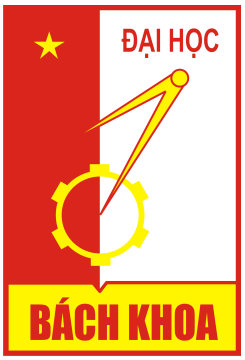
ĐẠI HỌC BÁCH KHOA HÀ NỘI
KHOA TOÁN - TIN



Ứng dụng nhóm mô hình Graph-based xây dựng bài toán dự đoán bệnh dựa trên triệu chứng

ĐỒ ÁN II

Chuyên ngành: Toán - Tin

Chuyên sâu: Toán ứng dụng

Giảng viên hướng dẫn: TS. Đoàn Duy Trung

Sinh viên thực hiện: Doãn Chí Thường

Mã số sinh viên: 20210831

HÀ NỘI – 2024

NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN

1. Mục tiêu và nội dung của đồ án

(a) Mục tiêu:

(b) Nội dung:

2. Kết quả đạt được

(a)

(b)

(c)

3. Ý thức làm việc của sinh viên:

(a)

(b)

(c)

Hà Nội, ngày ... tháng ... năm 2024

Giảng viên hướng dẫn

TS. Đoàn Duy Trung

Tóm tắt nội dung Đồ án

1. **Chương 1: Cơ sở lý thuyết** sẽ trình bày các kiến thức cơ bản liên quan đến đồ thị, kiến trúc *Graph Neural Network*, lý thuyết về phương pháp nhúng nông *Shallow Embedding*, quy tắc *Encoder, Decoder* cũng như trình bày về các cơ chế tổng hợp thông tin trong quá trình lan truyền thông điệp trên mạng thần kinh đồ thị.

2. **Chương 2: Ứng dụng nhóm mô hình Graph-based xây dựng bài toán dự đoán bệnh dựa trên triệu chứng** sẽ trình bày về hướng tiếp cận bài toán PDfS. Các phương pháp xây dựng đồ thị, liên kết cạnh và *Node feature*. Đồng thời đưa ra mô hình đề xuất và các chiến lược huấn luyện.

3. **Chương 3: Cài đặt chương trình và trình bày kết quả thử nghiệm** sẽ tiến hành cài đặt chương trình kiểm thử dựa trên các chiến lược tiếp cận đã đề xuất ở **Chương 2** cho 2 bộ dữ liệu *Disease Symptom Prediction* và *Pima Indian Diabetes* [8]. Đánh giá các kết quả thu được.

Hà Nội, ngày 22 tháng 12 năm 2024

Tác giả đồ án

Doãn Chí Thường

Mục lục

Bảng ký hiệu và chữ viết tắt

Bảng thuật ngữ tên viết tắt:

PDfS	Predicting Disease from Symptoms
EMRs	Electronic Medical Records
TF-IDF	Term Frequency-Inverse Document Frequency
GNNs	Graph Neural Networks
GCNs	Graph Convolutional Networks
MLP	Multi Layer Perceptron
GRU	Gated Recurrent Neural Network

Bảng ký hiệu:

$\hat{c}_p \in \{0,1\}^{\|C\|}$	Dãy nhị phân có độ dài $\|C\|$
$\|.\|_k$	Chuẩn vecto hợp lý k
$[.\|.]$	Phép nối hai vecto
$\triangleq$	Gán cho đối tượng toán học là một biểu thức
$A \gg B$	A lớn hơn nhiều lần B
$\mathbb{R}^+$	Tập các vecto không âm

Danh sách hình ảnh

Danh sách bảng

Mở đầu

Hồ sơ bệnh án điện tử (EMRs), viết tắt của *Electronic Medical Records*, là một hệ thống quản lý dữ liệu thường được sử dụng trong các bệnh viện để lưu trữ thông tin lâm sàng chi tiết từ các lần khám bệnh của bệnh nhân. Trong thời gian gần đây, những tiến bộ trong công nghệ thông tin và học máy đã giúp giảm khối lượng dữ liệu EMRs một cách hiệu quả. Do đó, việc phân tích EMRs bằng các kỹ thuật học máy và khai phá dữ liệu đã trở thành một hướng nghiên cứu nổi bật nhằm nâng cao chất lượng dịch vụ chăm sóc sức khỏe [13].

Một ứng dụng đáng chú ý của học máy trong lĩnh vực y tế là dự đoán bệnh, tập trung vào việc xác định liệu một bệnh nhân có mắc phải một căn bệnh cụ thể hay không. Nhiệm vụ này thường bao gồm việc huấn luyện một bộ phân loại để đưa ra dự đoán dựa trên thông tin được trích xuất từ EMRs [11], [4]. Ví dụ, *Palaniappan* và *Awang* đã áp dụng các kỹ thuật khai phá dữ liệu khác nhau, chẳng hạn như *Decision Tree* [9] và Mạng Nơ-ron [18], để xây dựng một hệ thống dự đoán bệnh tim [15]. Tận dụng khả năng của Mạng Nơ-ron Tích chập *(CNNs)*, Suo và cộng sự [11] đã ban đầu xác định sự tương đồng giữa các bệnh nhân dựa trên EMRs của họ và sau đó thực hiện dự đoán bệnh cá nhân hóa.

Các mạng nơ-ron học sâu hiện tại thiếu sự tương tác và mối liên kết giữa các bệnh nhân trong kiến trúc của chúng. Việc xem xét những mối quan hệ này có thể mang lại lợi ích, vì nó hỗ trợ phân tích và nghiên cứu các nhóm bệnh nhân tương đồng. Đồ thị cung cấp một phương tiện tự nhiên để biểu diễn các tương tác trong một quần thể. Việc xây dựng đồ thị giữa các bệnh nhân dựa trên một tập hợp con các đặc điểm của họ cho phép tóm tắt các đặc điểm này trên các cạnh của đồ thị, giảm chiều dữ liệu đặc trưng và giảm thiểu hiện tượng quá khớp do số lượng đặc trưng quá lớn [3], [7].

Lời cảm ơn

Báo cáo này được thực hiện và hoàn thành tại Đại học Bách Khoa Hà Nội, nằm trong nội dung học phần *Đồ án II* của kì học 2024-1.

Em xin được dành lời cảm ơn chân thành đến **TS. Đoàn Duy Trung** là giảng viên đã và đang trực tiếp hướng dẫn cho em trong *Đồ án* này. Đồng thời thầy cũng đã giúp đỡ tận tình và có những đóng góp bổ ích để em có thể hoàn thành báo cáo này một cách tốt nhất. Em xin chúc thầy thật nhiều sức khỏe, vẫn nhiệt huyết với giáo dục. Em hy vọng có thể được tiếp tục làm việc với thầy trong các công việc không chỉ riêng *Đồ Án*.

Hà Nội, ngày 22 tháng 12 năm 2024
Tác giả đồ án

Doãn Chí Thường

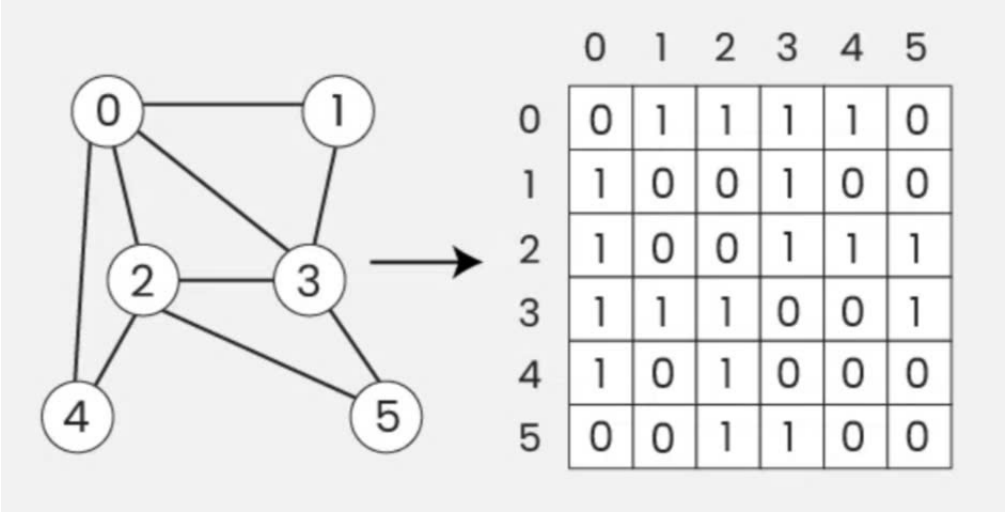
Chương 1

Cơ sở lý thuyết

1.1 Giới thiệu về đồ thị

Một đồ thị $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ được định nghĩa bởi một tập các nút $\mathcal{V}$ và một tập các cạnh $\mathcal{E}$ giữa các nút. Chúng ta ký hiệu một cạnh từ nút $u \in \mathcal{V}$ đến nút $v \in \mathcal{V}$ là $(u, v) \in \mathcal{E}$. Trong nhiều trường hợp, chúng ta sẽ chỉ quan tâm đến các đồ thị đơn giản, trong đó mỗi cặp nút có ít nhất một cạnh, không có cạnh nào giữa một nút và chính nó, và các cạnh đều là không có hướng $(u, v) \in \mathcal{E} \leftrightarrow (v, u) \in \mathcal{E}$.

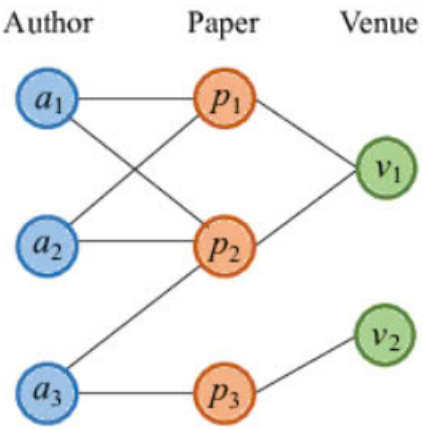
Trong thực tế, để đơn giản quan sát các liên kết giữa các nút sẽ thông qua ma trận kề (*adjacency matrix*) $A \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$. Để biểu diễn một đồ thị bằng ma trận kề, chúng ta sắp xếp các nút trong đồ thị sao cho mỗi nút chỉ mục cho một hàng và một cột cụ thể trong ma trận kề. Chúng ta cần biểu diễn mối liên kết (cạnh) giữa hai nút: $A[u, v] = A_{uv} = 1$ nếu $(u, v) \in \mathcal{E}$ và $A[u, v] = 0$ nếu ngược lại. Ma trận $\mathbf{A}$ sẽ đối xứng nếu đồ thị chỉ chứa các cạnh không có hướng. Tuy nhiên, nếu các cạnh có hướng (tức là phụ thuộc vào hướng giữa các nút), thì ma trận $\mathbf{A}$ sẽ không còn đối xứng nữa. Một số đồ thị cũng có thể có các cạnh có trọng số, trong đó các phần tử trong ma trận kề là các giá trị thực tùy ý thay vì $\{0, 1\}$. Ví dụ, một cạnh có trọng số trong đồ thị tương tác protein-protein có thể chỉ ra mức độ mạnh mẽ của sự liên kết giữa hai protein.



Hình 1.1: Biểu diễn ma trận kề

Đồ thị không đồng nhất (Heterogeneous graphs) [6]

Trong đồ thị không đồng nhất, các nút cũng được gán các kiểu, có nghĩa là chúng ta có thể phân chia tập hợp các nút thành các tập hợp rời rạc $\mathcal{V} = \mathcal{V}_1 \cup \mathcal{V}_2 \cup ... \cup \mathcal{V}_k$ sao cho $\mathcal{V}_i \cap \mathcal{V}_j = \emptyset, \forall i \neq j$. Các cạnh trong những đồ thị này thường bị ràng buộc bởi kiểu của các nút mà chúng kết nối. Điều này có nghĩa là không phải tất cả các nút đều có thể được kết nối bởi bất kỳ loại cạnh nào $(u, \tau_i, v) \in \mathcal{E} \rightarrow u \in \mathcal{V}_j, v \in \mathcal{V}_k$ trong đó có một cạnh τ_i kết nối nút u với nút v, và cạnh này thuộc về tập hợp các cạnh $\mathcal{E}$.

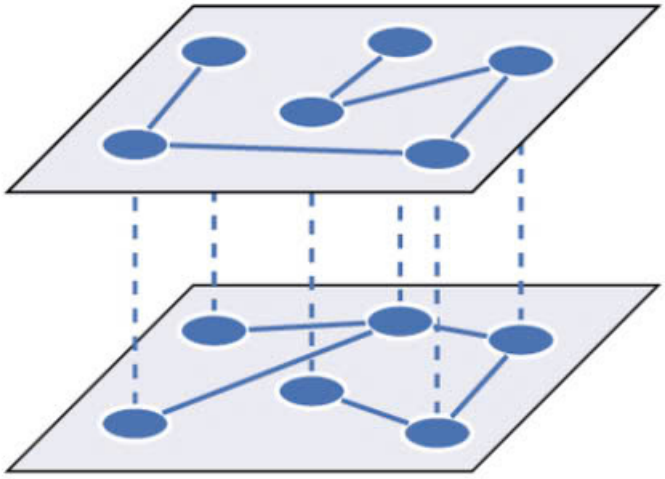


Hình 1.2: Đồ thị không đồng nhất

Đồ thị ghép kênh (Multiplex graphs) [6]

Đồ thị ghép kênh *(Multiplex graphs)* là một loại đồ thị đặc biệt, trong đó tồn tại nhiều loại mối quan hệ hoặc tương tác khác nhau giữa cùng một

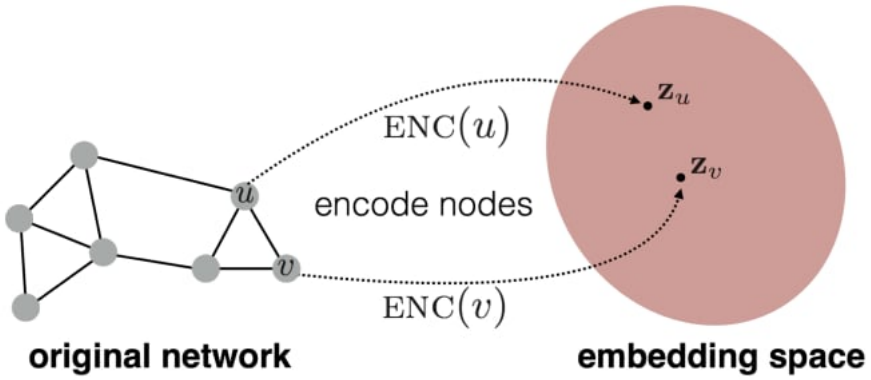
tập hợp các nút. Trong đồ thị ghép kênh, các nút vẫn giữ nguyên qua các lớp hoặc mạng khác nhau, nhưng các cạnh sẽ khác nhau tùy thuộc vào loại mối quan hệ đang được mô hình hóa. Mỗi lớp của đồ thị đại diện cho một loại kết nối khác nhau giữa các nút. Mỗi lớp trong đồ thị ghép kênh đại diện cho một loại mối quan hệ hoặc tương tác khác nhau. Các nút là giống nhau trong tất cả các lớp, nghĩa là cùng một tập hợp thực thể (người, tổ chức, đối tượng, v.v.) tồn tại trong mọi lớp của đồ thị. Những gì thay đổi là các cạnh giữa các nút, tùy thuộc vào mối quan hệ được thể hiện. Mỗi lớp có một tập hợp các cạnh riêng biệt, đại diện cho một loại tương tác cụ thể.



Hình 1.3: Đồ thị ghép kênh

1.2 Các phương pháp tái tạo lân cận (Neighborhood Reconstruction Methods)

Trong phần này, tác giả sẽ trình bày về phương pháp tái cấu trúc lân cận (*Neighborhood Reconstruction Methods* [6]). NRMs tập trung vào việc khai thác và học các biểu diễn của nút hoặc đồ thị, bảo toàn cấu trúc lân cận (neighborhood structure) trong không gian biểu diễn. Nói cách khác là việc chiếu các nút vào một không gian nhúng sao cho các mối quan hệ hình học trong không gian này tương ứng với các mối quan hệ trong biểu đồ hoặc mạng gốc.



Hình 1.4: Minh họa nhúng nút vào không gian giảm chiều

Mục tiêu của việc nhúng nút là học một bộ mã hóa *(enc)*, ánh xạ các nút tới không gian nhúng có chiều thấp hơn. Phần nhúng được tối ưu hóa sao cho khoảng cách trong không gian nhúng phản ánh vị trí tương đối của các nút trong biểu đồ gốc. Việc giảm chiều giúp dữ liệu trở nên gọn nhẹ hơn, dễ dàng và nhanh chóng hơn để thực hiện các tác vụ như phân cụm, phân loại hoặc tìm kiếm tương tự.

Trong khung ý tưởng của encoder-decoder sẽ xem xét việc biểu diễn quan hệ trên đồ thị thông qua hai hoạt động chính. Đầu tiên, một mô hình bộ mã hóa *(encoder)* sẽ ánh xạ từng nút trong biểu đồ thành một vectơ có chiều thấp hoặc nhúng. Tiếp theo, bộ giải mã mô hình *(decoder)* lấy các phần nhúng nút chiều thấp và sử dụng chúng để tái tạo lại thông tin về vùng lân cận của mỗi nút trong biểu đồ gốc.

1.2.1 Góc nhìn bộ mã hóa - giải mã

Bộ mã hóa (The encoder)

Về mặt hình thức, bộ mã hóa là một hàm ánh xạ các nút $v \in \mathcal{V}$ tới các vectơ nhúng $z_v \in \mathbb{R}^d$ (trong đó z_v tương ứng với việc nhúng cho nút $v \in \mathcal{V}$). Trong trường hợp đơn giản nhất, bộ mã hóa được biểu diễn như sau:

$$\text{ENC} : \mathcal{V} \to \mathbb{R}^d \tag{1.1}$$

nghĩa là bộ mã hóa lấy ID nút làm đầu vào để tạo ra các phần nhúng nút. Trong hầu hết công việc về nhúng nút, bộ mã hóa dựa vào phương pháp

gọi là nhúng nông *(shallow embedding)*, trong đó chức năng mã hóa này chỉ đơn giản là một nhúng tra cứu dựa trên ID nút. Nói cách khác có thể biểu diễn như sau:

$$\text{ENC}(v) = Z[v] \tag{1.2}$$

trong đó $Z \in \mathbb{R}^{\mathcal{V} \times d}$ là ma trận chứa vecto nhúng cho tất cả các nút và $Z[v]$ biểu thị hàng Z tương ứng với nút v.

Bộ giải mã (The decoder)

Vai trò của bộ giải mã *(decoder)* là tái tạo lại các thống kê đồ thị nhất định từ các vector nhúng của nút *(node embeddings)* được tạo ra bởi bộ mã hóa *(encoder)*. Ví dụ, với vector nhúng z_u của một nút u, bộ giải mã có thể dự đoán tập các lân cận $\mathcal{N}(u)$ của u hoặc hàng $A[u]$ trong ma trận kề của đồ thị.

Mặc dù có thể sử dụng nhiều loại bộ giải mã khác nhau, nhưng phương pháp tiêu chuẩn thường là định nghĩa các bộ giải mã cặp đôi *(pairwise decoders)*, với cấu trúc như sau:

$$\text{DEC} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^+ \tag{1.3}$$

Các bộ giải mã cặp đôi có thể được hiểu là dự đoán mối quan hệ hoặc độ tương đồng giữa các cặp nút. Ví dụ, một bộ giải mã cặp đôi đơn giản có thể dự đoán liệu hai nút có phải là láng giềng trong đồ thị hay không.

Áp dụng bộ giải mã cặp đôi cho một cặp vector nhúng (z_u, z_v) sẽ dẫn đến việc tái tạo mối quan hệ giữa các nút u và v. Mục tiêu là tối ưu hóa bộ mã hóa *(encoder)* và bộ giải mã *(decoder)* để giảm thiểu hàm mất mát tái tạo, sao cho:

$$\text{DEC}(\text{ENC}(u), \text{ENC}(v)) = \text{DEC}(z_u, z_v) \approx S[u, v] \tag{1.4}$$

Ở đây, chúng ta giả định rằng $S[u, v]$ là một phép đo độ tương đồng dựa trên đồ thị giữa các nút. Mục tiêu tái tạo đơn giản nhằm dự đoán liệu hai nút có phải là láng giềng hay không sẽ tương ứng với $S[u, v] = A[u, v]$. Trong đó, $A[u, v]$ là giá trị trong ma trận kề. Tuy nhiên, $S[u, v]$ cũng có

thể định nghĩa một cách tốt quát hơn thông qua thống kê chồng lấp láng giềng nào đó.

Để đạt được mục tiêu tái tạo (Phương trình 1.4), phương pháp tiêu chuẩn là giảm thiểu hàm mất mát tái tạo thực nghiệm $\mathcal{L}$ trên một tập hợp các cặp nút huấn luyện $\mathcal{D}$:

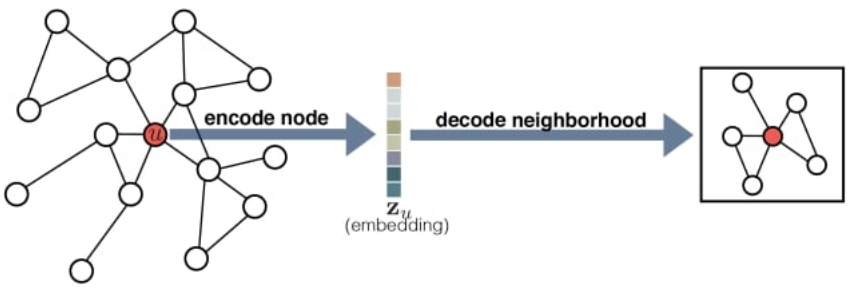
$$\mathcal{L} = \sum_{(u,v)\in\mathcal{D}} \ell(\text{DEC}(z_u, z_v), S[u, v]) \tag{1.5}$$

trong đó, $\ell : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ là một hàm mất mát đo lường sự chênh lệch giữa giá trị tương đồng được giải mã, tức là giá trị ước lượng $\text{DEC}(z_u, z_v)$ và giá trị thực $S[u, v]$. Tùy thuộc vào cách định nghĩa bộ giải mã *decoder* và hàm tương đồng **(S)** mà hàm mất mát ℓ có thể là lỗi bình phương trung bình *(mean-squared error)* hoặc một hàm mất mát phân loại, chẳng hạn như entropy chéo *(cross-entropy)*. Phần lớn các phương pháp tối thiểu hóa hàm mất mát trong (Phương trình 1.5) sử dụng thuật toán *gradient descent* ngẫu nhiên *(stochastic gradient descent)* [Robbins và Monro, 1951]. Tuy nhiên, trong một số trường hợp nhất định, các phương pháp tối ưu hóa đặc biệt hơn (ví dụ, dựa trên phân tích ma trận) có thể được sử dụng.

Method	Decoder	Similarity measure	Loss function
Lap. Eigenmaps	$\|z_u - z_v\|_2^2$	general	$\text{DEC}(z_u, z_v).S[u, v]$
Graph Fact.	$z_u^T.z_v$	$A[u, v]$	$\|\text{DEC}(z_u, z_v) - S[u, v]\|_2^2$
GraRep	$z_u^T.z_v$	$A[u, v], ..., A^k[u, v]$	$\|\text{DEC}(z_u, z_v) - S[u, v]\|_2^2$
HOPE	$z_u^T.z_v$	general	$\|\text{DEC}(z_u, z_v) - S[u, v]\|_2^2$
DeepWalk	$\frac{e^{z_u^T.z_v}}{\sum_{k\in\mathcal{V}} e^{z_u^T.z_k}}$	$p_{\mathcal{G}}(v\|u)$	$-S[u, v]\log((\text{DEC}(z_u, z_v)))$
node2vec	$\frac{e^{z_u^T.z_v}}{\sum_{k\in\mathcal{V}} e^{z_u^T.z_k}}$	$p_{\mathcal{G}}(v\|u)$(biased)	$-S[u, v]\log((\text{DEC}(z_u, z_v)))$

Bảng 1.1: Một số thuật toán nhúng nông (shallow embedding)

các bộ giải mã (decoders) và các hàm tương đồng trong các phương pháp dựa trên random-walk là không đối xứng, với hàm tương đồng $p_{\mathcal{G}(v|u)}$ biểu thị xác suất ghé thăm v trong một random-walk có độ dài cố định bắt đầu từ u.



Hình 1.5: Chiến lược mã hóa - giải mã

1.2.2 Các phương pháp dựa trên phân tích ma trận (Factorization-based approaches)

Bản đồ riêng Laplacian (Laplacian eigenmaps)

Một trong những phương pháp dựa trên phân tích ma trận sớm nhất và có ảnh hưởng lớn nhất là kỹ thuật *Laplacian Eigenmaps (LE)* được xây dựng dựa trên các ý tưởng phân cụm phổ *(spectral clustering)*. Trong phương pháp này, bộ giải mã *(decoder)* được định nghĩa dựa trên khoảng cách L_2 giữa các vector nhúng của các nút:

$$\text{DEC}(z_u, z_v) = \|z_u - z_v\|_2^2$$

Hàm mất mát sau đó gán trọng số cho các cặp nút dựa trên độ tương đồng của chúng trong đồ thị:

$$\ell = \sum_{(u,v)\in\mathcal{D}} \text{DEC}(z_u, z_v).S[u, v] \tag{1.6}$$

Nếu S được xây dựng sao cho nó thỏa mãn các tính chất của một ma trận Laplacian, thì các vector nhúng của nút tối thiểu hóa hàm mất mát trong (Phương trình 1.6) sẽ giống hệt với nghiệm của bài toán phân cụm phổ *(spectral clustering)*.

Phương pháp Inner-product

Các nghiên cứu gần đây thường sử dụng một bộ giải mã dựa trên tích vô hướng *(inner product)*, được định nghĩa như sau:

$$\text{DEC}(z_u, z_v) = z_u^T z_v \tag{1.7}$$

Ở đây, chúng ta giả định rằng độ tương đồng giữa hai nút - ví dụ, mức độ chồng lấp giữa các nút lân cận cục bộ của chúng - tỷ lệ thuận với tích vô hướng *(dot product)* giữa các vector nhúng của chúng. Một số ví dụ về các thuật toán nhúng nút theo cách này bao gồm phương pháp *Graph Factorization (GF)* [Ahmed và cộng sự, 2013], *GraRep* [Cao và cộng sự, 2015], và *HOPE* [Ou và cộng sự, 2016]. Cả ba phương pháp này kết hợp bộ giải mã dựa trên tích vô hướng (Phương trình 1.7) với hàm mất mát bình phương trung bình sau:

$$\ell = \sum_{(u,v)\in\mathcal{D}} \|\text{DEC}(z_u, z_v) - S[u, v]\|_2^2 \tag{1.8}$$

Các phương pháp này được gọi là phương pháp phân tích ma trận (matrix-factorization approaches), vì hàm mất mát của chúng có thể được tối thiểu hóa bằng các thuật toán phân tích, chẳng hạn như phân tích giá trị kỳ dị *(singular value decomposition, SVD)*. Bằng cách xếp chồng các vector nhúng nút $z_u \in \mathbb{R}^d$ thành một ma trận $Z \in \mathbb{R}^{|\mathcal{V}\times d|}$, mục tiêu tái tạo cho các phương pháp này có thể được viết lại dưới dạng:

$$\ell \approx \|ZZ^T - S\|_2^2 \tag{1.9}$$

Điều này tương ứng với một phép phân tích ma trận S (ma trận độ tương đồng nút-nút) thành các thành phần có thứ nguyên thấp. Tổng quát là đằng sau các phương pháp này là học các vector nhúng cho mỗi nút sao cho tích vô hướng giữa các vector nhúng được học gần đúng với một phép đo độ tương đồng nút xác định trước.

1.2.3 Nhúng dựa trên bước nhảy ngẫu nhiên

DeepWalk and node2vec

Tương tự như các phương pháp phân tích ma trận đã được mô tả ở trên, **DeepWalk** và **node2vec** sử dụng cách tiếp cận nhúng nông *(shallow embedding)* cùng với bộ giải mã dựa trên tích vô hướng. Sự khác biệt chính trong các phương pháp này nằm ở cách chúng định nghĩa khái niệm về độ tương đồng giữa các nút và việc tái tạo nút láng giềng.

$$\text{DEC}(z_u, z_v) \triangleq \frac{e^{z_u^T z_v}}{\sum_{v_k \in \mathcal{V}} e^{z_u^T z_k}} \approx p_{\mathcal{G},\mathcal{T}}(v|u) \tag{1.10}$$

Trong đó, $p_{\mathcal{G},\mathcal{T}}(v|u)$ là xác suất ghé thăm v trong một random walk có độ dài T, bắt đầu từ u, T thường được định nghĩa trong khoảng $T \in \{2, ..., 10\}$. Sự khác biệt chính giữa các phương pháp phân tích ma trận và bước nhảy ngẫu nhiên nằm ở hàm xác định độ đo tương đồng. Để huấn luyện các nhúng dựa trên *random walk*, chiến lược tổng quát là sử dụng bộ giải mã từ (Phương trình 1.10) và tối thiểu hóa hàm mất mát entropy chéo sau:

$$\ell = \sum_{(u,v) \in \mathcal{D}} -\log(\text{DEC}(z_u, z_v)) \tag{1.11}$$

Ở đây, chúng ta sử dụng $\mathcal{D}$ để chỉ tập huấn luyện của các *random walks*, được tạo ra bằng cách lấy mẫu các *random walks* bắt đầu từ mỗi nút. Chúng ta có thể giả định rằng N cặp nút cùng xuất hiện cho mỗi nút u được lấy mẫu từ phân phối $(u, v) \sim p_{\mathcal{G},\mathcal{T}}(v|u)$. Trên thực tế, việc tính toán hàm mất mát trên sẽ gây tiêu tốn về mặt chi phí. Chính vì vậy mà một số chiến lược đã được đề xuất để giải quyết vấn đề trên. *DeepWalk* sử dụng *hierarchical softmax* để xấp xỉ (Phương trình 1.10), điều này bao gồm việc tận dụng cấu trúc cây nhị phân để tăng tốc quá trình tính toán. Mặt khác, *node2vec* sử dụng phương pháp *noise contrastive estimation* để xấp xỉ (Phương trình 1.11), trong đó yếu tố chuẩn hóa được xấp xỉ bằng cách sử dụng các mẫu âm (negative samples) theo cách sau:

$$\ell = \sum_{(u,v) \in \mathcal{D}} -\log(\sigma(z_u^T z_v)) - \gamma \mathbb{E}_{v_n \sim P_n(\mathcal{V})}[\log(-\sigma(z_u^T z_{v_n}))] \tag{1.12}$$

với σ được hiểu là hàm *Logistic*, $P_n(\mathcal{V})$ để chỉ phân phối trên tập nút $\mathcal{V}$ và giả sử rằng $\gamma > 0$ là một siêu tham số. Thực tế, $P_n(\mathcal{V})$ thường được định nghĩa là một phân phối đều, và kỳ vọng được xấp xỉ bằng cách sử dụng lấy mẫu Monte Carlo.

Nhúng mạng thông tin quy mô lớn (LINE)

Ngoài DeepWalk và node2vec, thuật toán **LINE** cũng thường được thảo luận trong bối cảnh các phương pháp tiếp cận dựa trên random-walk. Mặc

dù phương pháp **LINE** không trực tiếp tận dụng *random walks*, nhưng nó chia sẻ những động lực khái niệm tương tự với DeepWalk và node2vec. Ý tưởng cơ bản trong LINE là kết hợp hai mục tiêu mã hóa-giải mã. Mục tiêu đầu tiên nhắm tới việc mã hóa thông tin liền kề bậc nhất *(first-order adjacency)* và sử dụng bộ giải mã sau đây:

$$\text{DEC}(z_u, z_v) = \frac{1}{1 + e^{-z_u^T z_v}} \tag{1.13}$$

với một thước đo tương đồng dựa trên ma trận liền k (tức là, $S[u, v] = A[u, v]$). Mục tiêu thứ hai tương tự hơn với các phương pháp tiếp cận dựa trên random walk. Nó sử dụng cùng bộ giải mã như trong (Phương trình 1.10), nhưng được huấn luyện bằng cách sử dụng *KL-divergence* để mã hóa thông tin liền kề hai bước (tức là thông tin trong A^2). Vì vậy, **LINE** có mối liên hệ khái niệm với node2vec và DeepWalk. Phương pháp này sử dụng một bộ giải mã xác suất và hàm mất mát xác suất (dựa trên *KL-divergence*). Tuy nhiên, thay vì lấy mẫu *random walk*, nó tái kiến trúc trực tiếp thông tin của hàng xóm bậc nhất và bậc hai.

1.3 Mô hình Mạng nơ-ron đồ thị (Graph Neural Networks-GNNs)

Trong chương này, tác giả sẽ giới thiệu về cách xây dựng mô hình mạng nơ-ron đồ thị [1], [6]. Trước khi mô hình GNNs ra đời, nhiều thách thức đã đặt ra khi các mô hình mạng thần kinh trước đó đang chỉ phù hợp với cấu trúc dữ liệu cụ thể của nó. Với *Mạng tích chập (CNNs)* chỉ được định nghĩa tốt trên đầu vào có cấu trúc lưới (như hình ảnh), hay *Mạng nơ-ron hồi quy (RNNs)* chỉ được định nghĩa tốt trên các chuỗi (như văn bản). Để định nghĩa một mạng nơ-ron sâu trên đồ thị tổng quát, chúng ta cần thiết kế một loại kiến trúc học sâu mới.

1.3.1 Truyền thông điệp Nơ-ron (Neural Message Passing)

Tổng quan về Khung Lan Truyền Thông Điệp

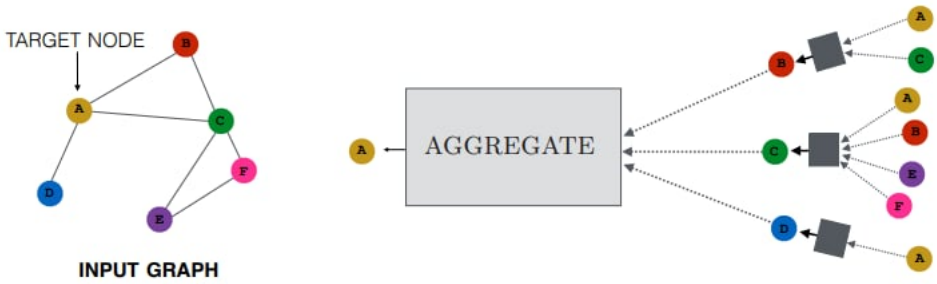
Với mỗi lần lặp lan truyền thông điệp trong một GNN, một biểu diễn ẩn $h_u^{(k)}$ tương ứng với mỗi nút $u \in \mathcal{V}$ được cập nhật dựa trên thông tin được

tổng hợp từ vùng lân cận $\mathcal{N}(u)$ của u. Việc cập nhật lan truyền thông điệp này có thể được biểu diễn như sau:

$$h_u^{(k+1)} = \text{UPDATE}^{(k)}\big(h_u^{(k)}, \text{AGGREGATE}^{(k)}(\{h_v^{(k)}, \forall v \in \mathcal{N}(u)\})\big) \quad (1.14)$$

$$= \text{UPDATE}^{(k)}\big(h_u^{(k)}, m_{\mathcal{N}(u)}^{(k)}\big) \quad (1.15)$$

Trong đó, **UPDATE** và **AGGREGATE** là các hàm khả vi tùy ý và $m_{\mathcal{N}(u)}$ là "thông điệp"được tổng hợp từ vùng lân cận $\mathcal{N}(u)$ của nút u. Chúng ta sử dụng chỉ số mũ để phân biệt các biểu diễn và hàm tại các bước lặp khác nhau của quá trình lan truyền thông điệp.



Hình 1.6: Tổng quan về cách một nút tổng hợp các tin nhắn từ các hàng xóm lân cận

Trong mỗi bước lặp k của GNN, hàm **AGGREGATE** nhận vào tập hợp các biểu diễn của các nút trong vùng lân cận $\mathcal{N}(u)$ của nút u và tạo ra một thông điệp $m_{\mathcal{N}(k)}(u)$ dựa trên thông tin tổng hợp từ vùng lân cận này. Sau đó, hàm UPDATE kết hợp thông điệp $m_{\mathcal{N}(k)}(u)$ với biểu diễn trước đó $h_u^{(k-1)}$ của nút u để tạo cập nhật $h_u^{(k)}$. Các biểu diễn ban đầu tại $k = 0$ được thiết lập bằng các đặc trưng đầu vào cho tất cả các nút, tức là $h_u^{(0)} = x_u, \forall u \in \mathcal{V}$. Sau khi thực hiện K bước lặp của quá trình lan truyền thông điệp GNN, chúng ta có thể sử dụng đầu ra của lớp cuối cùng để xác định các biểu diễn cho mỗi nút, tức là:

$$z_u = h_u^{(K)}, \forall u \in \mathcal{V} \quad (1.16)$$

Khác với các phương pháp nhúng nông, GNNs yêu cầu chúng ta phải có các đặc trưng nút $x_u, \forall u \in \mathcal{V}$ làm đầu vào cho mô hình. Trong nhiều đồ thị, chúng ta sẽ có các đặc trưng nút phong phú để sử dụng (ví dụ, các đặc trưng biểu hiện gen trong các mạng sinh học hoặc các đặc trưng văn

bản trong các mạng xã hội). Tuy nhiên, trong những trường hợp không có đặc trưng nút, vẫn có một số lựa chọn. Một trong số đó là sử dụng thống kê hay một cách phổ biến khác là sử dụng đặc trưng nhận dạng, trong đó chúng ta gán mỗi nút với một đặc trưng chỉ báo one-hot, đặc trưng này giúp xác định duy nhất nút đó.

Mạng nơ-ron đồ thị cơ bản (The basic GNN)

Để chuyển khung GNN trừu tượng được định nghĩa trong (Phương trình 1.14) thành một thứ gì đó có thể thực hiện được, chúng ta phải cung cấp các triển khai cụ thể cho các hàm **UPDATE** và **AGGREGATE.** Chúng ta bắt đầu ở đây với khung GNN cơ bản nhất, là sự đơn giản hóa của các mô hình GNN gốc được đề xuất bởi Merkwirth và Lengauer [2005] và Scarselli et al [2009].

Việc GNN truyền thông điệp cơ bản sẽ được định nghĩa như sau:

$$h_u^{(k)} = \sigma\left(W_{self}^{(k)} h_u^{(k-1)} + W_{neigh}^{(k)} \sum_{v \in \mathcal{N}(u)} h_v^{(k-1)} + b^{(k)}\right) \tag{1.17}$$

trong đó, $W_{self}^{(k)}, W_{neigh}^{(k)} \in \mathbb{R}^{d^{(k)} \times d^{(k-1)}}$ là các ma trận tham số có thể huấn luyện và σ biểu diễn một hàm phi tuyến áp dụng từng phần tử (ví dụ: hàm tanh hoặc ReLU). Thành phần bù *(bias)* $b^{(k)} \in \mathbb{R}^{d^{(k)}}$ thường được bỏ qua để đơn giản hóa ký hiệu, nhưng việc bao gồm thành phần bù có thể quan trọng để đạt hiệu suất cao.

Quá trình truyền thông điệp *(message passing)* trong khung cơ bản của GNN tương tự như một mạng nơ-ron truyền thẳng nhiều lớp (MLP) hoặc mạng nơ-ron hồi quy vì nó dựa trên các phép toán tuyến tính tiếp theo là một hàm phi tuyến áp dụng từng phần tử. Đầu tiên, chúng ta tổng hợp các thông điệp nhận được từ các nút láng giềng, sau đó, chúng ta kết hợp thông tin từ láng giềng với biểu diễn trước đó của nút thông qua một tổ hợp tuyến tính, và cuối cùng, áp dụng một hàm phi tuyến từng phần tử.

Chúng ta có thể định nghĩa tương đương GNN cơ bản thông qua các hàm **UPDATE** và **AGGREGATE**:

$$m_{\mathcal{N}(u)} = \sum_{v \in \mathcal{N}(u)} h_v \tag{1.18}$$

$$\text{UPDATE}(h_u, m_{\mathcal{N}(u)}) = \sigma(W_{self} h_u + W_{neigh} m_{\mathcal{N}(u)}) \tag{1.19}$$

và, thông điệp gửi từ nút u sẽ được sử dụng dưới dạng:

$$m_{\mathcal{N}(u)} = \text{AGGREGATE}^{(k)}(\{h_v^{(k)}, \forall v \in \mathcal{N}(u)\}) \tag{1.20}$$

Truyền thông điệp với các vòng tự lặp (Self-loops)

Như một sự đơn giản hóa của phương pháp truyền thông điệp thần kinh, việc thêm các *self-loop* vào đồ thị đầu vào và bỏ qua bước cập nhật rõ ràng là điều phổ biến. Trong phương pháp này, chúng ta định nghĩa việc truyền thông điệp một cách đơn giản như sau:

$$h_u^{(k)} = \text{AGGREGATE}(\{h_v^{(k-1)}, \forall v \in \mathcal{N}(u) \cup \{u\}\}) \tag{1.21}$$

Phép tổng hợp được thực hiện trên tập hợp $\mathcal{N}(u) \cup \{u\}$ tức là các nút lân cận của nút cũng như chính nút đó. Lợi ích của phương pháp này là chúng ta không cần phải định nghĩa một hàm cập nhật rõ ràng, vì việc cập nhật được định nghĩa ngầm thông qua phương pháp tổng hợp. Việc đơn giản hóa quá trình truyền thông điệp theo cách này có thể giúp giảm thiểu hiện tượng overfitting, nhưng nó cũng hạn chế đáng kể khả năng biểu đạt của GNN, vì thông tin từ các nút lân cận không thể được phân biệt với thông tin từ chính nút đó.

Trong trường hợp của GNN cơ bản, việc thêm các *self-loop* tương đương với việc chia sẻ các tham số giữa ma trận W_{self}, W_{neigh}, điều này cho phép cập nhật cấp đồ thị như sau:

$$H^{(t)} = \sigma\Big((A + I)H^{(t-1)}W^{(t)}\Big) \tag{1.22}$$

1.3.2 Phương pháp tổng hợp nút lân cận (Neighborhood Aggregation Methods)

Sự chuẩn hóa lân cận

Phép tổng hợp khu vực các nút láng giềng cơ bản nhất đơn giản là chỉ lấy tổng các *embedding* của các nút láng giềng. Một vấn đề với cách tiếp cận này là nó có thể không ổn định và cực kỳ nhạy cảm với bậc của các nút. giả sử nút u có số láng giềng gấp 100 lần nút u' (tức là bậc của u cao hơn nhiều so với bậc của u'), khi đó có thể mong đợi rằng $\| \sum_{v \in \mathcal{N}(u)} h_v >> sum_{v' \in \mathcal{N}(u')} h'_v \|$ (đối với bất kỳ chuẩn vector hợp lý nào $\|.\|_k$). Sự khác biệt rõ rệt về độ lớn này có thể dẫn đến các bất ổn về số học cũng như gặp khó khăn trong quá trình tối ưu hóa.

Một giải pháp cho vấn đề này là đơn giản hóa bằng cách chuẩn hóa phép tổng hợp dựa trên bậc của các nút liên quan. Cách tiếp cận đơn giản nhất là lấy trung bình thay vì tổng:

$$m_{\mathcal{N}(u)} = \frac{\sum_{v \in \mathcal{N}(u)} h_v}{|\mathcal{N}(u)|} \tag{1.23}$$

Tuy nhiên, các nhà nghiên cứu cũng đã đạt được thành công với các yếu tố chuẩn hóa khác, chẳng hạn như phép chuẩn hóa đối xứng *(symmetric normalization)* sau đây được Kipf và Welling sử dụng:

$$m_{\mathcal{N}(u)} = \sum_{v \in \mathcal{N}(u)} \frac{h_v}{\sqrt{|\mathcal{N}(u)||\mathcal{N}(v)|}} \tag{1.24}$$

Chuẩn hóa đối xứng cũng có thể được thúc đẩy dựa trên lý thuyết phổ đồ thị. Cụ thể, việc kết hợp gộp chuẩn hóa đối xứng (Phương trình 1.24) cùng với hàm cập nhật GNN cơ bản dẫn đến một xấp xỉ bậc nhất của một tích chập đồ thị phổ.

Graph convolutional networks (GCNs)

Một trong những mô hình mạng nơ-ron đồ thị cơ bản phổ biến nhất là mạng tích chập đồ thị **(GCN)** - sử dụng phương pháp tổng hợp chuẩn hóa đối xứng *(symmetric-normalized aggregation)* cũng như cách tiếp cận *self-loop update*. Từ đó, mô hình GCN định nghĩa hàm truyền thông điệp như sau:

$$h_u^{(k)} = \sigma\left(W^{(k)} \sum_{v \in \mathcal{N}(u) \cup \{u\}} \frac{h_v}{\sqrt{|\mathcal{N}(u)||\mathcal{N}(v)|}} \right) \tag{1.25}$$

Cách tiếp cận này lần đầu tiên được đề xuất và đã chứng minh là một trong những kiến trúc GNN cơ bản phổ biến và hiệu quả nhất. Một câu hỏi đặt ra là có nên chuẩn hóa hay không? Điều này phụ thuộc vào bài toán và ứng dụng cụ thể. Thông thường, chuẩn hóa hữu ích nhất trong các bài toán mà thông tin đặc trưng của nút quan trọng hơn nhiều so với thông tin cấu trúc, hoặc khi có sự chênh lệch lớn về bậc của các nút, có thể gây ra sự bất ổn định trong quá trình tối ưu hóa.

Cơ chế chú ý (Attention mechanisms)

Ngoài các dạng tổng hợp *(aggregation)* tổng quát, một chiến lược phổ biến để cải thiện lớp tổng hợp trong GNNs là áp dụng cơ chế *attention*. Ý tưởng cơ bản là gán một trọng số *attention* hoặc mức độ quan trọng cho từng lân cận, được sử dụng để cân nhắc ảnh hưởng của lân cận đó trong bước tổng hợp. Mô hình GNN đầu tiên áp dụng kiểu attention này là Graph Attention Network (GAT), sử dụng trọng số *attention* để định nghĩa một tổng có trọng số của các lân cận:

$$m_{\mathcal{N}(u)} = \sum_{v \in \mathcal{N}(u)} \alpha_{u,v} h_v$$

trong đó, a_u, v biểu thị mức độ *attention* đối với lân cận $v \in \mathcal{N}(u)$ khi chúng ta tổng hợp thông tin tại nút u. Trên cấu trúc gốc của GAT, trọng số chú ý *(attention weight)* được định nghĩa như sau:

$$\alpha_{u,v} = \frac{\exp\big(a^T[Wh_u \oplus Wh_v]\big)}{\sum_{v' \in \mathcal{N}(u)} \exp\big(a^T[Wh_u \oplus Wh_{v'}]\big)} \tag{1.26}$$

với a là một *vector attention* có thể huấn luyện được và W là một ma trận có thể huấn luyện. Việc tính toán *attention* theo phong cách GAT được biết là hoạt động hiệu quả với dữ liệu đồ thị. Tuy nhiên, về nguyên tắc, bất kỳ mô hình *attention* tiêu chuẩn nào trong lĩnh vực học sâu cũng có thể được sử dụng. Các biến thể *attention* phổ biến bao gồm mô hình attention song tuyến tính *(bilinear attention)*.

$$\alpha_{u,v} = \frac{\exp\big(h_u^T W h_v\big)}{\sum_{v' \in \mathcal{N}(u)} \exp\big(h_u^T W h_{v'}\big)} \tag{1.27}$$

Cũng như các biến thể của lớp attention sử dụng MLPs, như:

$$\alpha_{u,v} = \frac{\exp\big(\text{MLP}(h_u, h_v)\big)}{\sum_{v' \in \mathcal{N}(u)} \exp\big(\text{MLP}(h_u, h_{v'})\big)} \tag{1.28}$$

trong đó, MLP bị giới hạn chỉ có một đầu ra vô hướng (scalar). Việc thêm attention là một chiến lược hữu ích để tăng cường khả năng biểu diễn của mô hình GNN, đặc biệt là trong các trường hợp khi chúng ta có cơ sở để chỉ ra rằng một số lân cận có thể cung cấp thông tin quan trọng hơn những lân cận khác.

1.3.3 Phương pháp cập nhật tổng quát (Update Methods)

Quá trình truyền thông điệp trong GNN bao gồm hai bước chính: tổng hợp *(aggregation)* và cập nhật *(updating)*, và ở nhiều khía cạnh, toán tử **UPDATE** đóng vai trò quan trọng không kém trong việc xác định sức mạnh và thiên kiến quy nạp của mô hình GNN.

Vấn đề quá mượt *(over-smoothing)* và ảnh hưởng lân cận: một vấn đề phổ biến với GNN. Một vấn đề phổ biến với GNN mà các phương pháp cập nhật tổng quát hóa có thể giúp giải quyết được gọi là quá mượt *(over-smoothing)*. Ý tưởng chính của quá mượt là sau một vài lần lặp quá trình truyền thông điệp của GNN, các biểu diễn *(representations)* của tất cả các nút trong đồ thị có thể trở nên rất giống nhau. Xu hướng này đặc biệt phổ biến trong các mô hình GNN cơ bản và các mô hình sử dụng phương pháp cập nhật *(self-loop)*. Quá mượt là một vấn đề vì nó khiến việc xây dựng các mô hình GNN sâu – vốn dựa vào các phụ thuộc dài hạn trong đồ thị trở nên bất khả thi, do các mô hình GNN sâu này thường chỉ tạo ra các *embedding* bị quá mượt. Vấn đề quá mượt trong GNN có thể được biểu diễn một cách hình thức bằng cách định nghĩa ảnh hưởng của đặc trưng đầu vào $h_u^{(0)} = x_u$ của mỗi nút u lên embedding của lớp cuối cùng của tất cả các nút khác trong đồ thị, tức là $h_v^{(K)} \forall v \in \mathcal{V}$. Với mỗi cặp nút (u, v) ta có thể định lượng mức độ ảnh hưởng của nút u lên nút v trong GNN thông qua ma trận *Jacobi* tương ứng:

$$I_k(u, v) = 1^T \left(\frac{\partial h_v^{(K)}}{\partial h_u^{(0)}} \right) 1$$

trong đó, 1 là một vecto chứa toàn giá trị bằng 1. Giá trị $I_k(u, v)$ sử dụng tổng các giá trị trong ma trận jacobi $\frac{\partial h_v^{(K)}}{\partial h_u^{(0)}}$ làm thước đo của embedding ban đầu của nút u ảnh hưởng đến embedding cuối cùng của nút v.

Nối tiếp và bỏ qua kết nối

Các phương pháp nối tiếp *(concatenation)* và bỏ qua kết nối *(skip-connection)* có thể được sử dụng cùng với hầu hết các phương pháp cập nhật GNN khác. Do đó, để đảm bảo tính tổng quát, chúng ta sẽ sử dụng $\textbf{UPDATE}_{\text{base}}$ để biểu thị hàm cập nhật cơ bản mà chúng ta xây dựng, ta có thể giả định rằng hàm $\textbf{UPDATE}_{\text{base}}$ được biểu diễn bởi (Phương trình 1.19) và sau đó sẽ định nghĩa các phương pháp cập nhật kết nối đi kèm. Một trong những cách đơn giản nhất để thực hiện cập nhật kết nối là sử dụng phép nối *(concatenation)* nhằm bảo toàn nhiều thông tin ở cấp độ nút hơn trong quá trình truyền thông điệp:

$$\text{UPDATE}_{\text{concat}}(h_u, m_{\mathcal{N}(u)}) = [\text{UPDATE}_{\text{base}}(h_u, m_{\mathcal{N}(u)}) \oplus h_u]$$

ở đây, chúng ta đơn giản chỉ nối đầu ra của hàm cập nhật cơ bản với biểu diễn của lớp trước của nút. Ưu tiên mô hình phân tách thông tin trong quá trình truyền tin, tách biệt thông tin đến từ các nút lân cận (tức là $m_{\mathcal{N}(u)}$) với biểu diễn hiện tại của từng nút (tức là h_u). Tuy nhiên, ngoài phép nối, chúng ta cũng có thể áp dụng các dạng kết nối bỏ qua khác, chẳng hạn như phương pháp nội suy tuyến tính:

$$\text{UPDATE}_{\text{interpolate}}(h_u, m_{\mathcal{N}(u)}) = \alpha_1 \circ \text{UPDATE}_{\text{base}}(h_u, m_{\mathcal{N}(u)}) + \alpha_2 \odot h_u$$

trong đó, $\alpha_1, \alpha_2 \in [0, 1]^d$ là các vector với $\alpha_2 = 1 - \alpha_1$ và $\circ$ biểu diễn phép nhân từng phần tử. Trong phương pháp này, biểu diễn cuối cùng được cập nhật là một phép nội suy tuyến tính giữa biểu diễn trước đó và biểu diễn đã được cập nhật dựa trên thông tin từ các nút lân cận. Các tham số từ α_1 có thể được học cùng với mô hình theo nhiều cách khác nhau, có thể áp

dụng các phương pháp đơn giản, chẳng hạn như học trực tiếp các tham số α_1 cho từng lớp truyền tin hoặc sử dụng một MLP trên các biểu diễn nút hiện tại để tạo ra các tham số này.

Gated Updates

Bên cạnh cách tiếp cận từ viêc bỏ qua các kết nối từ nút như trên thì còn một cách nhìn nhận thuật toán truyền tin của GNN là hàm tổng hợp nhận một quan sát từ các nút lân cận, sau đó được sử dụng để cập nhật trạng thái ẩn của từng nút. Theo cách nhìn này, chúng ta có thể áp dụng trực tiếp các phương pháp được sử dụng để cập nhật trạng thái ẩn của các kiến trúc RNN dựa trên quan sát. Một trong những kiến trúc GNN sớm nhất định nghĩa hàm cập nhật như sau:

$$h_u^{(k)} = \text{GRU}(h_u^{(k-1)}, m_{\mathcal{N}(u)}^{(k)}) \tag{1.29}$$

với GRU biểu thị phương trình cập nhật của tế bào đơn vị hồi tiếp có cổng (GRU). Các phương pháp khác đã sử dụng cập nhật dựa trên kiến trúc LSTM. bất kỳ hàm cập nhật nào được định nghĩa cho RNN đều có thể được sử dụng trong ngữ cảnh của GNN. Chúng ta chỉ cần thay thế đối số trạng thái ẩn của hàm cập nhật RNN (thường được ký hiệu là $h^{(t)}$ bằng trạng thái ẩn của nút và thay thế vector quan sát (thường được ký hiệu là $x^{(t)}$) bằng thông điệp được tổng hợp từ vùng lân cận cục bộ. Trên thực tế, các nhà nghiên cứu thường chia sẻ các tham số của hàm cập nhật qua các tầng truyền tin của GNN.

1.3.4 GNNs cho phân loại nút (Node Classification)

Phân loại nút là một trong những nhiệm vụ chuẩn phổ biến nhất cho GNNs. Khi các phương pháp GNN bắt đầu trở nên nổi bật trong lĩnh vực học máy-nghiên cứu về GNNs chủ yếu xoay quanh các bộ dữ liệu chuẩn của mạng trích dẫn Cora, Citeseer. Các bộ dữ liệu này yêu cầu phân loại danh mục hoặc chủ đề của các bài báo khoa học dựa trên vị trí của chúng trong mạng trích dẫn, với các đặc trưng nút dựa trên ngôn ngữ (ví dụ: vector từ) và chỉ một số lượng rất nhỏ các ví dụ dương tính *(positive)* được cung cấp cho mỗi lớp (thường ít hơn 10% số lượng nút).

Cách tiêu chuẩn để áp dụng GNNs vào nhiệm vụ phân loại nút như vậy là huấn luyện GNNs theo phương pháp có giám sát hoàn toàn, trong đó chúng ta định nghĩa hàm mất mát sử dụng hàm phân loại *softmax* và hàm mất mát *negative log-likehood*:

$$\mathcal{L} = \sum_{u \in \mathcal{V}_{\text{train}}} -\log(\text{softmax}(z_u, y_u)) \tag{1.30}$$

Chúng ta sẽ giả định rằng $y_u \in \mathbb{Z}^c$ là một vector *one-hot* chỉ ra lớp của nút huấn luyện $u \in \mathcal{V}_{\text{train}}$. Chúng ta sử dụng $\text{softmax}(z_u, y_u)$ để biểu thị xác suất dự đoán rằng nút thuộc về lớp y_u, được tính thông qua hàm *softmax* như sau:

$$\text{softmax}(z_u, y_u) = \sum_{i=1}^{c} y_u[i] \frac{e^{z_u^T w_i}}{\sum_{j=1}^{c} e^{z_u^T w_j}} \tag{1.31}$$

trong đó, $w_i \in \mathbb{R}^d$, với $i = 1, ..., c$ là các tham số có thể học. Có các biến thể khác của các hàm mất mát có giám sát cho phân loại nút, nhưng việc huấn luyện GNNs theo cách có giám sát dựa trên hàm mất mát trong (Phương trình 1.30) là một trong những chiến lược tối ưu phổ biến nhất cho GNNs.

Chương 2

Ứng dụng nhóm mô hình Graph-based xây dựng bài toán dự đoán bệnh dựa trên triệu chứng

2.1 Phương pháp đề xuất tiếp cận xây dựng bài toán bài toán PDfS

Trong những năm gần đây, các mạng nơ-ron đồ thị (GCNs) đã được áp dụng trong nhiều ứng dụng khác nhau, đặc biệt là trong lĩnh vực y tế, ví dụ như phân tích não bộ [16], phân tích nhũ ảnh [5],...Vấn đề dự đoán bệnh cũng đã được nghiên cứu rộng rãi bằng các phương pháp dựa trên GCN [12], [19]. Theo những kết quả đầy hứa hẹn của GCN trong lĩnh vực y tế, [14] đã khai thác GCN trong vấn đề dự đoán bệnh với các tập dữ liệu đa phương thức. Họ đã chọn *ChebyNet* với kích thước bộ lọc cố định làm bộ phân loại và nghiên cứu cách xây dựng đồ thị quan hệ bằng cách tính khoảng cách giữa các đối tượng trong phân tích não bộ. *InceptionGCN* [2] mở rộng *ChebyNet*, thiết kế các bộ lọc với các kích thước nhân khác nhau và chọn bộ lọc tối ưu nhất cho việc dự đoán bệnh cuối cùng. Một nhánh các phương pháp tập trung vào cải thiện cấu trúc đồ thị. [20] và [17] bắt đầu với một đồ thị được xây dựng sẵn và cập nhật nó trong quá trình huấn luyện, nhưng [10] học toàn bộ đồ thị trực tiếp từ các đặc trưng theo cách *end-to-end*.

Trong chương này, tác giả đồ án sẽ trình bày cách tiếp cận để xây dựng

mô hình dựa trên các loại mô hình đồ thị khác nhau. Bao gồm cả đơn đồ thị đầu vào và đa đồ thị đầu vào. Ngoài ra, tác giả sẽ thực hiện kết hợp các cơ chế khác nhau để tổng hợp và cập nhật trong quá trình truyền thông điệp giữa các nút.

2.1.1 Định nghĩa bài toán

Thực hiện xem xét một đô thị $\mathcal{G}$ với $\mathcal{N}$ nút, được biểu diễn bởi $\mathcal{G}(\mathcal{V}, \mathcal{E}, \mathcal{X})$. Trong đó, $\mathcal{V}$ là tập các nút ($|\mathcal{V}| = \mathcal{N}$), $\mathcal{E}$ là tập các cạnh liên kết giữa các nút và $\mathcal{X} \in \mathbb{R}^{\mathcal{N}\times\mathcal{F}}$ là ma trận đặc trưng của các nút ($\mathcal{F}$ là tập các đặc trưng). Ma trận kề $A \in \mathbb{R}^{\mathcal{N}\times\mathcal{N}}$ là không có trọng số và không có hướng phản ánh các kết nối giữa các nút. Mỗi nút v_i có một đặc trưng liên quan là x_i, một vecto *one-hot* y_i và một nhãn lớp thực $c_i \in C$ (với C là tập hợp các lớp). Thông tin nhãn chỉ có sẵn cho một tập con của các nút, và mục tiêu là học một hàm tham số $f_\theta(\mathcal{X}, A)$ nhận ma trận kề và đặc trưng của các nút làm đầu vào, sau đó dự đoán nhãn thực cho các nút chưa được gán nhãn. Điều quan trọng cần lưu ý là một bộ phân loại xác suất sẽ tạo ra một phân phối xác suất trên $|C|$ lớp và nhãn có xác suất cao nhất sẽ được chọn. Trong phương pháp được đề xuất, q_i đại diện cho phân phối xác suất đầu ra của bộ phân loại, được định nghĩa trên $|C|$ lớp cho mẫu x_i, trong đó phần tử thứ c thể hiện độ tin cậy của bộ phân loại trong việc gán nhãn c cho x_i. Do đó, bài toán có thể được biểu diễn như sau:

$$Q = f_\theta(\mathcal{X}, A) \tag{2.1}$$

trong đó, $Q \in \mathbb{R}^{\mathcal{N}\times|C|}$ là ma trận dự đoán của bộ phân loại cho tất cả các nút, bao gồm cả những nút chưa được gán nhãn.

2.1.2 Phương pháp mã hóa đồ thị

Đầu tiên, trong ngữ cảnh của một đồ thị $\mathcal{G}$, tác giả đồ án sẽ sử dụng một biểu diễn đồng nhất cho các nút bệnh, triệu chứng hoặc bệnh nhân là $v \in \mathcal{G}$ để đơn giản hóa quá trình. Tại tầng lan truyền thông điệp thứ l, embedding $h_v^{(l)}$ của nút v được tính như sau:

$$h_{\mathcal{N}(v)}^{(l)} = \text{AGGREGATE}\Big(\{h_{v'}^{(l-1)}, \forall v' \in \mathcal{N}(v)\}\Big) \tag{2.2}$$

$$h_v^{(l)} = \sigma\Big(W^{(l)}.[h_v^{(l-1)} \| h_{\mathcal{N}(v)}^{(l)}]\Big) \tag{2.3}$$

Ở đây, $W^{(l)}$ biểu thị ma trận trọng số cần học tại tầng thứ l, $h_v^{(l-1)}$ đại diện cho embedding của nút v tại tầng trước đó và L biểu thị tổng số tầng. Ký hiệu $[.\|.]$ dùng để chỉ phép nối hai vector và $\mathcal{N}(v)$ đại diện cho tập hợp các nút lân cận của v được lấy mẫu ngẫu nhiên. Một điều quan trọng cần lưu ý, đó là với $l = 0$, embedding nút $h_v^{(0)} \in \mathbb{R}^d$ được khởi tạo bằng các giá trị ngẫu nhiên hoặc bằng các thông tin phụ có từ dữ liệu, tùy thuộc vào sự có sẵn của nó. Ví dụ trong trường hợp nút bệnh nhân, nếu thông tin nhân khẩu và hồ sơ y tế của bệnh nhân có trong dữ liệu hồ sơ y tế điện tử (EMR), $h_v^{(0)}$ sẽ được khởi tạo dưới dạng một vector đặc trưng thực, dày đặc. Mỗi phần tử trong $h_v^{(0)}$ sẽ biểu diễn một giá trị quan sát được của một chiều đặc trưng cụ thể (ví dụ: tuổi). $h_{\mathcal{N}(v)}^{(l)}$ đại diện cho biểu diễn kết hợp thu được thông qua hàm tổng hợp, được thiết kế để kết hợp các embedding của các nút láng giềng của v từ tầng $l-1$, σ biểu thị hàm kích hoạt phi tuyến như hàm hyperbolic tangent (tanh), và bộ tổng hợp có thể được chọn từ các tùy chọn như mean, max pooling, mạng nơ-ron hồi tiếp (RNN), và nhiều tùy chọn khác. Sau đó, chúng ta sẽ thực hiện bước chuẩn hóa trước khi đạt được embedding cuối cùng cho tất cả các nút tại tầng cuối cùng L:

$$h_v = \frac{h_v^{(L)}}{\|h_v^{(L)}\|_2}, \forall v \in \mathcal{G} \tag{2.4}$$

Sau khi đi qua các tầng GCN, chúng ta sẽ đưa h_v qua một tầng MLP để thu được biểu diễn embedding cuối cùng.

$$z_v = \sigma(W h_v), \forall v \in \mathcal{G} \tag{2.5}$$

trong đó, W là ma trận trọng số có thể học được và z_v là embedding cuối cùng của nút v.

2.1.3 Áp dụng cơ chế chú ý cho mã hóa đồ thị

Chúng ta có thể áp dụng cơ chế *attention* để mã hóa chọn lọc thông tin từ các nút láng giềng dựa trên mức độ quan trọng của chúng đối với nút mục tiêu v. Điều này được thực hiện bằng cách tính tổng có trọng số các biểu diễn của tất cả các nút láng giềng của nút v:

$$h_v^{(l)} = \sum_{v' \in \mathcal{N}(v)} \alpha_{v'v} M h_{v'}^{(l-1)} \tag{2.6}$$

trong đó, M là ma trận chuyển trọng số và $\alpha_{v'v}$ là trọng số attention biểu thị mức độ quan trọng của nút láng giềng $v' \in \mathcal{N}(v)$ khi tính toán $h_v^{(l)}$. Mỗi $\alpha_{v'v}$ được tính thông qua mạng *attention* sau:

$$\alpha_{v'v} = \frac{\exp\Big(\text{LeakyReLU}\Big(a^T \Big[N h_v^{(l-1)} \| N h_{v'}^{(l-1)}\Big]\Big)\Big)}{\sum_{k \in \mathcal{N}(v)} \exp\Big(\text{LeakyReLU}\Big(a^T \Big[N h_v^{(l-1)} \| N h_k^{(l-1)}\Big]\Big)\Big)} \tag{2.7}$$

Với một vector chiếu a và ma trận trọng số N. Về cơ bản, các trọng số attention được học giúp bộ tổng hợp tập trung nhiều hơn vào các nút láng giềng có đóng góp lớn hơn trong quá trình truyền thông điệp, từ đó tạo ra các embedding của nút có tính biểu đạt cao.

2.1.4 Dự đoán bệnh cho bệnh nhân

Bộ giải mã đồ thị trong mô hình đóng vai trò quan trọng trong việc chuyển đổi thông tin được mã hóa bởi các nút triệu chứng, bệnh, và bệnh nhân thành các dự đoán về các bệnh tiềm năng liên quan đến một bệnh nhân nhất định. Cụ thể, khi được cung cấp vector mã hóa z_p của một bệnh nhân p, bộ giải mã đồ thị chuyển đổi nó thành một đầu ra dạng vector hóa $\hat{c}_p \in \{0,1\}^{|C|}$, đóng vai trò như một xấp xỉ của nhãn đa giá trị $c_p \in \{0,1\}^{|C|}$ biểu thị bệnh của bệnh nhân. Trong quá trình giải mã, từng phần tử của $\hat{c}_p$ được tính toán như sau:

$$\hat{c}_{p,n} = \text{sigmoid}(z_p^T Q_n), \forall m_n \in \mathcal{V}_M \tag{2.8}$$

Trong đó, $\hat{c}_{p,n}$ là phần tử thứ n trong $\hat{c}_p$, với $n \leq |\mathcal{V}_M|$ được sử dụng để đánh chỉ mục tất cả các bệnh m. Giá trị $\hat{c}_{p,n}$ càng gần 1 thì bệnh nhân p

càng có khả năng được chẩn đoán là mắc bệnh m_n. Ma trận $Q \in \mathbb{R}^{|\mathcal{V}_M| \times d}$ chứa các trọng số hồi quy tương ứng cho tất cả các bệnh, và Q_n là cột thứ n của ma trận này. Để huấn luyện mô hình, chúng ta định lượng hàm lỗi dự đoán thông qua hàm mất mát *negative log likelihood* như sau:

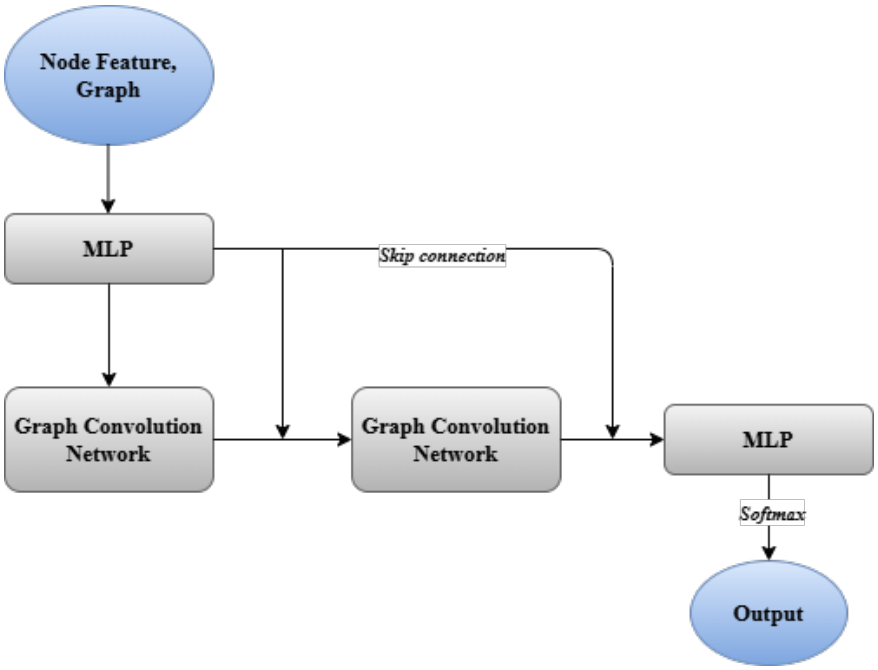
$$\mathcal{L} = -\sum_{n=1}^{|\mathcal{V}_M|} c_{p,n} \log(\hat{c}_{p,n}) \tag{2.9}$$

2.2 Mô hình đề xuất

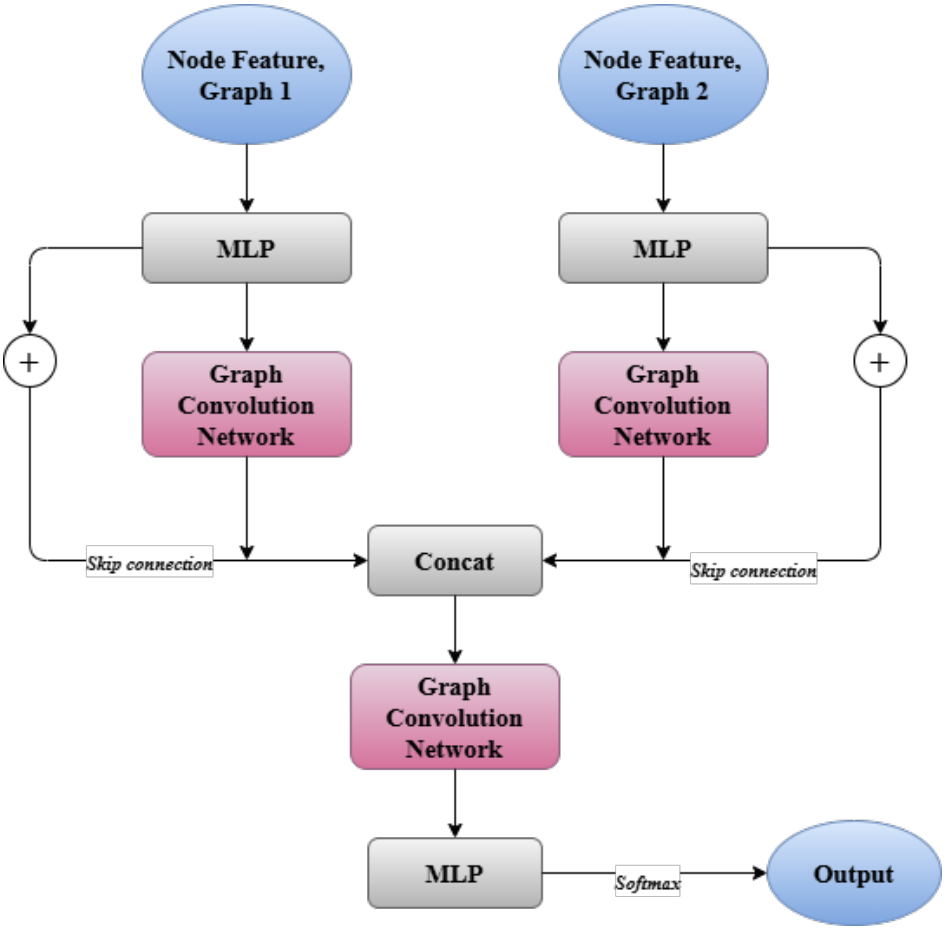
Tác giả đồ án sẽ xây dựng hai loại mô hình dựa trên đồ thị khác nhau, sử dụng một đồ thị đơn hoặc nhiều đồ thị làm đầu vào. Để tránh vấn đề *"over-smoothing"*, tác giả sẽ chỉ sử dụng hai lớp Mạng Tích chập Đồ thị (GCN) và tích hợp phương pháp bỏ qua kết nối *(skip connections)* để bảo toàn thông tin. Để tăng cường khả năng biểu diễn của mô hình, chúng ta sẽ kết hợp các lớp MLP, bao gồm các lớp *Fully Connected* và *Batch Normalization*, nhằm trích xuất thông tin một cách hiệu quả. Hơn nữa, trong quá trình huấn luyện sẽ áp dụng các cơ chế khác nhau để tổng hợp thông tin như SUM, MAX, MEAN và Attention, nhằm kết hợp hiệu quả thông tin từ các nút lân cận. Để cập nhật biểu diễn của các nút, chúng ta sẽ sử dụng các lớp MLP để học các đồ thị con đẳng cấu hoặc không đẳng cấu, đồng thời sử dụng mạng GRU để học biểu diễn tuần tự của các nút lân cận.

Phương pháp huấn luyện

Khi huấn luyện mô hình, tôi đưa ra giả thuyết: Việc giảm kích thước đồ thị đầu vào bằng cách lấy mẫu một tỷ lệ nhỏ các kết nối (5%, 10 20%, 30%, v.v.) sẽ không làm giảm hiệu suất của mô hình nếu thông tin được kết hợp bằng cơ chế *Attention*. Cơ chế *Attention* giúp mô hình học được các kết nối quan trọng trong đồ thị. Một nút sẽ học cách xác định các kết nối nào là quan trọng với các nút lân cận của nó. Vì vậy, việc loại bỏ các cạnh có xác suất thấp sẽ không ảnh hưởng đến hiệu suất. Các kết quả thực nghiệm trong phần tiếp theo sẽ minh chứng cho hiệu quả của cách tiếp cận huấn luyện này.



Hình 2.1: Mô hình đề xuất với đơn đồ thị đầu vào



Hình 2.2: Mô hình đề xuất với hai đồ thị đầu vào

Tạo đặc trưng cho đồ thị bệnh tật

Tôi đề xuất xây dựng các đặc trưng cho đồ thị bệnh dựa trên công thức TF-IDF thông qua các bước sau:

(1) Tính 3 triệu chứng phổ biến nhất cho mỗi bệnh.

(2) TF (Term Frequency): Đếm số lượng cá nhân mắc mỗi bệnh biểu hiện triệu chứng phổ biến tương ứng.

(3) IDF (Inverse Document Frequency): Đếm số lần xuất hiện của mỗi triệu chứng phổ biến trên toàn bộ tập dữ liệu.

(4) Tính trọng số bằng cách nhân TF với 1/TDF.

Như vậy, chúng ta sẽ có một ma trận đặc trưng cho đồ thị bệnh.

2.3 Chiến lược xây dựng đồ thị cho bài toán PDfS

Trong phần này, tác giả sẽ trình bày các chiến lược đề xuất để tạo đồ thị cho quá trình huấn luyện. Tác giả đồ án sẽ đề xuất bốn chiến lược xoay quanh các cơ chế tổng hợp và truyền thông điệp đã trình bày ở trên.

Chiến lược	Mô tả	Cơ chế tổng hợp
User-User-Disease	Tạo liên kết giữa 2 người có cùng bệnh	Mean +GRU
User-User-Symptom	Tạo liên kết giữa 2 người bệnh có cùng triệu chứng	Sum + Concat
User-User-Cosine (20%sample)	Xây dựng đồ thị dựa trên độ tương đồng giữa 2 người	Attention + GRU
MultiGraph-Input	Xây dựng đồ thị bệnh nhân và đồ thị bệnh tật	Mean + GRU

Bảng 2.1: Chiến lược xây dựng đồ thị

Về cơ bản, các đồ thị với *Node feature* sẽ được tạo ra thông qua trọng số mức độ nghiêm trọng của từng triệu chứng. Riêng với chiến lược xây dựng đa đồ thị đầu vào, *Node feature* của đồ thị bệnh tật sẽ xây dựng thông qua tính toán trọng số TF-IDF.

Đối với chiến lược *User-User-Cosine*, chỉ số Cosine similarity được tính như sau:

$$\text{Cosine similarity} = \cos(\theta) = \frac{A.B}{\|A\|\|B\|}$$

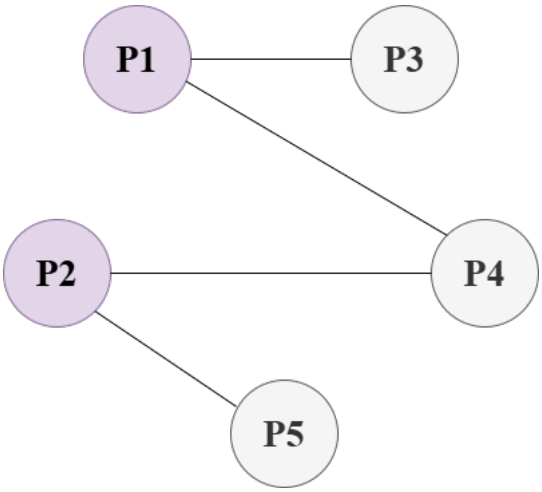
Cosine similarity chỉ tập trung so sánh hướng của vector, không quan tâm

đến độ lớn của chúng. Điều này hữu ích trong các bài toán khi ta chỉ quan tâm đến mô hình phân bố đặc trưng mà không cần quan tâm đến giá trị tuyệt đối.

Đối với chiến lược *MultiGraph-Input*, sẽ sử dụng 2 đồ thị làm đầu vào bao gồm *patient graph* và *Disease graph* các đồ thị sẽ được khởi tạo theo các dưới đây:

Patient Graph

Thực hiện tạo đồ thị giữa các bệnh, nếu giữa hai bệnh nhân có cùng một bệnh tật thì sẽ tạo liên kết (cạnh) trên đồ thị (đồ thị này tương đồng với đồ thị sử dụng ở chiến lược thứ nhất).

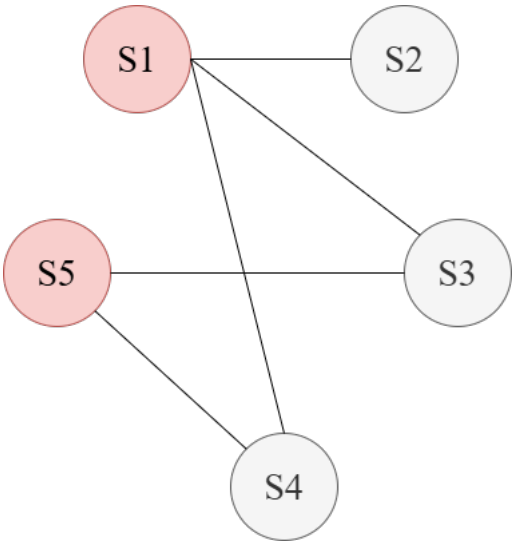


Hình 2.3: Đồ thị bệnh nhân

Disease Graph

Đồ thị bệnh tật sẽ được xây dựng dựa trên các bước sau:

1. Tính tổng số lần xuất của từng triệu chứng đối với mỗi loại bệnh
2. Lấy ra 3 triệu chứng phổ biến nhất của mỗi loại bệnh
3. Tạo liên kết (cạnh) giữa hai bệnh có cùng 2 triệu chứng phổ biến nhất.



Hình 2.4: Đồ thị bệnh tật

4. Xây dựng *Node feature* dựa trên trọng số TF-IDF, trong đó:

$$\text{TF}(s,d) = \frac{f_{s,d}}{\sum_{s' \in d} f_{s',d}}$$

- $f_{s,d}$: số lần xuất hiện của triệu chứng s của bệnh d
- $\sum_{s' \in d} f_{s',d}$: tổng số lần xuất hiện của các triệu chứng đối với bệnh d

$$\text{IDF}(s) = \log \frac{N}{1 + n_s}$$

- N là tổng số lần xuất hiện của triệu chứng trên toàn bộ bệnh.
- n_s là số bệnh chứa triệu chứng s

Khi đó công thức TF-IDF là:

$$\text{TF-IDF}(s,d) = \frac{f_{s,d}}{\sum_{s' \in d} f_{s',d}} \log \frac{N}{1 + n_s}$$

Chương 3

Cài đặt chương trình và trình bày kết quả thử nghiệm

3.1 Khảo sát dữ liệu và cài đặt mô hình

Trong báo cáo này, tác giả sử dụng hai bộ dữ liệu để dự đoán bệnh (bao gồm *Disease Symptom Prediction*[1] và *Pima Indian Diabetes* [8]) dựa trên các triệu chứng liên quan nhằm xây dựng các bài toán phân loại đa lớp và phân loại nhị phân.

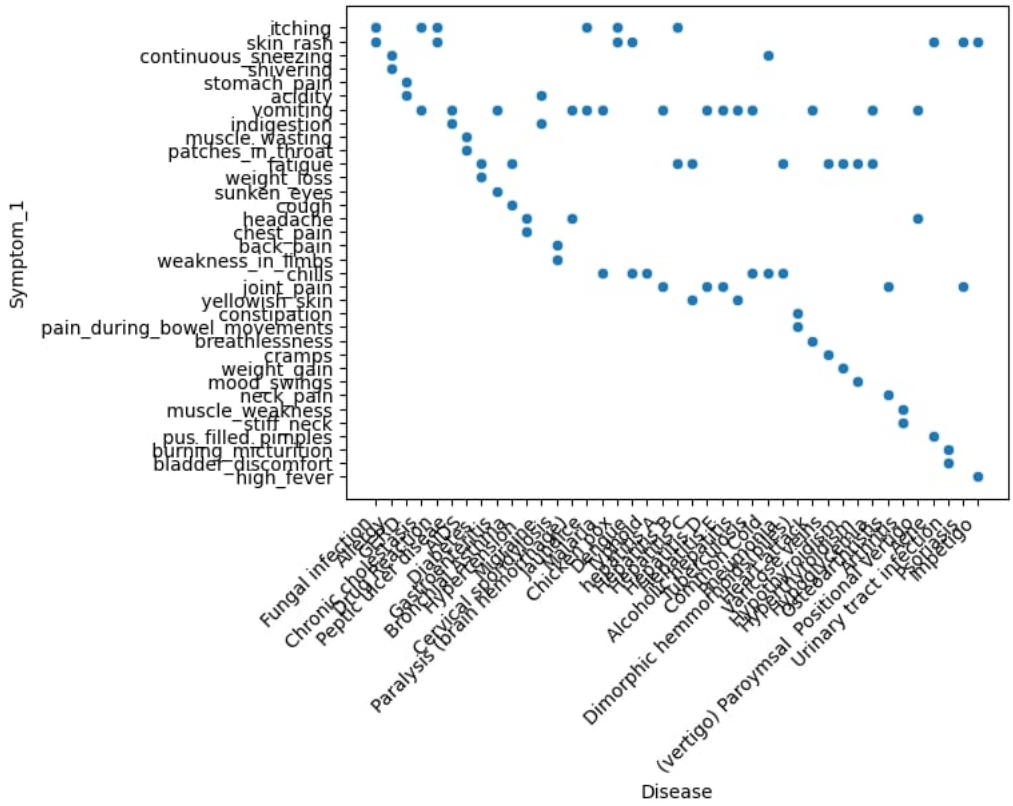
3.1.1 Khảo sát dữ liệu

Disease Symptom Prediction[1]

Bộ dữ liệu *Disease Symptom Prediction* này bao gồm 4920 bản ghi với 18 trường khác nhau là bệnh và đại diện cho các triệu chứng của bệnh. Có 41 loại bệnh riêng biệt, mỗi loại bệnh có 120 bản ghi. Sau khi thực hiện Phân tích Dữ liệu Khám phá (EDA), tôi nhận thấy rằng mỗi loại bệnh có 2-3 triệu chứng đặc trưng. Ngoài ra, cũng có các triệu chứng chung được chia sẻ giữa các loại bệnh khác nhau, chẳng hạn như đau đầu và mệt mỏi. Bên cạnh đó, các bệnh nhân mắc cùng một loại bệnh thường có 1-2 triệu chứng tương tự nhau. Những phát hiện này gợi ý ý tưởng xây dựng đồ thị cho các loại bệnh khác nhau và đồ thị bệnh nhân dựa trên các triệu chứng chung. Các đồ thị này nhằm khái quát hóa thông tin của bệnh nhân, và bằng cách sử dụng Mạng Nơ-ron Đồ thị (GNN), chúng ta có thể trích xuất

[1] https://www.kaggle.com/datasets/itachi9604/disease-symptom-description-dataset

các đặc trưng tiềm ẩn từ dữ liệu.



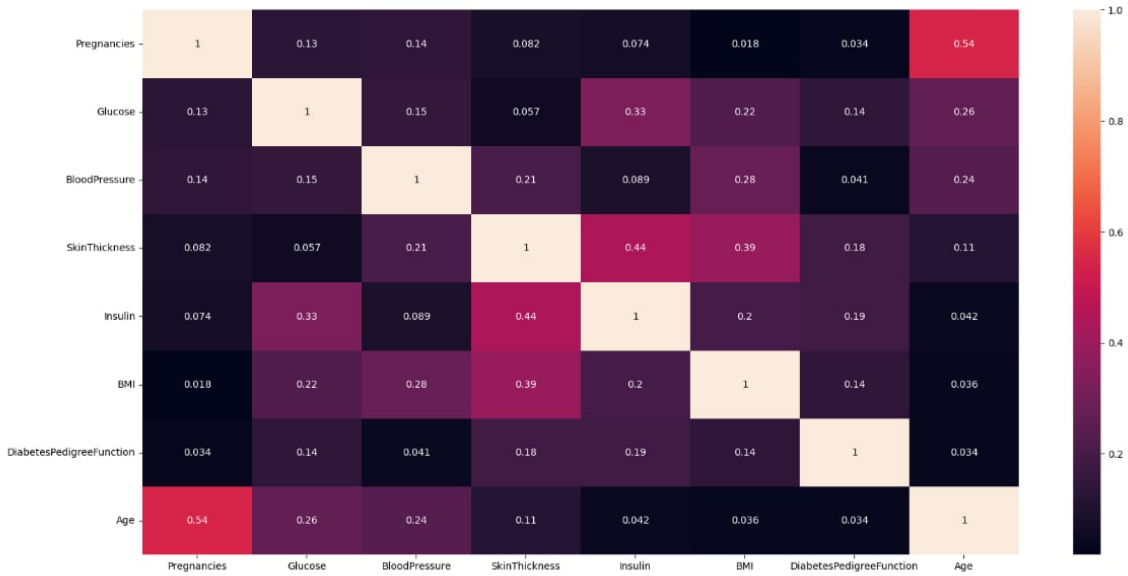
Hình 3.1: Mối quan hệ giữa bệnh tật và triệu chứng

Ngoài ra, đối với bộ dữ liệu này các triệu chứng còn có thêm trọng số chỉ mức độ nguy hiểm tương ứng. 133 triệu chứng được ghi nhận tương ứng với trọng số nguy hiểm trên thang đo từ 1 đến 7.

Pima Indian Diabetes [15]

Bộ dữ liệu này được đưa ra bởi *"National Institute of Diabetes and Digestive and Kidney Diseases"*. Mục tiêu của bộ dữ liệu là nhận diện tình trạng tiểu đường của bệnh nhân (bài toán phân loại nhị phân). Mỗi bệnh nhân có 7 đặc trưng dạng số, thể hiện các chỉ số chẩn đoán bệnh tiểu đường, bao gồm số lần mang thai, glucose huyết tương, huyết áp, độ dày da, chỉ số khối cơ thể (BMI), mức insulin, chức năng di truyền tiểu đường và tuổi. Để quan sát rõ hơn về mối liên hệ giữa các trường dữ liệu. Tôi thực hiện xây dựng ma trận tương quan để xem xét hạn chế hiện tượng *"oversmoothing"* - khi mối tương quan cao giữa các đặc trưng làm tăng khả năng các node chia sẻ thông tin quá mức, GNNs có thể rơi vào trạng

thái *"oversmoothing"*, nơi các node trở nên quá giống nhau và mất đi tính đặc trưng của nó.



Hình 3.2: Ma trận tương quan giữa các chỉ số bệnh PIMA Ấn Độ

Dựa trên ma trận tương quan, chúng ta có thể quan sát thấy mối tương quan tương đối cao giữa trường tuổi và các trường khác trong bộ dữ liệu. Do đó, sẽ thực hiện phân nhóm *(binning)* trên trường tuổi để tạo ra các nhóm tuổi khác nhau và sử dụng các nhóm này để xây dựng các kết nối đồ thị giữa các nhóm tuổi.

Nhóm	1	2	3	4	5
Tuổi	18-25	25-35	35-45	45-60	60-80

3.1.2 Cài đặt chương trình cho mô hình

Môi trường và phần cứng chạy chương trình

1. Cả code nguồn cho quá trình huấn luyện cũng như kiểm tra mô hình đều được thực hiện trên *Jupyter Notebook* của *Kaggle*.
2. Phần cứng bao gồm: CPU Ram 29GB, Disk space 73GB, GPU P100 memory 16GB, Output file max_size 19.5GB.
3. Ngôn ngữ lập trình: *Python*.
4. Mã nguồn *Disease Symptom Prediction* : *GNNs for Disease Symptom Prediction Dataset*

5. Mã nguồn *Pima Indian Diabetes*: *GNNs for Pima Indian Diabetes Dataset*

3.2 Đánh giá và so sánh kết quả kiểm thử

Trong báo cáo này, để đánh giá hiệu suất phân loại của mô hình, tôi tối ưu các tham số dựa trên các chỉ số F1-macro, Recall-macro và Precision-macro. Cách tiếp cận này được lựa chọn vì trong thực tế, có rất nhiều bệnh hiếm với rất ít bản ghi dữ liệu. Việc tối ưu tham số bằng chiến lược macro giúp giải quyết phần nào vấn đề mất cân bằng dữ liệu.

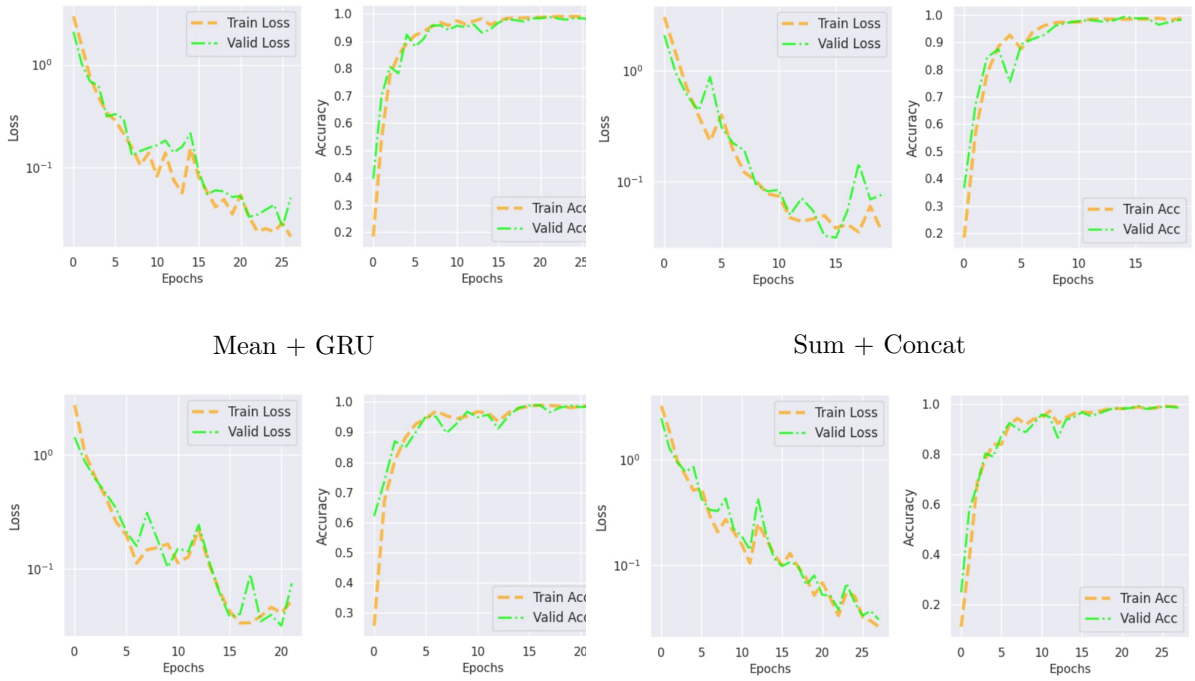
Multi-class Classification:

Bộ dữ liệu *'Disease Symptom Prediction'*. Sau khi tiến hành các thí nghiệm trên bộ dữ liệu trên, chúng tôi thu được các kết quả sau đây cho từng chiến lược xây dựng đồ thị và các phương pháp tổng hợp thông tin từ các *node* lân cận cũng như cập nhật biểu diễn của các *node*.

Model	Accuracy	F1-macro	Recall-macro	Precision-macro
Baseline XGBoost	0.90 ± 0.01	0.88 ± 0.01	0.89 ± 0.01	0.94 ± 0.01
Strategy 1 (Mean + GRU)	0.99 ± 0.01	0.99 ± 0.01	0.99 ± 0.01	0.99 ± 0.01
Strategy 2 (Sum + Concat)	0.98 ± 0.01	0.98 ± 0.01	0.99 ± 0.01	0.99 ± 0.01
Strategy 3 (Attention + GRU + sample 20%)	0.97 ± 0.01	0.96 ± 0.01	0.97 ± 0.01	0.97 ± 0.01
Strategy 4: Multi-Graph Input (Mean + GRU)	0.99 ± 0.01	0.99 ± 0.01	0.99 ± 0.01	0.99 ± 0.01

Bảng 3.1: Kết quả đánh giá mô hình cho phân loại nhiều lớp (Disease Symptom Prediction)

Chúng ta có thể quan sát rằng các mô hình dựa trên đồ thị đều cho kết quả vượt trội hơn so với mô hình cơ bản XGBoost đạt khoảng 0.90 *accuracy*, chứng minh hiệu quả của chúng trên tập dữ liệu này. Hơn nữa, phương pháp huấn luyện giảm kích thước đồ thị đầu vào và sử dụng lớp tổng hợp với cơ chế *Attention* giúp mô hình tập trung vào các kết nối quan trọng trong đồ thị, dẫn đến kết quả phân loại xuất sắc.



Hình 3.3: Xu hướng mất mát và độ chính xác của mô hình

Quan sát các biểu đồ, chúng ta có thể nhận ra được các mô hình hội tụ khá nhanh sau khoảng 15 epoch. Bên cạnh đó, có thể thấy rằng chiến lược thứ nhất (Mean + GRU) cho độ ổn định tốt hơn so với các chiến lược còn lại. Việc xây dựng nhiều loại đồ thị khác nhau đều cho kết quả cao phần nào cũng chứng tỏ tính mạnh mẽ của GNNs với nhiều cách tiếp cận.

Binary Classification:

Đối với bộ dữ liệu *'Pima Indian Diabetes'*, tôi đã tiến hành xây dựng mô hình phân loại nhị phân sử dụng tập dữ liệu trên. Để xây dựng đồ thị theo chiến lược thứ hai, tôi đã thực hiện chia tuổi thành các nhóm *(binning)* để tạo thành các nhóm tuổi và thiết lập các kết nối giữa các nhóm tuổi của bệnh nhân. Chúng ta có thể thấy rằng các mô hình dựa trên đồ thị vẫn rất hiệu quả. Tuy nhiên, đối với bộ dữ liệu này tôi sẽ không sử dụng chiến lược thứ tư. Ngoài ra, để mô hình tránh bị ảnh hưởng bởi yếu tố mất cân bằng dữ liệu tôi áp dụng *class weight* để tăng ảnh hưởng của các mẫu từ lớp thiểu số trong quá trình tính toán hàm mất mát, buộc mô hình phải chú ý hơn đến lớp này.

Model	Accuracy	F1-macro	Recall-macro	Precision-macro
Baseline KNN	0.75 ± 0.01	0.72 ± 0.01	0.71 ± 0.01	0.72 ± 0.01
Strategy 1 (Mean + GRU)	0.77 ± 0.01	0.71 ± 0.01	0.78 ± 0.01	0.70 ± 0.01
Strategy 2 (Sum + Concat)	0.78 ± 0.01	0.74 ± 0.01	0.78 ± 0.01	0.72 ± 0.01
Strategy 3 (Attention + GRU + sample 20%)	0.76 ± 0.01	0.74 ± 0.01	0.74 ± 0.01	0.73 ± 0.01

Bảng 3.2: Kết quả đánh giá mô hình cho phân loại nhị phân (Pima Indian Diabetes)

Dựa trên các kết quả phân tích và so sánh, mô hình GNNs đề xuất đã chứng minh tính hiệu quả cao trong việc giải quyết bài toán **PDfS**, đặc biệt nhờ khả năng khai thác thông tin cấu trúc từ dữ liệu. Với hiệu suất cao trên các chỉ số đánh giá và khả năng ứng dụng linh hoạt, mô hình này không chỉ đáp ứng tốt yêu cầu hiện tại mà còn mở ra tiềm năng phát triển và áp dụng vào các bài toán phức tạp hơn trong tương lai.

Kết luận

Đồ án đã đạt được mục tiêu đề ra

Kết quả của đồ án

1. Trình bày nội dung lý thuyết về *GNN*s cơ bản và các cơ chế tổng hợp trong quá trình truyền thông điệp.
2. Hiểu và xây dựng được các mô hình Graph-based như GraphSage, Graph Attention,... với một hoặc nhiều đồ thị làm input.
3. Đề xuất các cách xây dựng đồ thị khác nhau (Đồ thị bệnh, đồ thị triệu chứng, xây dựng bộ feature cho bệnh dựa trên áp dụng độ đo TF-IDF với các triệu chứng nổi bật,...)
4. Đồ án đã đưa ra các kết quả thực nghiệm và đánh giá mô hình sau quá trình cài đặt nhóm mô hình *Graph-Based* đề xuất trên thực tế.

Kỹ năng đạt được

1. Đọc và nghiên cứu các báo cáo khoa học chuyên ngành, tài liệu tham khảo uy tín trên thế giới.
2. Viết đồ án và báo cáo có trình tự, rành mạch, cụ thể.
3. Rèn luyện khả năng tư duy cho một mô hình lớn.
4. Kỹ năng code cũng trở lên thành thạo và tối ưu hơn.

Hướng phát triển của đồ án trong tương lai

Trong thời gian tới, em sẽ tiếp tục nghiên cứu một số bài toán áp dụng tới *Học máy* đặc biệt là với *Graph Neural Networks* để xây dựng ý tưởng cho đề tài hiện tại và một số đề tài sau:

- Hệ thống Giao thông Thông minh (ITS): Dự báo giao thông hoặc phương tiện giao thông tự hành.
- Nghiên cứu các cách xây dựng đồ thị khác với nhiều kiểu biểu diễn khác nhau. Thiết kế trọng số cạnh cho đồ thị.
- Kết hợp các phương pháp xử lý đồ thị động phức tạp *(Dynamic Graphs)* cho phép xây dựng mạng lưới thông tin người dùng cho bài toán cá nhân hóa hệ thống gợi ý *(Personalized Recommendation)*.

Tài liệu tham khảo

Tiếng Việt

[1] Nguyễn, T. K. (2023). *"Nghiên cứu phát triển một số kỹ thuật gợi ý mua hàng theo phiên dựa trên mô hình học sâu"*. (Doctoral dissertation, Học viện Khoa học và Công nghệ).

Tiếng Anh

[2] Anees Kazi, Shayan Shekarforoush, S Arvind Krishna, Hendrik Burwinkel, Gerome Vivar, Karsten Kortüm, Seyed-Ahmad Ahmadi, Shadi Albarqouni, and Nassir Navab. Inceptiongcn: *"receptive field aware graph convolutional network for disease prediction"*. In Information Processing in Medical Imaging: 26th International Conference, IPMI 2019, Hong Kong, China, June 2–7, 2019, Proceedings 26, pages 73–85. Springer, 2019.

[3] Christopher M Bishop and Nasser M Nasrabadi. *"Pattern recognition and machine learning"*, volume 4. Springer, 2006.

[4] Fenglong Ma, Jing Gao, Qiuling Suo, Quanzeng You, Jing Zhou, and Aidong Zhang. *"Risk prediction on electronic health records with prior medical knowledge"*. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pages 1910–1919, 2018.

[5] Hao Du, Jiashi Feng, and Mengling Feng. Zoom in to where it matters: *"a hierarchical graph based model for mammogram analysis"*. arXiv preprint arXiv:1912.07517, 2019.

[6] Hamilton, W. L. (2020). *"Graph representation learning"*. Morgan & Claypool Publishers.

[7] Honglei Zhang and Mancef Gabbouj. *"Feature dimensionality reduction with graph embedding and generalized hamming distance"*. In *2018 25th IEEE International Conference on Image Processing (ICIP)*, pages 1083–1087. IEEE, 2018.

[8] Jack W Smith, James E Everhart, WC Dickson, William C Knowler, and Robert Scott Johannes. *"Using the adap learning algorithm to forecast the onset of diabetes mellitus"*. In Proceedings of the annual symposium on computer application in medical care, page 261. American Medical Informatics Association, 1988.

[9] J. Ross Quinlan. Simplifying decision trees. *"International journal of man-machine studies"*, 27(3):221–234, 1987.

[10] Luca Cosmo, Anees Kazi, Seyed-Ahmad Ahmadi, Nassir Navab, and Michael Bronstein. *"Latent-graph learning for disease prediction"*. In Medical Image Computing and Computer Assisted Intervention–MICCAI 2020: 23rd International Conference, Lima, Peru, October 4–8, 2020, Proceedings, Part II 23, pages 643–653. Springer, 2020.

[11] Qiuling Suo, Fenglong Ma, Ye Yuan, Mengdi Huai, Weida Zhong, Aidong Zhang, and Jing Gao. *"Personalized disease prediction using a cnn-based similarity learning method"*. In 2017 IEEE International Conference on Bioinformatics and Biomedicine (BIBM), pages 811–816. IEEE, 2017.

[12] Rushil Anirudh and Jayaraman J Thiagarajan. *"Bootstrapping graph convolutional neural networks for autism spectrum disorder classification"*. In ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 3197–3201. IEEE, 2019.

[13] Richard Hillestad, James Bigelow, Anthony Bower, Federico Girosi, Robin Meili, Richard Scoville, and Roger Taylor. *"Can electronic*

medical record systems transform health care? potential health benefits, savings, and costs". Health affairs, 24(5):1103–1117, 2005.

[14] Sarah Parisot, Sofia Ira Ktena, Enzo Ferrante, Matthew Lee, Ricardo Guerrerro Moreno, Ben Glocker, and Daniel Rueckert. *"Spectral graph convolutions for population-based disease prediction"*. In Medical Image Computing and Computer Assisted Intervention- MICCAI 2017: 20th International Conference, Quebec City, QC, Canada, September 11-13, 2017, Proceedings, Part III 20, pages 177–185. Springer, 2017.

[15] Sellappan Palaniappan and Rafiah Awang. *"Intelligent heart disease prediction system using data mining techniques"*. In 2008 IEEE/ACS international conference on computer systems and applications, pages 108–115. IEEE, 2008.

[16] Xiaoxiao Li, Yuan Zhou, Nicha Dvornek, Muhan Zhang, Siyuan Gao, Juntang Zhuang, Dustin Scheinost, Lawrence H Staib, Pamela Ventola, and James S Duncan. Braingnn: *"Interpretable brain graph neural network for fmri analysis"*. Medical Image Analysis, 74:102233, 2021.

[17] Xuegang Song, Feng Zhou, Alejandro F Frangi, Jiuwen Cao, Xiaohua Xiao, Yi Lei, Tianfu Wang, and Baiying Lei. *"Graph convolution network with similarity awareness and adaptive calibration for disease-induced deterioration prediction"*. Medical Image Analysis, 69:101947, 2021.

[18] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. *"Deep learning. nature"*, 521(7553):436–444, 2015.

[19] Yang Li, Buyue Qian, Xianli Zhang, and Hui Liu. *"Graph neural network-based diagnosis prediction"*. Big Data, 8(5):379–390, 2020.

[20] Yongxiang Huang and Albert CS Chung. *"Edge-variational graph convolutional networks for uncertainty-aware disease prediction"*. In Medical Image Computing and Computer Assisted Intervention–MICCAI 2020: 23rd International Conference, Lima, Peru, October 4–8, 2020, Proceedings, Part VII 23, pages 562–572. Springer, 2020.

Chỉ mục