**PHẦN 2. CƠ SỞ LÝ THUYẾT**

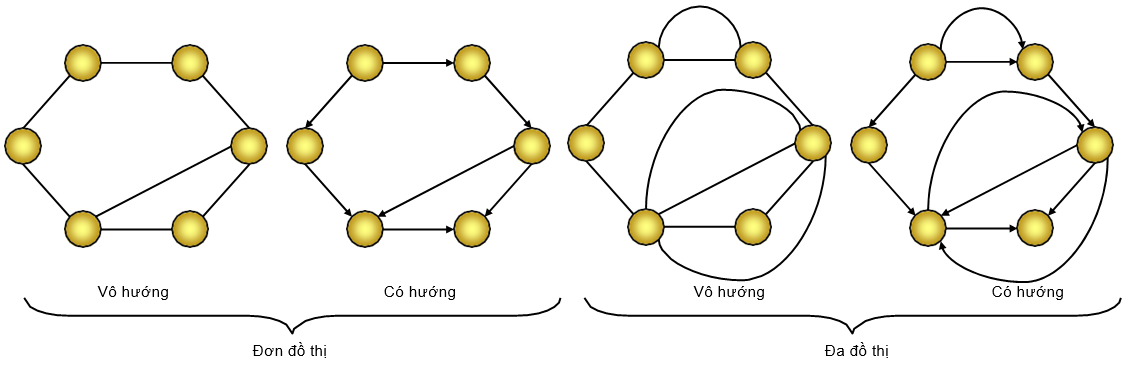
**CHƯƠNG 1: LÝ THUYẾT CƠ BẢN LIÊN QUAN ĐẾN ĐỒ THỊ**

**1.1. Lý thuyết đồ thị cơ bản**

Đồ thị là một cấu trúc rời rạc, trong đó gồm một tập các đối tượng được gọi là các đỉnh (hoặc nút) nối với nhau bởi các cạnh (hoặc cung).

Đồ thị được gọi là có hướng hay vô hướng tùy thuộc vào các cạnh trong đồ thị có hướng hay không. Trong đồ thị nếu như giữa đỉnh bất kỳ có nhiều hơn một cạnh nối hai đỉnh đó thì được gọi là đa đồ thị, ngược lại thì gọi là đơn đồ thị.

Ngoài các khái niệm trên, trong đồ thị còn có một số các khái niệm khác có thể kể đến như: tính kề nhau, bậc, khuyên đường dẫn, …



Hình 1. Hình minh họa về các loại đồ thị

Lý thuyết đồ thị (graph theory) là một nhánh trong lĩnh vực của toán học, nó nghiên cứu về các khái niệm, tính chất và cấu trúc của một đồ thị.

<https://en.wikipedia.org/wiki/Graph_theory>

<https://viblo.asia/p/gioi-thieu-ve-ly-thuyet-do-thi-07LKXQ1eZV4>

**1.2. Mạng Neural (Neural Network) cơ bản**

Neural network (hay còn gọi là mạng nơ-ron) là mô hình toán học hay mô hình tính toán phức tạp được xây dựng để xử lý thông tin. Mạng nơ-ron được xây dựng dựa trên cấu trúc và cách hoạt động của các tế bào thần kinh trong não con người. Tương tự như bộ não con người, các nút trong mạng nơ-ron cũng liên kết với nhau tạo thành một mạng lưới các nút.

<https://en.wikipedia.org/wiki/Neural_network_(machine_learning)>

<https://bizflycloud.vn/tin-tuc/neural-network-la-gi-20220512174329187.htm>

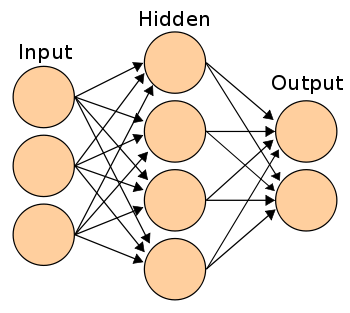
Mỗi một mạng nơ-ron gồm có ba thành phần chính:

* Tầng vào (input layer): Tầng này gồm các nút nhận dữ liệu đầu vào cho mô hình.
* Tầng ẩn (hidden layer): Là tầng nằm giữa của mô hình, thể hiện quá trình suy luận logic của mô hình. Tầng này sẽ nhận vào tập hợp các kết quả đầu ra của các nút ở lớp phía trước và các trọng số liên kết. Sau đó áp dụng hàm kích hoạt của nút để tạo ra kết quả đầu ra.
* Tầng ra (output layer): Tầng này gồm các nút trả ra kết quả của dữ liệu sau khi đưa vào mô hình xử lý.

Các hàm kích hoạt có thể áp dụng trong mô hình có thể là hàm sigmoid, hàm Tanh, hàm Relu, hàm Leaky Relu, …

Trong mạng nơ-ron gồm có hai quá trình:

* Quá trình lan truyền xuôi (feedforward): Là quá trình truyền dữ liệu từ tầng, vào qua tầng ẩn, rồi đến tầng ra để trả ra kết quả.
* Quá trình lan truyền ngược (back propagation): Là quá trình dựa trên kết quả của quá trình lan truyền xuôi so với kết quả đầu ra mong muốn để tính hành cập nhật các trọng số liên kết cho mô hình, cải thiện hiệu suất và kết quả của mô hình.



Hình 2. Hình ảnh minh họa về cấu trúc của một Neural Network

**CHƯƠNG 2: GRAPH NEURAL NETWORKS (GNNs)**

**2.1. Giới thiệu về Graph Neural Networks**

**2.1.1. Lịch sử hình thành của Graph Neural Networks**

Bắt đầu từ những năm 1980, khi các nhà nghiên cứu tìm cách áp dụng mạng nơ-ron cho dữ liệu có cấu trúc đồ thị. Đến những năm 1990, mô hình Recursive Neural Network (RNN) được phát triển để xử lý dữ liệu dạng cây, đặt nền móng cho GNNs sau này.

Thuật ngữ “Graph Neural Networks” được giới thiệu chính thức vào năm 2008 bởi Scarselli và cộng sự. GNNs được phát triển để mô hình hóa các hệ thống động sử dụng dữ liệu đồ thị và nhanh chóng trở thành một lĩnh vực nghiên cứu và ứng dụng quan trọng. Kể từ đó, nhiều kiến trúc GNNs khác nhau đã được đề xuất, triển khai các phương pháp truyền thông điệp khác nhau, bắt đầu bằng các cách tiếp cận đệ quy hoặc tích chập.

Tuy nhiên, phải đến đầu những năm 2010, GNNs mới bắt đầu trở nên phổ biến nhờ sự phát triển của các kiến ​​trúc và thuật toán học tập mới. Tới giai đoạn 2015- 2016, nhờ sự phát triển của Graph Convolutional Networks (GCNs) bởi Thomas Kipf và Max Welling đã mở ra một hướng tiếp cận mới cho việc xử lý dữ liệu trên đồ thị.

Kể từ đó, GNNs tiếp tục phát triển với sự xuất hiện của nhiều biến thể và ứng dụng mới. Các nhà nghiên cứu đang tập trung vào việc tối ưu hóa hiệu suất và áp dụng GNNs vào nhiều lĩnh vực thực tế như hóa học, dự báo tài chính, và y học.

<https://medium.com/@saluem/graph-neural-networks-gnn-93b32567a6d9>

**2.1.2. Định nghĩa về Graph Neural Networks**

Mạng nơ-ron đồ thị (Graph Neural Networks – GNNs) là một loại mạng nơ-ron nhân tạo được thiết kế để thực hiện xử lý và suy luận các loại dữ liệu được biểu diễn dưới dạng đồ thị.

<https://en.wikipedia.org/wiki/Graph_neural_network>

**2.2. Các kiến trúc phổ biến của Graph Neural Networks**

**2.2.1. Graph Convolutional Networks (GCNs)**

**2.2.2. Graph Sample and Aggregation (GraphSAGE)**

**2.2.3. Graph Attention Networks (GATs)**

**2.2.3.1. Định nghĩa và lý do ra đời**

- Graph Attention Network (GAT) là một kiến trúc mạng nơ-ron mới hoạt động dựa trên dữ liệu có cấu trúc đồ thị, được giới thiệu bởi Petar Veličković và các cộng sự vào năm 2018. GAT là sự kết hợp giữa mạng thần kinh đồ thị (GNNs) và lớp chú ý (attention layer), sử dụng các lớp tự chú ý được che giấu để giải quyết những hạn chế của các phương pháp trước đó dựa trên tích chập đồ thị hoặc các xấp xỉ của chúng.

Với các mô hình truyền thống hoặc cụ thể hơn là mô hình GNNs thì các mô hình đó thường giả định rằng tất cả các nút trong một đồ thị đều có quan hệ như nhau. Tuy nhiên, điều này thường không đúng với thực tế. Trong một số trường hợp, một số nút trong một đồ thị có liên quan chặt chẽ hơn với nhau hơn những nút khác. Mà khi ta sử dụng các mô hình đó có thể làm tổn hại đến tính tổng quát của dữ liệu có cấu trúc đồ thị do sự tổng hợp thông tin của cấu trúc đồ thị, cụ thể ở đây mà ta có thể kể đến như GCN. Chính vì điều đó, họ đã tận dụng được khả năng xử lý dữ liệu có cấu trúc đồ thị của GNNs và cơ chế trích xuất thông tin có hữu ích từ dữ liệu lớn của Attention Layer lại với nhau để tạo ra GAT.

Veličković, P. (2017, October 30). Graph Attention Networks. arXiv.org. <https://arxiv.org/abs/1710.10903>

**2.2.3.2. Kiến trúc của mô hình**

GAT khác với các mô hình GNNs khác ở chỗ là nó sự kết hợp với graph attentional layer (GAT layer). Mục tiêu của GAT layer là tạo ra một tập hợp các vectơ đặc trưng mới của các đỉnh trong đồ thị. Sau đó, ta có thể áp dụng các bước tiếp theo của GNNs như bình thường để thực hiện các bài toán khác nhau. GAT layer hoạt động như sau:

- Tính trọng số chú ý (Attention Weights):

+ Đầu tiên ta có đầu vào là tập hợp các vectơ đặc trưng của các đỉnh trong đồ thị, được biểu diễn bằng .Trong đó: N là số nút, F là số lượng các đặc trưng của mỗi nút.

+ Sau đó, ta sử dụng một biến đổi tuyến tính để chuyển đổi đặc trưng của mỗi đỉnh thành một không gian đặc trưng mới: , trong đó là vectơ đặc trưng của đỉnh *i*.

+ Sau đó, chúng ta tính toán trọng số chú ý (attention weights) cho mỗi cặp đỉnh bằng cách sử dụng một cơ chế chú ý độc lập, trong đó mỗi cặp đỉnh được biểu diễn bởi hai vectơ đặc trưng tương ứng:

Trong đó, là vectơ trọng số của cơ chế chú ý với a: , và LeakyReLU là hàm kích hoạt phi tuyến.

- Chuẩn hóa trọng số chú ý:

+ Chuẩn hóa trọng số chú ý bằng cách sử dụng hàm softmax, biến chúng thành các hệ số chuẩn hóa tổng hợp thành 1:

Trong đó, *Nj* là tập hợp các đỉnh lân cận của đỉnh *i* trong đồ thị.

- Tính tổ hợp tuyến tính của đặc trưng:

+ Sử dụng các trọng số chú ý chuẩn hóa để tính tổ hợp tuyến tính của các đặc trưng của các đỉnh lân cận: . Với là hàm kích hoạt phi tuyến tùy chọn.

- Mở rộng sang chú ý đa phần (Multi-Head Attention):

+ Để cải thiện ổn định và đa dạng của quá trình học, chúng ta có thể sử dụng nhiều cơ chế chú ý độc lập (multi-head attention) trong đó mỗi cơ chế chú ý đại diện cho một "đầu".

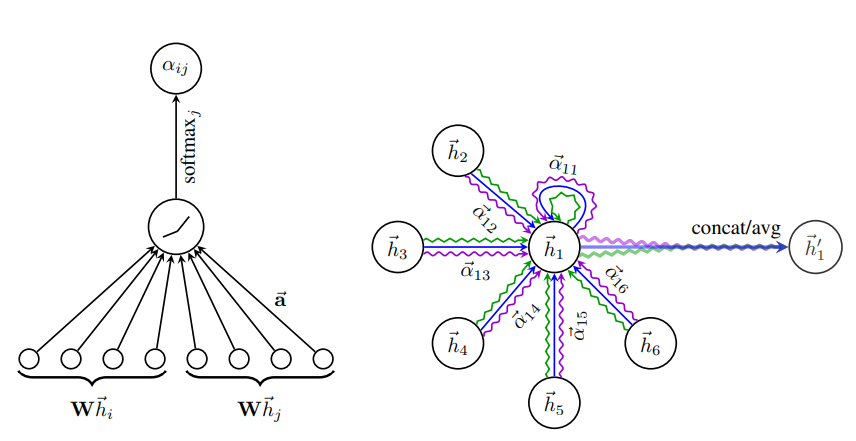
+ Quá trình này áp dụng cho toàn bộ hoặc một phần của đồ thị. Thông thường, trong quá trình này, chúng ta sẽ có nhiều cơ chế chú ý (heads) độc lập, và kết quả từ mỗi head sẽ được kết hợp hoặc tổng hợp để tạo ra đặc trưng cuối cùng cho mỗi nút.

Trong đó, *K* là số lượng cơ chế chú ý đa phần, *Wk* là ma trận trọng số tương ứng với cơ chế chú ý thứ *k* và là các hệ số chú ý chuẩn hóa được tính bằng cách sử dụng cơ chế chú ý thứ *k.*

- Xử lý lớp cuối cùng (Final Layer Processing):

+ Nếu như ta thực hiện multi-head attention trên layer cuối của mạng, việc sử dụng concat sẽ không còn hợp lý. Thay vào đó, chúng ta tính trung bình cộng của chúng và sau đó áp dụng một hàm kích hoạt cuối cùng để tạo ra kết quả cuối cùng của Attention Layer.

+ Hàm kích hoạt này thích hợp cho nhiệm vụ cụ thể như phân loại hoặc dự đoán, thường là softmax cho phân loại nhiều lớp hoặc sigmoid cho phân loại nhị phân.



Veličković, P. (2017, October 30). Graph Attention Networks. arXiv.org. <https://arxiv.org/abs/1710.10903>

**2.2. Ứng dụng của Graph Neural Networks**

GNNs được ứng dụng ở nhiều lĩnh vực trong thực tế. Một số ứng dụng có thể kể đến như:

- Phân tích mạng xã hội: Dùng GNNs để tìm hiểu và phân tích cấu trúc của một mạng xã hội. Từ đó có thể đưa ra những gợi ý, đề xuất kết bạn hoặc theo dõi cho người dùng, …

- Khám phá các loại thuốc: Dùng GNNs để tìm hiểu và phân tích sự tương tác giữa các loại nguyên tử và phân tử có trong các loại thuốc. Rồi từ đó có thể dự đoán được tiềm năng và công dụng của các loại thuốc trong từng trường hợp cụ thể.

- Xử lý ngôn ngữ tự nhiên: Có thể phân tích một đoạn văn bản bằng các sử dụng GNNs khi biểu diễn mối quan hệ giữa các từ có trong đoạn văn thành một sơ đồ.

- Phát hiện gian lận: Có thể phát hiện việc gian lận trong kinh doanh bằng việc mô hình hóa các giao dịch tài chính dưới dạng sơ đồ, rồi sau đó dùng GNNs để tiến hành phân tích và xử lý.

<https://www.xenonstack.com/blog/graph-neural-network-applications>

**2.3. Hạn chế của Graph Neural Networks**

Mặc dù hiện nay, GNNs được ứng dụng trong nhiều vấn đề của thực tế, nhưng GNNs vẫn gặp nhiều hạn chế mà các nhà khoa học nghiên cứu đang cố gắng khắc phục. Một số hạn chế có thể kể đến như:

* Hầu hết các mạng thần kinh có thể đi sâu để có được hiệu suất tốt hơn, trong khi GNNs lại bị hạn chế về hiệu suất khi gặp các tập dữ liệu lớn.
* GNNs đối mặt nhiều vấn đề về mở rộng hoặc áp dụng vào trong sản xuất do mô hình này rất tốn kém về mặt tính toán và lưu trữ nếu gặp cấu trúc đồ thị lớn và phức tạp.
* Việc đào tạo mô hình GNNs trở nên khó khăn hơn khi cấu trúc đồ thị liên tục thay đổi.

<https://www.xenonstack.com/blog/graph-neural-network-applications>

**2.4. Tương lai của Graph Neural Networks**

Trong tương lai, GNNs không chỉ phát triển trong nhiều lĩnh vực và ứng dụng cụ thể, mà GNNs sẽ còn tiếp tục phát triển và cải thiện thêm nhiều mô hình kỹ thuật của GNNs. Một số cải tiến trong tương lai có thể kể đến như:

* Có thể kết hợp thêm nhiều cơ chế và kỹ thuật khác để cải thiện hiệu suất mô hình.
* GNNs có thể cải thiện khả năng mở rộng và tốc độ tính toán đối với các đồ thị lớn và phức tạp hơn.
* Hiện nay trong lĩnh vực học có giám sát thì GNNs được nghiên cứu khá nhiều, nếu trong tương lai có khả năng thì GNNs có thể phát triển sang lĩnh vực học không giám sát.

<https://www.xenonstack.com/blog/graph-neural-network-applications>

**PHẦN 3. ỨNG DỤNG CỦA GRAPH NEURAL NETWORK TRONG BÀI TOÁN LINK PREDICTION**

**CHƯƠNG 1. GIỚI THIỆU VỀ BÀI TOÁN LINK PREDICTION**

**1.1. Định nghĩa bài toán Link Prediction trong mạng đồ thị**

Dự đoán liên kết (Link Prediction) trong mạng đồ thị là việc dự đoán sự tồn tại của một liên kết trong tương lai giữa hai nút, biết rằng không có mối liên hệ nào giữa các nút đó ở trạng thái hiện tại của đồ thị. Nó là một ứng dụng quan trọng của GNNs.

Ví dụ về dự đoán liên kết có thể là dự đoán sự kết bạn giữa hai người dùng trong mạng xã hội, dự đoán sự tương tác giữa gen và protein trong mạng sinh học, …

<https://en.wikipedia.org/wiki/Link_prediction>

**1.2. Cách tiếp cận và phương pháp**

**1.2.1. Các phương pháp truyền thống**

Có 3 lớp phương pháp link prediction truyền thống: lớp phương pháp heuristic (heiristic methods), lớp phương pháp dữ trên đặc trưng ẩn (tatent- feature methods), lớp phương pháp dự trên nội dung (content- base methods).

**1.2.1.1. Phương pháp heuristic (heiristic methods)**

Với x và y lần lượt là nút nguồn và nút đích trong việc dự đoán liên kết. Khí đó, ta có F(x) là tập hợp các nút hàng xóm của x, cũng như F(y) là tập hợp các nút hàng xóm của y.

**1.2.1.1.1. Local Heuristic**

- Một phương pháp heuristic đơn giản nhất trong phần này là common neighbors (CNs), trong đó thì ta sẽ đếm xem giữa các hàng xóm của hai nút cần dự đoán thì có bao nhiêu nút chung, nếu càng nhiều nút chung thì khả năng liên kết của 2 nút trong tương lai là cao.

- Ngoài phương pháp common neighbors, thì còn có Jaccard Score (dựa vào việc tính tỷ lệ hàng xóm chung so với tổng số hàng xóm của cả 2 nút cần dự đoán), Preferential Attachment (PA - dựa trên nguyên tắc là đỉnh có nhiều liên kết hiện tại sẽ có xu hướng tạo ra nhiều liên kết mới hơn trong tương lai), Adamic-Adar (được tính dựa trên độ nút của các nút có liên kết kề chung với 2 nút cần dự đoán), …

**1.2.1.1.2. Global Heuristic**

- Katz Index là phương pháp dựa trên đường đi ngắn nối liền 2 đỉnh cần dự đoán với nhau, hay nói cách khác thì nếu hai đỉnh gần nhau trong mạng, có nhiều đường đi ngắn kết nối chúng, thì khả năng có liên kết giữa chúng là cao:

Trong đó: b là hệ số giảm dần nằm trong khoảng 0 đến 1, đếm số lượng đường đi có độ dài l bước giữa x và y.

- Ngoài phương pháp Katz Index, thì trong phần này còn một số phương pháp khác có thể kể đến như Rooted PageRank (một hướng mở rộng của PageRank), SimRank (một phương pháp dựa trên giả thiết rằng hai nút sẽ càng tương tự nhau nếu các nút láng giềng của chúng cũng tương tự nhau)

**1.2.1.2. Phương pháp dựa trên đặc trưng ẩn (Tatent- feature methods)**

Là lớp phương pháp thứ 2 trong phương pháp link prediction truyền thống. Lớp này sẽ tính toán các đặc trưng ẩn hay biểu diễn ẩn của các nút, thường thu thấp bằng cách phân tích ma trận xấp xỉ từ mạng như ma trận kề hoặc ma trận xác suất chuyển tiếp.

**1.2.1.2.1. Phương pháp phân tích ma trận (Matrix Factorization)**

Phương pháp phân tích ma trận (Matrix Factorization) là một trong những phương pháp Latent-Feature Methods phổ biến nhất trong dự đoán liên kết. Matrix Factorization là một phương pháp trong đó ma trận kề của mạng được phân tách thành hai ma trận nhỏ hơn có kích thước thấp hơn.

Trong đó:

- Ma trận kề A mô tả mối liên hệ hiện có trong mạng.

- Ma trận biểu diễn ẩn Z có kích thước N x k, với k << N. Mỗi hàng của Z cho biểu diễn vector k-chiều của một nút.

- Phương pháp này tối ưu hóa sai số của A so với tích ZZT nhằm học được ma trận biểu diễn ẩn.

Sau khi được học, biểu diễn vector của hai nút được sử dụng để dự đoán xác suất kết nối của chúng:

**1.2.1.2.2. Phương pháp Network Embedding**

Network embedding là một phương pháp Latent-Feature Methods phổ biến để học biểu diễn vector cho các nút trong mạng. Là một phương pháp trong đó mỗi đỉnh trong mạng được ánh xạ vào một không gian vectơ có số chiều thấp hơn. Thuật toán bao gồm các bước tuần hoàn trên mạng, mỗi bước trích xuất một cặp nút gần nhau rồi cập nhật vector đại diện cho chúng. Phương pháp này thường sử dụng các mô hình học sâu như Skip-gram, DeepWalk, Node2vec, hoặc các biến thể của các mô hình này để học các vectơ nhúng cho các đỉnh. Đầu ra là các vector đặc trưng kích thước thấp hơn cho mỗi nút. Vector này được sử dụng thay cho nút trong các bài toán phân loại, dự đoán.

**1.2.1.3. Phương pháp dựa trên nội dung (Content- base methods)**

Content-based methods sử dụng các đặc trưng riêng của đỉnh (explicit content features) liên kết với các đỉnh để dự đoán liên kết.

Content-based methods thường có hiệu suất kém hơn các phương pháp Heuristic và Latent-feature methods do không khai thác cấu trúc đồ thị. Do đó, chúng thường được kết hợp với hai loại phương pháp kia nhằm tăng cường hiệu suất dự đoán liên kết.

**1.2.2. Áp dụng Graph Neural Networks cho bài toán Link prediction**

**1.2.2.1. Phương pháp dựa trên nút (Node-Based Methods)**

Các phương pháp dựa trên nút đầu tiên học cách biểu diễn một nút thông qua một GNNs, sau đó tổng hợp các biểu diễn của các cặp nút để tạo ra biểu diễn cho liên kết (link representation) và dùng để dự đoán liên kết.

Graph AutoEncoder (GAE) là một trong những mô hình phổ biến của phương pháp này. Phương pháp này tập trung vào việc học biểu diễn nút và sử dụng thông tin về cặp nút để dự đoán liên kết. Graph AutoEncoder (GAE) gồm có hai bộ chính là:

- Bộ mã hóa (encoder): Bộ này sẽ áp dụng GNNs để biến đổi biểu đồ đầu vào thành một biểu diễn tiềm ẩn (latent representation) có chiều thấp hơn.

- Bộ giải mã (decoder) nhằm tái tạo lại biểu đồ từ biểu diễn tiềm ẩn đã học trước đó.

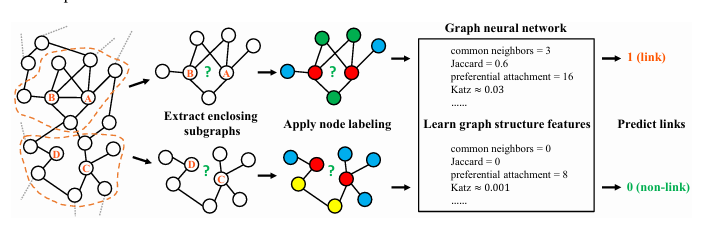
Có nhiều biến thể của GAE. Mỗi một biến thể lại có một đặc trưng riêng nhằm tối ưu mô hình, có thể kể đến như: Variational Graph AutoEncoder (VGAE), MGAE (Marginalized Graph AutoEncoder), S-VAE (Spherical Variational AutoEncoder), …

**1.2.2.2. Phương pháp dựa trên tiểu đồ thị (Subgraph-Based Methods)**

Phương pháp subgraph-based sẽ chia đồ thị lớn thành các tiểu đồ thị nhỏ xung quanh mỗi liên kết mục tiêu. Sau đó, mỗi tiểu đồ thị này sẽ được đưa vào một mô hình để học biểu diễn của nó. Điều này giúp mô hình học được các đặc trưng cục bộ và cấu trúc của đồ thị xung quanh mỗi liên kết mục tiêu, từ đó dự đoán khả năng tồn tại của liên kết.

**1.2.2.2.1. SEAL Framework (Subgraph-based Embedding and Aggregation Learning Framework)**

Công trình tiên phong của các phương pháp dựa trên đồ thị con là SEAL. Khái quát ý tưởng: Đầu tiên SEAL sẽ tách đồ thị ban đầu thành các đồ thị con cục bộ xung quanh mục tiêu cần dự đoán. Rồi tiến hành gán nhãn cho từng nút để phân biệt vai trò của từng nút trong đồ thị. Sau đó, áp dụng GNNs cho tìm hiểu cấu trúc đồ thị và dự đoán liên kết (Hình minh họa)



Các bước của SEAL được tiến hành cụ thể như sau:

- Trích xuất và gán nhãn đỉnh của đồ thị con:

+ SEAL bắt đầu bằng cách trích xuất đồ thị con *Gh* chứa các đỉnh *x* và *y*, đó là các đỉnh mục tiêu nằm giữa các liên kết mục tiêu. Ngoài các đỉnh mục tiêu thì đồ thị con này còn có các đỉnh và liên kết liên quan trực tiếp đến liên kết mục tiêu. Quá trình trích xuất đồ thị con giúp tập trung vào những thông tin quan trọng nhất cho dự đoán liên kết.

+ Tiếp theo, nó áp dụng một kỹ thuật Double Radius Node Label (DRNL) để gán nhãn số nguyên cho mỗi đỉnh trong đồ thị con. Mục đích là phân biệt các đỉnh dựa trên vai trò của chúng trong đồ thị con. Ví dụ, các đỉnh trung tâm *x* và *y* được phân biệt với các đỉnh khác, và các đỉnh cách xa khác nhau từ *x* và *y* có thể có ý nghĩa cấu trúc khác nhau đối với sự tồn tại của liên kết.

- Mã hóa các đặc trưng:

+ Sau khi có các nhãn số nguyên từ kỹ thuật DRNL, SEAL sẽ chuyển đổi chúng thành các vector mã hóa one-hot hoặc đưa chúng vào một lớp embedding để tạo ra các embedding.

+ Các vector đặc trưng mới này được nối với các đặc trưng nội dung đỉnh ban đầu, nếu có, để tạo ra biểu diễn đặc trưng đỉnh được cập nhật.

+ Ngoài ra, SEAL cũng cho phép nối các embedding đỉnh được huấn luyện trước, như các embedding của kỹ thuật node2vec, với các đặc trưng đỉnh. Tuy nhiên, dựa trên các thí nghiệm cho thấy việc thêm này không cải thiện đáng kể hiệu suất và có thể làm mất khả năng học có quy định của SEAL.

- Xử lý mạng nơ-ron đồ thị (GNNs):

+ Các đồ thị con đã được xử lý cùng với các biểu diễn đặc trưng đỉnh đã được cập nhật được đưa vào một graph-level GNNs, Dynamic Graph Convolutional Neural Network (DGCNN) để học một hàm phân loại đồ thị.

- Huấn luyện và dự đoán:

+ SEAL sẽ ngẫu nhiên lấy mẫu N liên kết hiện có từ mạng làm các ví dụ huấn luyện tích cực (positive training links) và chọn một số bằng nhau của các liên kết chưa quan sát (có nghĩa là cặp đỉnh ngẫu nhiên chưa có liên kết) làm các ví dụ huấn luyện tiêu cực (negative training links).

+ Sau khi huấn luyện, SEAL sẽ áp dụng GNNs đã được huấn luyện vào các đồ thị con của các cặp đỉnh mới chưa quan sát (cặp đỉnh chưa có liên kết) để dự đoán các liên kết của chúng.

**1.2.2.2.2. Weisfeiler-Lehman Graph Neural Networks**

**1.3. Ứng dụng của bài toán Link prediction**

Dự đoán liên kết có rất nhiều ứng dụng trong các ứng dụng trong thế giới thực. Dưới đây là một số trường hợp sử dụng quan trọng của dự đoán liên kết:

- Dự đoán khách hàng nào có khả năng mua sản phẩm nào trên các thị trường trực tuyến như Amazon từ những thông tin, sở thích của người dùng. Nó có thể giúp đưa ra khuyến nghị sản phẩm tốt hơn.

- Dự đoán sự kết bạn giữa hai người dùng trên các mạng xã hội như facebook. Từ đó, nó giúp cho nền tảng mạng xã hội có thể gởi các đề suất kết bạn mới cho người dùng.

- Có thể đề xuất sự tương tác hoặc hợp tác giữa các nhân viên trong các dự án hoặc trong các tổ chức.

- Dựa trên những thông tin hiện có, nó có thể trích xuất được những hiểu biết quan trọng từ các mạng lưới khủng bố.

<https://www.analyticsvidhya.com/blog/2020/01/link-prediction-how-to-predict-your-future-connections-on-facebook/>

**1.4. Các thách thức của bài toán Link prediction**

- Độ chính xác: Việc dự đoán chính xác các liên kết tiềm năng là không dễ dàng, đặc biệt khi mạng xã hội phát triển và thay đổi liên tục.

- Độ phức tạp: Mạng xã hội là động và phức tạp với số lượng lớn các biến tham số, làm tăng độ khó của việc mô hình hóa và dự đoán.

- Sự đa dạng của mối quan hệ: Các mối quan hệ trong mạng xã hội có thể rất đa dạng và phức tạp, từ bạn bè, đồng nghiệp, đến các mối quan hệ gia đình, yêu cầu các phương pháp tiếp cận khác nhau để hiểu và dự đoán.

- Tính động của mạng: Mạng xã hội không ngừng thay đổi với các nút và cạnh mới được thêm vào, điều này đòi hỏi các mô hình dự đoán phải liên tục cập nhật và thích ứng.

- Dữ liệu thiếu hoặc nhiễu: Trong thực tế, dữ liệu thu thập được có thể không đầy đủ hoặc chứa nhiễu, làm giảm khả năng dự đoán chính xác của mô hình.

- Quy mô: Kích thước lớn của mạng xã hội đòi hỏi các thuật toán phải hiệu quả về mặt tính toán để có thể xử lý dữ liệu lớn mà không làm giảm hiệu suất.

- Sự ảnh hưởng của các nút khác: Các nút khác trong mạng có thể ảnh hưởng đến khả năng hình thành liên kết giữa hai nút cụ thể, làm cho việc dự đoán trở nên phức tạp hơn.

<https://lnhutnam.github.io/post/link-prediction-algorithms/>

**CHƯƠNG 2. THỰC NGHIỆM VÀ ĐÁNH GIÁ**

**2.1. Môi trường cần thiết**

Bài toán được thực hiện trên môi trường chính là Visual Studio Code. Visual Studio Code là mã nguồn mở IDE mạnh dễ sử dụng và được phát triển bởi Microsoft dành cho Windows, Linux và macOS. Visual Studio Code hỗ trợ trên nhiều ngôn ngữ lập trình và nhiều mở rộng khác nhau có thể kể đến như Java, JavaScript, TypeScript, Node.js, Python, C, C++, SQL, Git, Docker, ...

Nhóm sử dụng hệ điều hành Windows với phiên bản 10, Visual Studio Code phiên bản 1.85.1, Python 3.7.16 và sử dụng nhiều thư viện khác nhau của python để tiến hành chạy chương trình. Một số thư viện có thể kể đến như:

- Thư viện Streamlit: Streamlit là một framework mã nguồn mở và miễn phí để nhanh chóng xây dựng và chia sẻ các ứng dụng web khoa học dữ liệu và học máy tuyệt đẹp. Đây là một thư viện dựa trên Python được thiết kế dành riêng cho các kỹ sư học máy để có thể hiển thị dữ liệu và thu thập các tham số cần thiết cho mô hình hóa.

- Thư viện Tensorflow: Tensorflow là một thư viện mã nguồn mở được sử dụng cho tính toán số học và xử lý dữ liệu bằng cách sử dụng đồ thị luồng dữ liệu. Nó được phát triển bởi Google và là một trong những thư viện phổ biến nhất cho học máy và deep learning. Nó cung cấp khả năng tích hợp với nhiều thư viện machine learning, hỗ trợ tốt cho việc xây dựng và huấn luyện các mô hình học máy và nó sử dụng cấu trúc đồ thị (graph) để biểu diễn các phép tính và dữ liệu trong mô hình. <https://www.tensorflow.org/?hl=vi>

- Thư viện Scikit-learn (Sklearn) là một công cụ mạnh mẽ dành cho việc thực hiện các tác vụ học máy và mô hình hóa thống kê bằng ngôn ngữ Python. Nó cung cấp các công cụ đơn giản và hiệu quả cho phân tích dự đoán dữ liệu. Scikit-learn được xây dựng trên các thư viện NumPy, SciPy và matplotlib. <https://scikit-learn.org/>

- Thư viện Pandas: Pandas là một thư viện mã nguồn mở, hỗ trợ đắc lực trong thao tác dữ liệu. Đây cũng là bộ công cụ phân tích và xử lý dữ liệu mạnh mẽ của ngôn ngữ lập trình Python. Thư viện này sử dụng một cấu trúc dữ liệu riêng là Dataframe. <https://pandas.pydata.org/>

- Thư viện NumPy: NumPy là một công cụ mạnh mẽ trong Python dành cho tính toán số học và xử lý dữ liệu. Nó được phát triển bởi Travis Oliphant vào năm 2005 và là một trong những thư viện phổ biến nhất cho khoa học máy tính, tính toán số học trong Python và cung cấp một đối tượng mảng đa chiều hiệu suất cao và các công cụ để làm việc với các mảng này. <https://numpy.org/>

**2.2. Mô tả dữ liệu**

Tập dữ liệu "fb-pages-food" là một tập dữ liệu mạng xã hội, cụ thể là một mạng xã hội trên Facebook liên quan đến các trang có tích xanh về thực phẩm. Tập dữ liệu này được thu thập vào tháng 11 năm 2017 và được trình bày dưới dạng đồ thị vô hướng. Tập dữ liệu chứa thông tin về các trang Facebook và các mối quan tâm chung giữa chúng. Các chú thích và tài liệu tham khảo liên quan đến tập dữ liệu cũng được cung cấp.

Thông tin chi tiết về tập dữ liệu:

- Loại dữ liệu: Mạng xã hội

- Lĩnh vực: Thực phẩm

- Nguồn: Network Repository ( <https://networkrepository.com/fb-pages-food.php> )

- Kích thước tập dữ liệu: 620 nút (đại diện cho các trang facebook có liên quan đến thực phẩm) và hơn 2100 liên kết (đại diện cho mối quan hệ giữa các trang, ví dụ: theo dõi, thích, …)

- Một số thông tin thống kê khác: mật độ (density), bậc tối đa (maximum degree), bậc tối thiểu (minimum degree), bậc trung bình (average degree), hệ số đồng loạt (assortativity), số tam giác, hệ số gom cụm trung bình (average clustering coefficient), và nhiều thông tin khác về cấu trúc mạng.

**2.3. Xử lý dữ liệu**

**2.3.1. SEAL Framework**

Quá trình đọc dữ liệu:

- Đầu tiên, nhóm sẽ tiến hành đọc dữ liệu các cạnh có trong đồ thị, sau đó chuyển các dữ liệu đọc được thành mảng NumPy, rồi lưu mảng đó vào biến “*positive*”.

- Tiếp theo, dựa vào các cạnh có trong đồ thị từ bước trước để vẽ đồ thị vô hướng “*G*”, nhóm tiến hành trích xuất ra các cạnh không tồn tại trong đồ thị, rồi lưu danh sách đó vào biến “*negative*”, với số lượng các cạnh có trong danh sách này bằng với số cạnh có trong “*positive*”.

- Tiếp đến, vì để đảm bảo tính ngẫu nhiên và tính đa dạng trong quá trình chia dữ liệu, nhằm phục vụ cho quá trình huấn luyện và kiểm tra trên mô hình nên nhóm đã tiến hành xáo trộn danh sách “*positive*” và “*negative*”.

- Đồng thời, vì để đảm bảo rằng các chỉ số và giá trị của các cạnh có trong “*positive*” và các cạnh có trong “*negative*” phù hợp với chỉ số và giá trị của các nút trong đồ thị nên đã tiến hành giảm đi 1 nếu giá trị nhỏ nhất trong mảng là 1.

Quá trình nhúng dữ liệu:

-

**2.3.2. TwoWL**

**2.4. Huấn luyện mô hình**

**2.4.1. SEAL Framework**

**2.4.2. TwoWL**

**2.5. Đánh giá và nhận xét kết quả**

**PHẦN 4: KẾT LUẬN**

**1. Kết luận**

**2. Hạn chế**

**3. Hướng phát triển**

**PHẦN 5: TÀI LIỆU THAM KHẢO**