMAP sử dụng các nhúng đa chiều này & ánh xạ chúng vào không gian đa chiều thấp hơn (thường là 2D hoặc 3D). Việc ánh xạ này nhằm mục đích giữ các nút tương tự gần nhau đồng thời bảo toàn các mối quan hệ cấu trúc rộng hơn, cung cấp hình ảnh trực quan toàn diện hơn về cấu trúc đồ thị. Sau khi thu thập các nhúng N2V & chuyển đổi chúng thành một mảng số, chúng tôi khởi tạo mô hình UMAP với 2 thành phần để chiếu dữ liệu lên mặt phẳng 2D. Bằng cách lựa chọn cẩn thận các tham số, ví dụ: số lượng lân cận & khoảng cách tối thiểu, UMAP có thể cân bằng giữa việc thể hiện các mối quan hệ cục bộ chi tiết & duy trì khoảng cách toàn cục giữa các cụm.
By using UMAP, gain a more accurate & interpretable visualization of our graph embeddings as shown in Listing 2.2. Visualizing embeddings using UMAP: 1. Transforms embeddings into a list of vectors for UMAP. 2. Initializes & fits UMAP. 3. Plots nodes with UMAP embeddings & color by their value. allowing us to explore & analyze patterns, clusters, & structures more effectively than with traditional methods e.g. t-SNE.
– Bằng cách sử dụng UMAP, bạn có thể có được hình ảnh trực quan chính xác hơn & dễ hiểu hơn về các nhúng đồ thị của mình như được hiển thị trong Liệt kê 2.2. Hình ảnh trực quan các nhúng bằng UMAP: 1. Chuyển đổi các nhúng thành danh sách các vectơ cho UMAP. 2. Khởi tạo & phù hợp với UMAP. 3. Vẽ các nút bằng các nhúng UMAP & tô màu theo giá trị của chúng. cho phép chúng ta khám phá & phân tích các mẫu, cụm, & cấu trúc hiệu quả hơn so với các phương pháp truyền thống, ví dụ: t-SNE.
Resultant Fig. 2.4: Embeddings of Political Books dataset graph generated by N2V & visualized using UMAP. Shape & shading variations distinguish 3 political classes. encapsulates political book graph's embeddings as distilled by N2V & subsequently visualized through UMAP. Nodes appear in different shades according to their political alignment. Visualization unfolds a discernible structure, with potential clusters that correspond to various political learnings.
– Kết quả Hình 2.4: Đồ thị nhúng của tập dữ liệu Sách Chính trị được tạo bởi N2V & trực quan hóa bằng UMAP. Hình dạng & các biến thể tô bóng phân biệt 3 lớp chính trị. đóng gói các nhúng của đồ thị sách chính trị được tinh lọc bởi N2V & sau đó được trực quan hóa bằng UMAP. Các nút xuất hiện với các sắc thái khác nhau tùy theo sự liên kết chính trị của chúng. Trực quan hóa mở ra một cấu trúc rõ ràng, với các cụm tiềm năng tương ứng với các bài học chính trị khác nhau.
Might wonder why don't just reduce dimensions of N2V embeddings from 64 to 2 & visualize them directly, by passing UMAP altogether? In Listing 2.3: Visualizing 2D N2V embeddings without t-SNE, show this approach, applying a 2D N2V transformation directly to our `book_graph` object.
– Có thể bạn đang thắc mắc tại sao không giảm kích thước của các nhúng N2V từ 64 xuống còn 2 & trực quan hóa chúng trực tiếp bằng cách truyền hoàn toàn UMAP? Trong Liệt kê 2.3: Trực quan hóa các nhúng N2V 2D mà không cần t-SNE, hãy trình bày cách tiếp cận này, áp dụng phép biến đổi N2V 2D trực tiếp vào đối tượng `book_graph` của chúng ta.
`dimensions` parameter is set to 2, aiming for a direct 2D representation suitable for immediate visualization without further dimensionality reduction. Other parameters are kept the same.
– Tham số `dimensions` được đặt thành 2, hướng đến biểu diễn 2D trực tiếp, phù hợp để trực quan hóa ngay lập tức mà không cần giảm kích thước thêm. Các tham số khác được giữ nguyên.
Once model is fitted with specified window & word batch settings, extract 2D embeddings & store them in a dictionary keyed by string representation of each node. This enables a direct mapping from node to its embedding vector.
– Sau khi mô hình được điều chỉnh với các thiết lập hàng loạt cửa sổ & từ được chỉ định, hãy trích xuất các nhúng 2D & lưu trữ chúng trong một từ điển được khóa bằng chuỗi biểu diễn của mỗi nút. Điều này cho phép ánh xạ trực tiếp từ nút đến vectơ nhúng của nó.
Extracted 2D points are compiled into a NumPy array & plotted. Use standard Matplotlib library to create a scatterplot of these points using prepared color scheme to represent political leaning of each node visually.
– Các điểm 2D được trích xuất sẽ được biên dịch thành một mảng NumPy & vẽ đồ thị. Sử dụng thư viện Matplotlib chuẩn để tạo biểu đồ phân tán các điểm này bằng cách sử dụng bảng màu đã chuẩn bị sẵn để thể hiện trực quan khuynh hướng chính trị của từng nút.
Outcome shows how books are separated by political leanings, similar to UMAP result, but where books are more bunched together Fig. 2.5: Embeddings of Political Books dataset graph generated & visualized by N2V for 2D. Shape & shading variations distinguish 3 political classes. Here, see a similar clustering by political leaning as earlier in Fig. 2.5 but more bunched together. 2 embeddings are then shown in Fig. 2.6: Comparison of embeddings generated by N2V & t-SNE & a direct visualization of 2D Node2Vec.
– Kết quả cho thấy sách được phân loại theo khuynh hướng chính trị, tương tự như kết quả UMAP, nhưng sách được gom lại với nhau nhiều hơn Hình 2.5: Đồ thị nhúng của tập dữ liệu Sách Chính trị được tạo & trực quan hóa bằng N2V cho 2D. Các biến thể hình & tô bóng phân biệt 3 lớp chính trị. Ở đây, hãy xem một phân cụm tương tự theo khuynh hướng chính trị như trước đó trong Hình 2.5 nhưng được gom lại với nhau nhiều hơn. 2 nhúng sau đó được hiển thị trong Hình 2.6: So sánh các nhúng được tạo bởi N2V & t-SNE & trực quan hóa trực tiếp của Node2Vec 2D.
Clear: both methods know to separate books into groups based on political leanings. N2V is less expressive in how it

separates books, bunching them together across 2D. Meanwhile, UMAP is better for spreading out books in 2D. Relevant benefit or information contained within these dimensions depends on task at hand.
– Rõ ràng: cả hai phương pháp đều biết cách phân loại sách thành các nhóm dựa trên khuynh hướng chính trị. N2V kém biểu cảm hơn trong cách phân loại sách, gom chúng lại với nhau trên không gian 2 chiều. Trong khi đó, UMAP tốt hơn trong việc phân bổ sách trên không gian 2 chiều. Lợi ích hoặc thông tin liên quan chứa trong các chiều này phụ thuộc vào nhiệm vụ được giao.

∗ 2.1.4. Beyond visualization: Applications & considerations of N2V embeddings. While visualizing N2V embeddings offers intuitive insights into dataset's structure, their usage extends far beyond graphical representation. N2V is an embedding method designed specifically for graphs; it captures both local & global structural properties of nodes by simulating random walks through graph. This process allows N2V to create dense, numerical vectors that summarize position & context of each node within overall network.
– Vượt ra ngoài trực quan hóa: Ứng dụng & cân nhắc về nhúng N2V. Mặc dù trực quan hóa nhúng N2V mang lại cái nhìn sâu sắc trực quan về cấu trúc của tập dữ liệu, nhưng ứng dụng của chúng còn vượt xa việc biểu diễn đồ họa. N2V là một phương pháp nhúng được thiết kế riêng cho đồ thị; nó nắm bắt cả các thuộc tính cấu trúc cục bộ & toàn cục của các nút bằng cách mô phỏng các bước ngẫu nhiên qua đồ thị. Quá trình này cho phép N2V tạo ra các vectơ số dày đặc, tóm tắt vị trí & ngữ cảnh của mỗi nút trong toàn bộ mạng.
These embeddings can then serve as feature-rich inputs for a variety of ML tasks, e.g. classification, recommendation, or clustering. E.g. in our Political Books dataset, embeddings could help predict a book's political learning based on its co-purchase patterns or could recommend books to users with similar political interests. They might even be used to forecast future sales based on content of a book.
– Những nhúng này sau đó có thể đóng vai trò là dữ liệu đầu vào giàu tính năng cho nhiều tác vụ ML, ví dụ như phân loại, đề xuất hoặc phân cụm. Ví dụ: trong tập dữ liệu Sách Chính trị của chúng tôi, các nhúng có thể giúp dự đoán khả năng học hỏi chính trị của một cuốn sách dựa trên các mô hình mua chung hoặc có thể đề xuất sách cho người dùng có cùng sở thích chính trị. Chúng thậm chí có thể được sử dụng để dự báo doanh số bán hàng trong tương lai dựa trên nội dung của một cuốn sách.
However, important to understand nature of N2V's learning approach, which is *transductive*. Transductive learning is designed to work only with specific dataset it was trained on & cannot generalize to new, unseen nodes without retraining model. This characteristic makes N2V highly effective for static datasets where all nodes & edges are known up front but less suitable for dynamic settings where new data points or connections frequently appear. Essentially, N2V focuses on extracting detailed patterns & relationships from existing graph rather than developing a model that can easily adapt to new data.
– Tuy nhiên, điều quan trọng là phải hiểu bản chất của phương pháp học tập của N2V, đó là *transductive*. Học tập transductive được thiết kế để chỉ hoạt động với tập dữ liệu cụ thể mà nó được huấn luyện & không thể khái quát hóa sang các nút mới, chưa được biết đến mà không cần huấn luyện lại mô hình. Đặc điểm này khiến N2V rất hiệu quả đối với các tập dữ liệu tĩnh trong đó tất cả các nút & cạnh đều được biết trước nhưng ít phù hợp hơn với các thiết lập động, nơi các điểm dữ liệu hoặc kết nối mới thường xuyên xuất hiện. Về cơ bản, N2V tập trung vào việc trích xuất các mẫu & mối quan hệ chi tiết từ đồ thị hiện có thay vì phát triển một mô hình có thể dễ dàng thích ứng với dữ liệu mới.
While this transductive nature has its limitations, it also offer significant advantages. Because n2V uses full structure of graph during training, it can capture intricate relationships & dependencies that might be missed by more generalized methods. This makes N2V particularly powerful for tasks where complete, fixed structure of data is known & stable. However, to apply N2V effectively, it is crucial to ensure that graph data is represented in a way that captures all relevant features. In some cases, additional edges or nodes may need to be added to graph to fully represent underlying relationships.
– Mặc dù bản chất chuyển đổi này có những hạn chế, nhưng nó cũng mang lại những lợi thế đáng kể. Vì n2V sử dụng toàn bộ cấu trúc đồ thị trong quá trình huấn luyện, nó có thể nắm bắt được các mối quan hệ phức tạp & các mối phụ thuộc mà các phương pháp tổng quát hơn có thể bỏ sót. Điều này làm cho N2V đặc biệt mạnh mẽ đối với các tác vụ mà cấu trúc dữ liệu hoàn chỉnh, cố định đã biết & ổn định. Tuy nhiên, để áp dụng N2V hiệu quả, điều quan trọng là phải đảm bảo dữ liệu đồ thị được biểu diễn theo cách nắm bắt được tất cả các đặc điểm liên quan. Trong một số trường hợp, có thể cần thêm các cạnh hoặc nút bổ sung vào đồ thị để biểu diễn đầy đủ các mối quan hệ cơ bản.
For those interested in a deeper understanding of transductive models & how N2V's approach compares to other methods, further details are provided in Sect. 2.4.2, which explore tradeoffs between transductive & inductive learning in greater depth [6, 7], helping you understand when each approach is most appropriate.
– Đối với những người quan tâm đến việc hiểu sâu hơn về các mô hình chuyển đổi & cách tiếp cận của N2V so sánh với các phương pháp khác, các chi tiết bổ sung được cung cấp trong Phần 2.4.2, khám phá sự đánh đổi giữa học chuyển đổi & quy nạp sâu hơn [6, 7], giúp bạn hiểu khi nào thì mỗi cách tiếp cận là phù hợp nhất.
While N2V is effective for generating embeddings that capture structure of a fixed graph, real-world data often demands a more flexible & generalizable approach. This need brings us to our 1st GNN architecture for creating node embeddings. Unlike N2V, which is a transductive method limited to specific nodes & edges in training data, GNNs can learn in an *inductive* manner, i.e., GNNs are capable of generalizing to new, unseen nodes or edges without requiring retraining on entire graph.
– Mặc dù N2V hiệu quả trong việc tạo ra các phép nhúng nắm bắt cấu trúc của một đồ thị cố định, dữ liệu thực tế thường đòi hỏi một phương pháp linh hoạt hơn & có thể khái quát hóa. Nhu cầu này đưa chúng ta đến kiến trúc GNN đầu tiên để tạo ra các phép nhúng nút. Không giống như N2V, vốn là phương pháp chuyển đổi giới hạn ở các nút & cạnh cụ thể

trong dữ liệu huấn luyện, GNN có thể học theo cách *quy nạp*, tức là GNN có khả năng khái quát hóa sang các nút hoặc cạnh mới, chưa được biết đến mà không cần phải huấn luyện lại toàn bộ đồ thị.

GNNs achieve this by not only understanding network's complex structure but also by incorporating node features & relationships into learning process. This approach allows GNNs to adapt dynamically to changes in graph, making them well-suited for applications where data is continually evolving. Shift from N2V to GNNs represents a key transition from focusing on deep analysis within a static dataset to a broader applicability across diverse, evolving networks. This adaptability sets stage for a wider range of graph-based ML applications that require flexibility & scalability. In next sect, explore how GNNs go beyond capabilities of N2V & other transductive methods, allowing for ore versatile & powerful models that can handle dynamic nature of real-world data.

– GNN đạt được điều này không chỉ bằng cách hiểu cấu trúc phức tạp của mạng mà còn bằng cách kết hợp các đặc điểm nút & mối quan hệ vào quá trình học. Cách tiếp cận này cho phép GNN thích ứng linh hoạt với những thay đổi trong đồ thị, khiến chúng phù hợp với các ứng dụng mà dữ liệu liên tục phát triển. Sự chuyển đổi từ N2V sang GNN thể hiện một bước chuyển đổi quan trọng từ việc tập trung vào phân tích sâu trong một tập dữ liệu tĩnh sang khả năng ứng dụng rộng rãi hơn trên các mạng lưới đa dạng và đang phát triển. Khả năng thích ứng này đặt nền tảng cho một loạt các ứng dụng học máy dựa trên đồ thị đòi hỏi tính linh hoạt & khả năng mở rộng. Trong phần tiếp theo, hãy khám phá cách GNN vượt ra ngoài khả năng của N2V & các phương pháp chuyển đổi khác, cho phép tạo ra các mô hình linh hoạt & mạnh mẽ hơn có thể xử lý bản chất động của dữ liệu thực tế.

- 2.2. Creating embeddings with a GNN. While N2V provides a powerful method for generating embeddings by capturing local & global structure of a graph, it is fundamentally a transductive approach, i.e., it cannot easily generalize to unseen nodes or edges without retraining. Although there have been extensions to N2V that enable it to work in inductive settings, GNNs are inherently designed for inductive learning. I.e., they can learn general patterns from graph data that allow them to make predictions or to generate embeddings for new nodes or edges without need to retrain entire model. This gives GNNs a significant edge in scenarios where flexibility & adaptability are crucial.

  – Tạo nhúng với GNN. Mặc dù N2V cung cấp một phương pháp mạnh mẽ để tạo nhúng bằng cách nắm bắt cấu trúc cục bộ & toàn cục của đồ thị, nhưng về cơ bản, nó là một phương pháp chuyển đổi, tức là nó không thể dễ dàng khái quát hóa sang các nút hoặc cạnh chưa biết mà không cần đào tạo lại. Mặc dù đã có các phần mở rộng cho N2V cho phép nó hoạt động trong các thiết lập quy nạp, GNN vốn được thiết kế cho việc học quy nạp. Tức là, chúng có thể học các mẫu chung từ dữ liệu đồ thị, cho phép chúng đưa ra dự đoán hoặc tạo nhúng cho các nút hoặc cạnh mới mà không cần đào tạo lại toàn bộ mô hình. Điều này mang lại cho GNN một lợi thế đáng kể trong các tình huống mà tính linh hoạt & khả năng thích ứng là rất quan trọng.

  GNNs not only incorporate structural information of graph, like N2V, but they also use node features to create richer representations. This dual capacity allows GNNs to learn both complex relationships within graph & specific characteristics of individual nodes, enabling them to excel in tasks where both types of information are important.

  – GNN không chỉ kết hợp thông tin cấu trúc của đồ thị, như N2V, mà còn sử dụng các đặc điểm của nút để tạo ra các biểu diễn phong phú hơn. Khả năng kép này cho phép GNN học cả các mối quan hệ phức tạp trong đồ thị & các đặc điểm cụ thể của từng nút, cho phép chúng vượt trội trong các tác vụ mà cả hai loại thông tin đều quan trọng.

  That said, while GNNs have demonstrated impressive performance across many applications, they do not universally outperform methods e.g. N2V in all cases. E.g., N2V & other random walk-based methods can sometimes perform better in scenarios where labeled data is scarce or noisy, thanks to their ability to work with just graph structure without needing additional node features.

  – Tuy nhiên, mặc dù GNN đã chứng minh hiệu suất ấn tượng trong nhiều ứng dụng, chúng không phải lúc nào cũng vượt trội hơn các phương pháp như N2V trong mọi trường hợp. Ví dụ, N2V & các phương pháp dựa trên bước đi ngẫu nhiên khác đôi khi có thể hoạt động tốt hơn trong các tình huống dữ liệu được gắn nhãn khan hiếm hoặc nhiễu, nhờ khả năng hoạt động chỉ với cấu trúc đồ thị mà không cần thêm các đặc trưng nút.

  * 2.2.1. Constructing embeddings. Unlike N2V, GNNs learn graph representations & perform tasks e.g. node classification or link prediction simultaneously during training. Information from entire is processed through successive GNN layers, each refining node embeddings without requiring a separate step for their creation.

    – Xây dựng nhúng. Không giống như N2V, GNN học các biểu diễn đồ thị & thực hiện đồng thời các tác vụ như phân loại nút hoặc dự đoán liên kết trong quá trình huấn luyện. Thông tin từ toàn bộ được xử lý qua các lớp GNN kế tiếp, mỗi lớp tinh chỉnh các nhúng nút mà không cần một bước riêng biệt để tạo chúng.

    To demonstrate how a GNN extracts features from graph data, perform a straightforward pass-through using an untrained model to generate preliminary embeddings. Even without optimization typically involved in training, this approach will show how GNNs use message passing to update embeddings, capturing both graph's structure & its node features. When optimization is added, these embeddings become tailored to specific tasks e.g. node classification or link prediction.

    – Để minh họa cách GNN trích xuất các đặc điểm từ dữ liệu đồ thị, hãy thực hiện một phép truyền trực tiếp đơn giản bằng một mô hình chưa được huấn luyện để tạo ra các nhúng sơ bộ. Ngay cả khi không có quá trình tối ưu hóa thường được sử dụng trong huấn luyện, phương pháp này vẫn sẽ cho thấy cách GNN sử dụng việc truyền thông điệp để cập nhật các nhúng, nắm bắt cả cấu trúc của đồ thị và các đặc điểm nút của nó. Khi được tối ưu hóa, các nhúng này sẽ được điều chỉnh cho phù hợp với các tác vụ cụ thể, ví dụ như phân loại nút hoặc dự đoán liên kết.

    · Defining our GNN architecture. Initiate our process by defining a siple GCN architecture, as shown in Listing 2.4: `SimpleGNN` class. Our `SimpleGNN` class inherits from `torch.nn.Module` & is composed of 2 `GCN-Conv` layers, which

are building blocks of our GNN. This architecture is shown in Fig. 2.7: Architecture diagram of `SimpleGNN` model, consisting of 1st layer, a message passing layer `self.conv1`, an activation `torch.relu`, a dropout layer `torch.dropout`, & a 2nd message passing layer.
– Định nghĩa kiến trúc GNN của chúng ta. Bắt đầu quy trình bằng cách định nghĩa một kiến trúc GCN đơn, như được hiển thị trong Liệt kê 2.4: `SimpleGNN` class. Lớp `SimpleGNN` của chúng ta kế thừa từ `torch.nn.Module` & bao gồm 2 lớp `GCN-Conv`, là các khối xây dựng nên GNN của chúng ta. Kiến trúc này được hiển thị trong Hình 2.7: Sơ đồ kiến trúc của mô hình `SimpleGNN`, bao gồm lớp thứ nhất, lớp truyền tin nhắn `self.conv1`, lớp kích hoạt `torch.relu`, lớp dropout `torch.dropout`, & lớp truyền tin nhắn thứ hai.
Talk about architecture aspects specific to GNNs. Activation & dropout are common in many DL scenarios. GNN layers, however, are different from conventional DL layers in a fundamental way. Core principle that allows GNNs to learn from graph data is message passing. For each GNN layer, in addition to updating layer's weights, a "message" is gathered from every node or edge neighborhood & used to update an embedding. Essentially, each node sends messages to its neighbors & simultaneously receives messages from them. For every node, its new embedding is computed by combining its own features with aggregated messages from its neighbors, through a combination of nonlinear transformations.
– Nói về các khía cạnh kiến trúc đặc thù của GNN. Activation & dropout là phổ biến trong nhiều kịch bản DL. Tuy nhiên, các lớp GNN khác biệt cơ bản so với các lớp DL thông thường. Nguyên lý cốt lõi cho phép GNN học từ dữ liệu đồ thị là truyền thông điệp. Đối với mỗi lớp GNN, ngoài việc cập nhật trọng số của lớp, một "thông điệp" được thu thập từ mọi nút hoặc lân cận cạnh & được sử dụng để cập nhật nhúng. Về cơ bản, mỗi nút gửi thông điệp đến các nút lân cận & đồng thời nhận thông điệp từ chúng. Đối với mỗi nút, nhúng mới của nó được tính toán bằng cách kết hợp các đặc trưng của chính nó với các thông điệp tổng hợp từ các nút lân cận, thông qua sự kết hợp của các phép biến đổi phi tuyến tính.
In this example, we are going to be using a graph convolutional network (GCN) to act as our message-passing GNN layers. Describe GCNs in much more detail i Chap. 3. For now, just need to know that GCNs act as message-passing layers that are critical in constructing embeddings.
– Trong ví dụ này, chúng ta sẽ sử dụng mạng tích chập đồ thị (GCN) để hoạt động như các lớp GNN truyền thông điệp. Mô tả chi tiết hơn về GCN trong Chương 3. Hiện tại, chúng ta chỉ cần biết rằng GCN hoạt động như các lớp truyền thông điệp, đóng vai trò quan trọng trong việc xây dựng các hàm nhúng.

· Data preparation. Next, prepare our data. Start with same graph from previous section, `books_gml`, in its `NetworkX` form. Have to convert this `NetworkX` object into a tensor form that is suitable to use with PyTorch operations. Because PyTorch Geometric (PyG) has many functions that convert graph objects, we can do this quite simply with `data = from_NetworkX(gml_graph)`. Method `from_NetworkX` specifically translates edge lists & node.edge attributes into PyTorch tensors.
– Chuẩn bị dữ liệu. Tiếp theo, hãy chuẩn bị dữ liệu. Bắt đầu với cùng một đồ thị từ phần trước, `books_gml`, ở dạng `NetworkX`. Phải chuyển đổi đối tượng `NetworkX` này sang dạng tensor phù hợp để sử dụng với các phép toán PyTorch. Vì PyTorch Geometric (PyG) có nhiều hàm chuyển đổi đối tượng đồ thị, chúng ta có thể thực hiện việc này khá đơn giản với `data = from_NetworkX(gml_graph)`. Phương thức `from_NetworkX` đặc biệt chuyển đổi các thuộc tính danh sách cạnh & node.edge thành tensor PyTorch.
For GNNs, generating node embeddings requires initializing node features. In our case, we do not have any predefined node features. When no node features are available or they are not informative, it is common practice to initialize node features randomly. A more effective approach: use *Xavier initialization*, which sets initial node features with values drawn from a distribution that keeps variety of activations consistent across layers. This technique ensures: model starts with a balanced representation, preventing problems e.g. vanishing or exploding gradients.
– Đối với GNN, việc tạo nhúng nút đòi hỏi phải khởi tạo các đặc trưng nút. Trong trường hợp của chúng tôi, chúng tôi không có bất kỳ đặc trưng nút nào được xác định trước. Khi không có đặc trưng nút nào khả dụng hoặc chúng không cung cấp thông tin, việc khởi tạo các đặc trưng nút một cách ngẫu nhiên là một phương pháp phổ biến. Một cách tiếp cận hiệu quả hơn: sử dụng *Xavier initialization*, phương pháp này đặt các đặc trưng nút ban đầu với các giá trị được lấy từ một phân phối duy trì tính nhất quán của các phép kích hoạt trên các lớp. Kỹ thuật này đảm bảo: mô hình bắt đầu với một biểu diễn cân bằng, ngăn ngừa các vấn đề như gradient biến mất hoặc bùng nổ.
By initializing `data.x` with Xavier initialization, provide GNN with a starting point that allows it to learn meaningful node embeddings from noninformative features. During training, network adjusts these initial values to minimize loss function. When loss function is aligned with a specific target, e.g. node prediction, embeddings learned from initial random features will become tailored to task at hand, resulting in more effective representations. Randomize node features using following:
– Khởi tạo, cung cấp cho GNN một điểm khởi đầu cho phép nó học các phép nhúng nút có ý nghĩa từ các đặc trưng không mang tính thông tin. Trong quá trình huấn luyện, mạng sẽ điều chỉnh các giá trị ban đầu này để giảm thiểu hàm mất mát. Khi hàm mất mát được căn chỉnh với một mục tiêu cụ thể, ví dụ: dự đoán nút, các phép nhúng học được từ các đặc trưng ngẫu nhiên ban đầu sẽ được điều chỉnh cho phù hợp với tác vụ hiện tại, mang lại hiệu quả biểu diễn cao hơn. Ngẫu nhiên hóa các đặc trưng nút bằng cách sử dụng các bước sau:

```
1 data.x = torch.randn((data.num_nodes, 64), dtype = torch.float)
2 'nn.init.xavier_uniform_(data.x) '
```

We could have also used embeddings from N2V exercise to use as node features. Recall `node_embeddings` object from Sect. 2.1.3:
– Chúng ta cũng có thể sử dụng các nhúng từ bài tập N2V để làm đặc trưng nút. Hãy nhớ lại đối tượng `node_embeddings`

từ Mục 2.1.3:

```
node_embeddings = [embeddings[str(node)] for node in gml_graph.nodes()]
```

From this, can convert node embedding to a PyTorch tensor object & assign it to node feature object, `data.x`:
– Từ đó, có thể chuyển đổi nhúng nút thành đối tượng tenxơ PyTorch & gán nó cho đối tượng tính năng nút, `data.x`:

```
node_features = torch.tensor(node_embedding, dtype = torch.float)
data.x = node_features
```

- Passing graph through GNN. With structure of our GNN model defined & our graph data formatted for PyG, proceed to embedding generation step. Initialize our model, `SimpleGNN`, specifying number of features for each node & size of hidden channels within network.
– Truyền đồ thị qua GNN. Với cấu trúc mô hình GNN đã được xác định & dữ liệu đồ thị được định dạng cho PyG, hãy tiến hành bước tạo nhúng. Khởi tạo mô hình `SimpleGNN`, chỉ định số lượng đặc trưng cho mỗi nút & kích thước của các kênh ẩn trong mạng.

```
model = SimpleGNN(num_features = data.x.shape[1], hidden_channels = 64)
```

Here, specify 64 hidden channels because we want to compare resulting embeddings to the ones we produced using `node2vec` method, which had 64 dimensions. Because 2nd GNN layer is last layer, output will be a 64-element vector.
– Ở đây, ta chỉ định 64 kênh ẩn vì chúng ta muốn so sánh các nhúng kết quả với các nhúng chúng ta tạo ra bằng phương pháp `node2vec`, có 64 chiều. Vì lớp GNN thứ 2 là lớp cuối cùng, đầu ra sẽ là một vectơ 64 phần tử.
Once initialized, we switch model to evaluation mode using `model.eval()`. This mode is used during inference or validation phases when we want to make predictions or assess model performance without modifying model's parameters. Specifically, `model.eval()` turns off certain behaviors specific to training, e.g. *dropout*, which randomly deactivates some neurons to prevent overfitting, & *batch normalization*, which normalizes inputs across a mini-batch. By disabling these features, model provides consistent & deterministic outputs, ensuring: evaluation accurately reflects its true performance on unseen data.
– Sau khi khởi tạo, chúng tôi chuyển mô hình sang chế độ đánh giá bằng `model.eval()`. Chế độ này được sử dụng trong các giai đoạn suy luận hoặc xác thực khi chúng tôi muốn đưa ra dự đoán hoặc đánh giá hiệu suất mô hình mà không cần sửa đổi các tham số của mô hình. Cụ thể, `model.eval()` sẽ tắt một số hành vi cụ thể trong quá trình huấn luyện, ví dụ: `dropout`, vô hiệu hóa ngẫu nhiên một số nơ-ron để ngăn ngừa quá khớp, & `batch normalization`, chuẩn hóa đầu vào trên một mini-batch. Bằng cách vô hiệu hóa các tính năng này, mô hình cung cấp đầu ra nhất quán & xác định, đảm bảo: đánh giá phản ánh chính xác hiệu suất thực tế của nó trên dữ liệu chưa được biết đến.
Important to disable gradient computations because they are not necessary for forward pass & embedding generation. So, we employ `torch.no_grad()`, which ensures: computational graph that records operations for backpropagation is not constructed, preventing us from accidentally changing performance.
– Điều quan trọng là phải tắt tính toán gradient vì chúng không cần thiết cho việc tạo nhúng & truyền tiếp. Vì vậy, chúng tôi sử dụng `torch.no_grad()`, đảm bảo: đồ thị tính toán ghi lại các thao tác cho lan truyền ngược không được xây dựng, ngăn chúng tôi vô tình thay đổi hiệu suất.
Next, pass our node-feature matrix `data.x` & edge index `data.edge_index` through model. Result is `gnn_embeddings`, a tensor where each row corresponds to embedding of a node in our graph – a numerical representation learned by our GNN, ready for downstream tasks e.g. visualization or classification:
– Tiếp theo, truyền ma trận đặc trưng nút `data.x` & chỉ số cạnh `data.edge_index` qua mô hình. Kết quả là `gnn_embeddings`, một tenxơ trong đó mỗi hàng tương ứng với việc nhúng một nút vào đồ thị của chúng ta – một biểu diễn số được học bởi GNN, sẵn sàng cho các tác vụ tiếp theo, ví dụ như trực quan hóa hoặc phân loại:

```
model.eval()
with torch.no_grad():
    gnn_embeddings = model(data.x, data.edge_index)
```

After producing these embeddings, use UMAP to visualize them. Since we have been working with PyTorch tensor data types running on a GPU, need to convert our embeddings to a NumPy array data type to use analysis methods outside of PyTorch, which are done on a CPU:
– Sau khi tạo các nhúng này, hãy sử dụng UMAP để trực quan hóa chúng. Vì chúng ta đang làm việc với các kiểu dữ liệu tensor PyTorch chạy trên GPU, cần chuyển đổi các nhúng sang kiểu dữ liệu mảng NumPy để sử dụng các phương pháp phân tích bên ngoài PyTorch, vốn được thực hiện trên CPU:

```
gnn_embeddings_np = gnn_embeddings.detach().cpu().numpy()
```

With this convention, we can produce UMAP calculations & visualization following process we used in N2V case. Resulting scatterplot (Fig. 2.8: Visualization of embeddings generated from passing a graph through a GNN.) is a 1st glimpse at clusters within our graph. Add different shadings based on each node's label (left-, right-, or neural-leaning) to see that similar leaning books are fairly well grouped, given that these embeddings were constructed from topology alone.
– Với quy ước này, chúng ta có thể tạo ra các phép tính UMAP & trực quan hóa theo quy trình đã sử dụng trong trường hợp N2V. Biểu đồ phân tán kết quả (Hình 2.8: Trực quan hóa các nhúng được tạo ra khi truyền đồ thị qua mạng GNN.)

là cái nhìn đầu tiên về các cụm trong đồ thị của chúng ta. Hãy thêm các hiệu ứng tô bóng khác nhau dựa trên nhãn của mỗi nút (nghiêng trái, phải hoặc nghiêng nơ-ron) để thấy rằng các sách nghiêng tương tự được nhóm khá tốt, vì các nhúng này chỉ được xây dựng từ tôpô.

Next, discuss both how GNN embeddings are used & how they differ from those produced with N2V.

– Tiếp theo, thảo luận về cách sử dụng nhúng GNN & cách chúng khác với nhúng được tạo bằng N2V như thế nào.

* 2.2.2. GNN vs. N2V embeddings. Through this book, predominantly use GNNs to generate embeddings because this embedding process is intrinsic to a GNN's architecture. While embeddings play a pivotal role in methodologies & applications we explore in rest of book, their presence is often subtle & not always highlighted. This approach allows us to focus on broader concepts & applications of GNN-based ML without getting slowed down by technicalities. Nonetheless, important to acknowledge: underlying power & adaptability of embeddings are central to advanced techniques & insights we get into throughout text.

– GNN so với nhúng N2V. Trong cuốn sách này, chúng tôi chủ yếu sử dụng GNN để tạo nhúng vì quá trình nhúng này là một phần không thể thiếu trong kiến trúc của GNN. Mặc dù nhúng đóng vai trò then chốt trong các phương pháp luận & ứng dụng mà chúng tôi sẽ đề cập trong phần còn lại của cuốn sách, nhưng sự hiện diện của chúng thường rất tinh tế & không phải lúc nào cũng được nhấn mạnh. Cách tiếp cận này cho phép chúng tôi tập trung vào các khái niệm rộng hơn & ứng dụng của ML dựa trên GNN mà không bị chậm lại bởi các vấn đề kỹ thuật. Tuy nhiên, điều quan trọng cần lưu ý là: sức mạnh tiềm ẩn & khả năng thích ứng của nhúng là trọng tâm của các kỹ thuật tiên tiến & những hiểu biết sâu sắc mà chúng tôi sẽ tìm hiểu trong suốt cuốn sách.

GNN-produced node embeddings are particularly powerful because they enable us to tackle a broad range of graph-related tasks by using their inductive nature. Inductive learning allows these embeddings to generalize to new, unseen nodes or even entirely new graphs without needing to retrain model. In contrast, N2V embeddings are limited to specific graphs they were trained on & can't easily adapt to new data. Reiterate key ways in which GNN embeddings differ from other embedding methods, e.g. N2V [1, 3].

– Nhúng nút do GNN tạo ra đặc biệt mạnh mẽ vì chúng cho phép chúng ta giải quyết một loạt các tác vụ liên quan đến đồ thị bằng cách sử dụng tính chất quy nạp của chúng. Học quy nạp cho phép các nhúng này tổng quát hóa sang các nút mới, chưa từng thấy hoặc thậm chí là các đồ thị hoàn toàn mới mà không cần phải đào tạo lại mô hình. Ngược lại, nhúng N2V bị giới hạn trong các đồ thị cụ thể mà chúng được đào tạo & không thể dễ dàng thích ứng với dữ liệu mới. Nhắc lại những điểm chính mà nhúng GNN khác với các phương pháp nhúng khác, ví dụ: N2V [1, 3].

· Adaptability of new graphs. 1 of critical features of GNN embeddings is their adaptability. Because GNNs learn a function that maps node features to embeddings, this function can be applied to nodes in new graphs without needing to be retrained, provided nodes have similar feature spaces. This inductive capability is particularly valuable in dynamic environments where graph may evolve over time or in applications where model needs to be applied to different but structurally similar graphs. N2V, on other hand, needs to be reapplied for each new graph or set of nodes.

– Khả năng thích ứng của đồ thị mới. Một trong những đặc điểm quan trọng của nhúng GNN là khả năng thích ứng. Vì GNN học một hàm ánh xạ các đặc trưng của nút với các nhúng, hàm này có thể được áp dụng cho các nút trong đồ thị mới mà không cần phải đào tạo lại, miễn là các nút có không gian đặc trưng tương tự. Khả năng quy nạp này đặc biệt hữu ích trong các môi trường động, nơi đồ thị có thể phát triển theo thời gian hoặc trong các ứng dụng cần áp dụng mô hình cho các đồ thị khác nhau nhưng có cấu trúc tương tự. Mặt khác, N2V cần được áp dụng lại cho mỗi đồ thị hoặc tập hợp nút mới.

· Enhanced feature integration. GNNs inherently consider node features during embedding process, allowing for a complex & nuanced representation of each node. This integration of node features, alongside structural information, offers a more comprehensive view compared to N2V & other methods that focus on a graph's topology. This capability makes GNN embeddings particularly suited for tasks where node features contain significant additional information.

– Tích hợp tính năng nâng cao. GNN vốn đã xem xét các tính năng nút trong quá trình nhúng, cho phép biểu diễn phức tạp & sắc thái của từng nút. Việc tích hợp các tính năng nút này, cùng với thông tin cấu trúc, mang lại cái nhìn toàn diện hơn so với các phương pháp N2V & khác tập trung vào cấu trúc của đồ thị. Khả năng này làm cho nhúng GNN đặc biệt phù hợp cho các tác vụ mà các tính năng nút chứa thông tin bổ sung đáng kể.

· Task-specific optimization. GNNs embeddings are trained alongside specific tasks, e.g. node classification, link prediction, or even graph classification. Through end-to-end training, GNN model learns to optimize embeddings for task at hand, leading to potentially higher performance & efficiency compared to using pre-generated embeddings e.g. those from N2V.

– Tối ưu hóa theo tác vụ. Các nhúng GNN được huấn luyện cùng với các tác vụ cụ thể, ví dụ: phân loại nút, dự đoán liên kết, hoặc thậm chí phân loại đồ thị. Thông qua quá trình huấn luyện đầu cuối, mô hình GNN học cách tối ưu hóa các nhúng cho tác vụ đang thực hiện, dẫn đến hiệu suất & hiệu quả cao hơn so với việc sử dụng các nhúng được tạo sẵn, ví dụ như từ N2V.

That said, while GNN embeddings offer clear advantages in terms of adaptability & applicability to new data, N2V embeddings have their strengths, particularly in capturing nuanced patterns within a specific graph's structure. In practice, choice between GNN & N2V embeddings may depend on specific requirements of task, nature of graph data, & constraints of computational environment.

– Tuy nhiên, trong khi nhúng GNN mang lại những lợi thế rõ ràng về khả năng thích ứng & khả năng ứng dụng cho dữ liệu mới, nhúng N2V cũng có những điểm mạnh riêng, đặc biệt là trong việc nắm bắt các mẫu hình phức tạp trong cấu trúc đồ thị cụ thể. Trên thực tế, việc lựa chọn giữa nhúng GNN & N2V có thể phụ thuộc vào các yêu cầu cụ thể của tác vụ, bản chất của dữ liệu đồ thị, & các ràng buộc của môi trường tính toán.

For tasks where graph structure is static & well-defined, N2V might provide a simpler & computational efficient solution. Conversely, for dynamic graphs, large-scale applications, or scenarios requiring incorporation of node features, GNNs will often be the more robust & versatile choice. Additionally, when task itself is not well-defined & work is exploratory, N2V is likely faster & easier to use.

– Đối với các tác vụ có cấu trúc đồ thị tĩnh & được xác định rõ ràng, N2V có thể cung cấp một giải pháp đơn giản hơn & hiệu quả về mặt tính toán. Ngược lại, đối với các đồ thị động, ứng dụng quy mô lớn hoặc các tình huống yêu cầu tích hợp các đặc điểm nút, GNN thường là lựa chọn mạnh mẽ & linh hoạt hơn. Ngoài ra, khi bản thân tác vụ không được xác định rõ & công việc mang tính khám phá, N2V có thể nhanh hơn & dễ sử dụng hơn.

Have now successfully built our 1st GNN embedding. This is key 1st step for all GNN models, & everything from this point will build on it. In next sect, give an example of some of these next steps & show how to use embeddings to solve a ML problem.

– Chúng ta đã xây dựng thành công mô hình nhúng GNN đầu tiên. Đây là bước đầu tiên quan trọng cho tất cả các mô hình GNN, & mọi thứ từ đây sẽ được xây dựng dựa trên nó. Trong phần tiếp theo, hãy đưa ra ví dụ về một số bước tiếp theo & trình bày cách sử dụng nhúng để giải quyết bài toán học máy.

- 2.3. Using Node Embeddings. Semi-supervised learning, which involves a combination of labeled & unlabeled data, provides a valuable opportunity to compare different embedding techniques. In this chap, explore how GNN & N2V embeddings can be used to predict labels when majority of data lacks labels.

  – Sử dụng Nhúng Nút. Học bán giám sát, bao gồm sự kết hợp giữa dữ liệu có nhãn & không có nhãn, mang đến một cơ hội quý giá để so sánh các kỹ thuật nhúng khác nhau. Trong chương này, hãy khám phá cách sử dụng nhúng GNN & N2V để dự đoán nhãn khi phần lớn dữ liệu thiếu nhãn.

  Our task involves Political Books dataset `books_graph`, where nodes represent political books & edges indicate co-purchase relationships. To make process clearer, review steps taken so far & outline our next steps, as illustrated in Fig. 2.9: Overview of steps taken in Chap. 2: 1. Preprocess Political Books dataset for embedding. 2. Use N2V & GCN to create embeddings from preprocessed data. 3. Prepare N2V embeddings & GCN embeddings for semi-supervised classification. 4. Embeddings are used as features in a random forest classifier (tabular features) & a GCN classifier (node features).

  – Nhiệm vụ của chúng tôi liên quan đến tập dữ liệu Sách Chính trị `books_graph`, trong đó các nút biểu diễn các cuốn sách chính trị & các cạnh biểu diễn mối quan hệ đồng mua. Để làm rõ quy trình, hãy xem lại các bước đã thực hiện cho đến nay & phác thảo các bước tiếp theo, như minh họa trong Hình 2.9: Tổng quan các bước đã thực hiện trong Chương 2: 1. Tiền xử lý tập dữ liệu Sách Chính trị để nhúng. 2. Sử dụng N2V & GCN để tạo nhúng từ dữ liệu đã được xử lý trước. 3. Chuẩn bị nhúng N2V & nhúng GCN cho phân loại bán giám sát. 4. Nhúng được sử dụng làm đặc trưng trong bộ phân loại rừng ngẫu nhiên (đặc trưng dạng bảng) & bộ phân loại GCN (đặc trưng nút).

  Began with `books_graph` dataset in graph format & performed light preprocessing to prepare data for embedding. For N2V, this involved converting dataset from a `.gml` file to a `NetworkX` format. For GNN-based embeddings, converted `NetworkX` graph into a PyTorch tensor & initialized node features using Xavier initialization to ensure balanced variability across layers.

  – Bắt đầu với tập dữ liệu `books_graph` ở định dạng đồ thị & thực hiện tiền xử lý nhẹ để chuẩn bị dữ liệu cho nhúng. Đối với N2V, việc này bao gồm chuyển đổi tập dữ liệu từ tệp `.gml` sang định dạng `NetworkX`. Đối với nhúng dựa trên GNN, chuyển đổi đồ thị `NetworkX` thành tensor PyTorch & khởi tạo các đặc trưng nút bằng khởi tạo Xavier để đảm bảo tính biến thiên cân bằng giữa các lớp.

  After preparing data, we generated embeddings using both N2V & GCNs. Now, in this sect, apply these embeddings to a semi-supervised classification problem. This involves further processing to define classification task, where only 20% of book labels are retained, simulating a realistic scenario with sparse labeled data.

  – Sau khi chuẩn bị dữ liệu, chúng tôi đã tạo ra các nhúng sử dụng cả N2V & GCN. Trong phần này, chúng tôi áp dụng các nhúng này vào một bài toán phân loại bán giám sát. Điều này bao gồm việc xử lý thêm để xác định tác vụ phân loại, trong đó chỉ giữ lại 20% nhãn sách, mô phỏng một kịch bản thực tế với dữ liệu được gắn nhãn thưa thớt.

  Use 2 sets of embeddings (N2V & GCN) with 2 different classifiers: a random forest classifier (to use embeddings as tabular features) & a GCN classifier (to use graph structure & node features). Goal: predict political orientation of books, with remaining 80% of labels inferred based on given embeddings.

  – Sử dụng 2 bộ nhúng (N2V & GCN) với 2 bộ phân loại khác nhau: một bộ phân loại rừng ngẫu nhiên (để sử dụng nhúng làm đặc trưng dạng bảng) & một bộ phân loại GCN (để sử dụng cấu trúc đồ thị & đặc trưng nút). Mục tiêu: dự đoán khuynh hướng chính trị của sách, với 80% nhãn còn lại được suy ra dựa trên các nhúng đã cho.

  * 2.3.1. Data preprocessing. To start, we do a little more preprocessing to our `books_gml` dataset Listing 2.5: Preprocessing for semi-supervised problem. Must format labels in a suitable way for learning process. Because all nodes are labeled, we also have to set up semi-supervised problem by randomly selecting nodes from which we hide labels.

    – Tiền xử lý dữ liệu. Để bắt đầu, chúng ta thực hiện thêm một chút tiền xử lý cho tập dữ liệu `books_gml` Liệt kê 2.5: Tiền xử lý cho bài toán bán giám sát. Phải định dạng nhãn theo cách phù hợp cho quá trình học. Vì tất cả các nút đều được gắn nhãn, chúng ta cũng phải thiết lập bài toán bán giám sát bằng cách chọn ngẫu nhiên các nút mà chúng ta ẩn nhãn. Nodes associated with attribute `'c'` are classified as `'right'`, while those with `'l'` are classified as `'left'`. Nodes that do not fit these criteria, including those with neutral or unspecified attributes, are categorized as `'neutral'`. These

- 2.4. Under Hood.

- 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.

1.2 Geeks4Geeks/What are Graph Neural Networks?

https://www.geeksforgeeks.org/deep-learning/what-are-graph-neural-networks/.

2 Miscellaneous