TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN

Đào Đức Anh - Nguyễn Thành Nhân

Xây dựng hệ thống gợi ý sản phẩm dựa trên mô hình Auto-Encoder

KHÓA LUẬN TỐT NGHIỆP CỬ NHÂN
CHƯƠNG TRÌNH CHÍNH QUY

Tp. Hồ Chí Minh, tháng MM/YYYY

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN

Đào Đức Anh - 1712270
Nguyễn Thành Nhân - 1712631

Xây dựng hệ thống gợi ý sản phẩm dựa trên mô hình Auto-Encoder

KHÓA LUẬN TỐT NGHIỆP CỬ NHÂN
CHƯƠNG TRÌNH CHÍNH QUY

GIÁO VIÊN HƯỚNG DẪN
ThS. Trần Trung Kiên

Tp. Hồ Chí Minh, tháng MM/YYYY

Lời cảm ơn

Tôi xin chân thành cảm ơn ...

Mục lục

Danh sách hình

Danh sách bảng

TÓM TẮT

Chương 1

Giới thiệu

Hiện nay, với việc bùng nổ dữ liệu trên mạng Internet, người dùng có cơ hội tiếp cận nhiều hơn với đa dạng các sản phẩm trên nền tảng số. Song song đó, các nhà cung cấp dịch vụ cũng có cơ hội tiếp cận với người dùng nhiều hơn. Tuy nhiên, người dùng cũng đang gặp nhiều khó khăn khi tìm kiếm những nội dung phù hợp với nhu cầu của mình khi hiện nay có quá nhiều sự lựa chọn được đưa ra. Với mục đích nhằm giải quyết vấn đề trên, hệ thống gợi ý sản phẩm được xây dựng để có thể dự đoán cho người dùng những nội dung - hay còn được gọi là sản phẩm - phù hợp với họ. Hơn nữa, nó còn đóng vai trò quan trọng trong sự phát triển của các nhà cung cấp dịch vụ - doanh nghiệp, khi góp phần giúp nâng cao trải nghiệm người dùng cũng như tăng sự thu hút khách hàng. Theo số liệu tổng hợp được từ tổ chức Ivy Pro School [9], 38% lượt click từ người dùng Google đến từ hệ thống gợi ý và 35% sản phẩm được bán trên Amazon thông qua hệ thống gợi ý sản phẩm.

Trong lĩnh vực khoa học máy tính, hệ thống gợi ý sản phẩm là một chủ đề đang được quan tâm và nghiên cứu từ cộng đồng nghiên cứu khoa học. Bài toán xây dựng hệ thống gợi ý được phát biểu như sau:

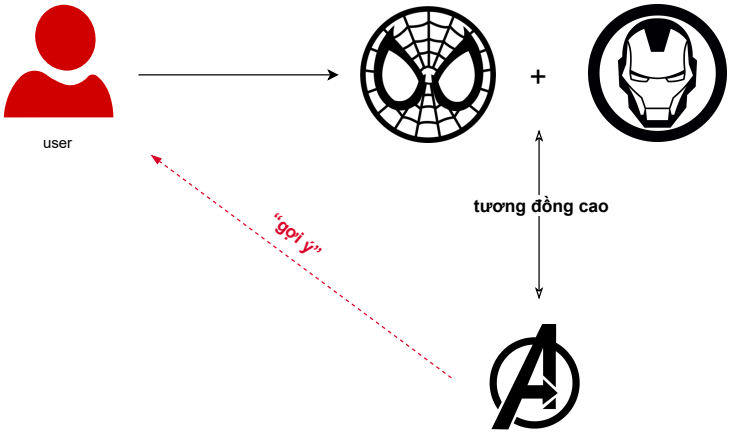
- Đầu vào là lịch sử tương tác của người dùng (user) với các sản phẩm (items) hoặc có thêm các thông tin mô tả của sản phẩm (các sản phẩm ở đây có thể là: quảng cáo, bộ phim, bài hát, văn bản để đọc,

... tùy thuộc vào lĩnh vực cụ thể).

- Yêu cầu máy tính tự động đưa ra các sản phẩm (không có trong lịch sử tương tác) được dự đoán là phù hợp với người dùng.

Tuy vậy, việc xây dựng một hệ thống gợi ý sản phẩm một cách hiệu quả là không đơn giản. Đầu tiên, tùy vào từng lĩnh vực cụ thể, ta cần xét đến nhiều yếu tố khác nhau khi xây dựng một hệ thống gợi ý. Từ thực tế cho thấy, các lĩnh vực mà sản phẩm "tiêu thụ" và "sản xuất" nhanh như: phim, hình ảnh, âm nhạc, ... thì hệ thống gợi ý sẽ ít nhiều đóng vai trò quan trọng. Cũng theo số liệu từ Ivy Pro School [9], 75% số bộ phim được thuê trên Netflix thông qua hệ thống gợi ý, chứng tỏ sự ảnh hưởng lớn của hệ thống gợi ý đối với lĩnh vực này. Mặt khác, hệ thống gợi ý tác động không nhiều đến các lĩnh vực cung cấp dịch vụ hay sản phẩm giá trị cao như: thuê nhà, phương tiện giao thông, thiết bị điện tử, ... vì người dùng cần đánh giá thông qua nhiều yếu tố mới có thể quyết định được. Thứ hai, tùy thuộc vào nhu cầu của người sử dụng mới có thể lựa chọn "cách" mà hệ thống gợi ý hoạt động. Việc gợi ý các sản phẩm phù hợp với người dùng dựa vào nhóm người dùng có sở thích tương tự với họ hay cách dựa trên các sản phẩm có liên quan với các sản phẩm mà họ đã "thích" trước đó là khác nhau. Bài toán gợi ý sản phẩm có thể xem như là một bài toán hồi quy ("regression") nếu mô hình dự đoán mức độ yêu thích của người dùng cho sản phẩm. Hoặc bài toán này có thể xem là một bài toán xếp hạng ("top-N ranking") nếu ta cần kết quả trả về là tập các sản phẩm phù hợp nhất với người dùng. Một điều nữa cũng có thể được xem là khó khăn thứ ba khi xây dựng hệ thống gợi ý, cả trong cộng đồng nghiên cứu khoa học cũng như thực tiễn, đó là ta cần một độ đo và một phương pháp để đánh giá một cách tổng thể và khách quan nhất, khi mà dữ liệu và các thuật toán để xây dựng hệ thống gợi ý là rất đa dạng.

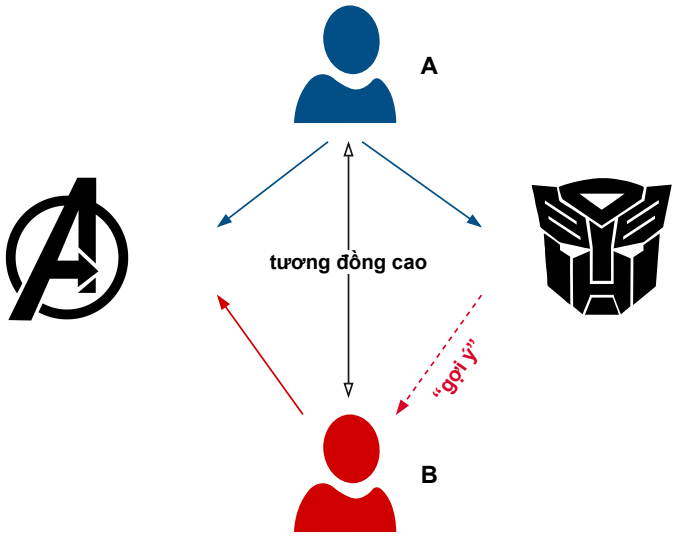
Để xây dựng hệ thống gợi ý, một hướng tiếp cận chúng ta thường nghĩ ngay đến đầu tiên là dự đoán các sản phẩm có "độ tương đồng" cao so với các sản phẩm người dùng đã "thích" trước đó, hướng tiếp cận này được



Hình 1.1: Minh họa cách hoạt động của "Content-Based Filtering": mô hình gợi ý bộ phim có độ tương đồng cao với các bộ phim người dùng đã xem trước đó

gọi là "Content-Based Filtering" (lọc dựa trên nội dung) (hình 1.1 mô tả hướng tiếp cận này). Với hướng tiếp cận này, mô hình chỉ cần các thuộc tính mô tả của sản phẩm mà không đòi hỏi dữ liệu tương tác từ người dùng khác vì các gợi ý là dành riêng cho từng cá nhân, do đó nó có khả năng nắm bắt tốt các sở thích đặc biệt của người dùng. Vì dựa trên "tính tương đồng" của sản phẩm, hệ thống có thể gợi ý một sản phẩm phù hợp với người dùng nhưng sản phẩm này có thể không được nhiều người dùng khác quan tâm. Vì dữ liệu đầu vào phụ thuộc vào từng lĩnh vực, ta cần chọn những thuộc tính có khả năng ảnh hưởng đến quyết định của mô hình cũng như loại bỏ các thuộc tính không cần thiết. Do đó cần phải có "domain knowledge" cụ thể khi muốn áp dụng "Content-Based Filtering" để xây dựng hệ thống gợi ý sản phẩm cho một lĩnh vực nào đó. Trong trường hợp các lĩnh vực có ít thông tin chi tiết về sản phẩm, hay các dữ liệu về sản phẩm thường không đầy đủ, rõ ràng thì hướng tiếp cận này sẽ tỏ ra không hiệu quả.

Một hướng tiếp cận khác là tìm ra "độ tương đồng" giữa các người dùng với nhau, hay tìm ra được một nhóm người dùng có cùng sở thích dựa trên dữ liệu tương tác của tất cả người dùng. Khi đó, để có thể gợi ý cho một người dùng cụ thể, hệ thống sẽ tìm ra các sản phẩm không có



Hình 1.2: Minh họa cách hoạt động của "Collaborative Filtering": hai người dùng cùng xem một (hoặc nhiều) bộ phim sẽ được hệ thống đánh giá là hai người dùng "tương đồng" nhau, khi đó một bộ phim được người dùng A xem sẽ được gợi ý cho người dùng B

trong lịch sử tương tác của người dùng đó, và đã được những người dùng "tương đồng" với họ tương tác trước đó, thì hướng tiếp cận này được gọi là "Collaborative Filtering" (lọc cộng tác) (hình 1.2 mô tả hướng tiếp cận này). Đối với hướng tiếp cận "Collaborative Filtering", mô hình dựa vào lịch sử tương tác từ người dùng khác và không dùng các thuộc tính mô tả của sản phẩm, do đó nó có khả năng tạo ra sự tình cờ cho người dùng: hệ thống có thể gợi ý một sản phẩm "tốt" cho người dùng trong trường hợp sản phẩm đó có ít điểm tương đồng so với các sản phẩm người dùng đã "thích" trước đó. Dữ liệu đầu vào của hướng tiếp cận này là các tương tác của người dùng với các sản phẩm, do đó có thể áp dụng cho nhiều lĩnh vực khác nhau mà không cần thiết phải thay đổi cấu trúc hệ thống hoặc nếu có thì cũng không cần phải thay đổi quá nhiều. Tuy nhiên, "Collaborative Filtering" gặp phải một vấn đề được gọi là "khởi động nguội" ("cold-start"), đó là khi mà một người dùng chưa có hoặc có ít tương tác thì hệ thống sẽ thường khó đưa ra được gợi ý "tốt" cho họ. "Collaborative Filtering" vẫn sẽ đưa ra gợi ý cho người dùng này tuy nhiên mức độ phù hợp đối với người dùng không cao, giảm khả năng thu hút khách hàng. Đối với

"Content-based Filtering", việc gợi ý các sản phẩm liên quan đến sản phẩm đòi hỏi người dùng đã tương tác với một vài sản phẩm trước đó, dựa trên đó mới có thể tính độ tương đồng giữa sản phẩm để đưa ra gợi ý.

"Cold-start" cũng là khi các sản phẩm chỉ được một số ít người dùng tương tác, các sản phẩm này sẽ ít được hệ thống gợi ý cho người dùng.

Trong giới hạn của khóa luận này, chúng tôi chỉ tập trung tìm hiểu về hướng tiếp cận "Collaborative Filtering" vì ba lý do chính là:

- "Collaborative Filtering" có thể áp dụng cho nhiều lĩnh vực khác nhau bởi đầu vào của mô hình là ma trận tương tác giữa người dùng và sản phẩm, khác với "Content-Based Filtering" cần thiết kế mô hình cụ thể cho từng lĩnh vực khác nhau.

- Với số lượng lớn và cùng với sự đa dạng của các "sản phẩm" hiện nay, việc gợi ý các sản phẩm tương đồng với nhau dễ trở nên nhàm chán, thay vào đó "Collaborative Filtering" gợi ý sản phẩm dựa trên sở thích của người dùng, giúp các sản phẩm bớt nhàm chán hơn.

- Ngoài ra, khi số lượng người dùng trên mạng Internet càng ngày càng tăng nhanh, thì "Collaborative Filtering" sẽ có được nhiều lợi thế hơn khi có thể kết hợp dữ liệu tương tác của các người dùng khác để đưa ra gợi ý.

Việc kết hợp giữa thông tin tương tác từ người dùng khác và thông tin chi tiết của sản phẩm được gọi là hướng tiếp cận "Hybrid", đây là hướng tiếp cận kết hợp giữa "Collaborative Filtering" và "Content-Based Filtering". Hướng tiếp này sẽ giúp một hệ thống gợi ý sản phẩm hoạt động hiệu quả hơn. Nhưng đây là một điều không đơn giản bởi nó phụ thuộc vào "domain knowledge" ở từng lĩnh vực và chúng tôi để lại như một định hướng trong việc nghiên cứu và phát triển trong tương tai.

Phương pháp đầu tiên trong việc xây dựng một hệ thống gợi ý sản phẩm theo hướng tiếp cận "Collaborative Filtering" là thuật toán "Matrix Factorization" được giới thiệu bởi Hu [5]. Ý tưởng là từ dữ liệu tương tác

(ví dụ, điểm đánh giá) của các người dùng với các sản phẩm, ta sẽ tìm ra các véc-tơ đặc trưng cho người dùng và các véc-tơ đặc trưng cho sản phẩm. Sau đó, ta sẽ dùng các véc-tơ đặc trưng này để dự đoán tương tác của người dùng với cácc sản phẩm mà họ chưa tương tác; từ kết quả dự đoán, ta sẽ đưa ra các đề xuất sản phẩm cho các người dùng. Cho đến hiện nay, "Matrix Factorization" vẫn là một phương pháp đơn giản nhưng vẫn mang lại kết quả cao. Tuy nhiên, thuật toán này có các nhược điểm mà khó có thể được áp dụng để xây dựng một hệ thống gợi ý sản phẩm quy mô lớn. Đầu tiên, đó là số lượng tham số của mô hình tỉ lệ tuyến tính vào cả số lượng người dùng và số lượng sản phẩm. Ngày nay, số lượng người dùng và sản phẩm tăng rất nhanh theo thời gian, do đó số lượng tham số cũng như chi phí tính toán của mô hình là rất lớn. Vì mô hình cần phải tìm ra các véc-tơ đặc trưng cho cả người dùng và sản phẩm, khi có người dùng mới, mô hình cần phải thực hiện một số bước tính toán để có thể tìm được đặc trưng của người dùng đó. Ngoài ra, "Matrix Factorization" vẫn còn hạn chế đó là mô hình này là một mô hình tuyến tính, do đó nó chưa có khả năng "học" được các "đặc trưng phi tuyến" của dữ liệu.

"Asymetric Matrix Factorization" là một phương pháp cải tiến từ "Matrix Factorization" với ý tưởng rút trích các đặc trưng của người dùng thông qua các sản phẩm mà họ đã tương tác. Phương pháp này đã khắc phục được nhược điểm của "Matrix Factorization" khi mà số lượng tham số của mô hình giờ chỉ phụ thuộc vào số lượng sản phẩm có trong hệ thống. Trong thực tế, số lượng sản phẩm sẽ tăng chậm hơn đáng kể so với số lượng người dùng trong hệ thống thì khắc phục này sẽ là một điểm mạnh của "Asymetric Matrix Factorization". Cũng như mô hình này đã giảm bớt được chi phí tính toán để đưa ra dự đoán cho người dùng mới. Tuy nhiên, với hạn chế của các hàm tuyến tính nên phương pháp này vẫn chưa thực sự hiệu quả.

Ở công trình nghiên cứu [11] của tác giả Steck đã chỉ ra rằng "Asymetric Matrix Factorization" có thể được xem như là một mô hình "Auto-Encoder" tuyến tính. "Auto-Encoder" là một mô hình học đặc trưng ẩn không giám

sát. Mô hình này thường được sử dụng trong những tác vụ như rút trích đặc trưng hay giảm chiều dữ liệu, ...

Dựa trên ý tưởng là tồn tại các "đặc trưng ẩn" thể hiện sở thích của người dùng, các đặc trưng ẩn này sẽ dẫn đến người dùng "tương tác" với sản phẩm nào. Việc áp dụng mô hình "Auto-Encoder" hoàn toàn khớp với ý tưởng này, theo đó "Auto-Encoder" sẽ rút trích các "đặc trưng ẩn" này thông qua "encoder" (một phần trong mô hình "Auto-Encoder"), sau đó tái tạo lại tương tác của người dùng dựa vào đặc trưng ẩn này. Tương tác được tái tạo lại sẽ bao gồm các gợi ý mà hệ thống đưa ra.

Trong thời gian gần đây đã có nhiều nghiên cứu áp dụng mô hình "Auto-Encoder" trong bài toán xây dựng hệ thống gợi ý sản phẩm để có thể tận dụng sức mạnh của các hàm phi tuyến, cụ thể là sử dụng mạng nơ-ron với các hàm kích hoạt phi tuyến (là kiến trúc cơ bản của các mô hình được dùng huấn luyện trong lĩnh vực học máy) để có được một mô hình "mạnh" hơn so với các phương pháp tuyến tính trước đó. "AutoRec" [10] được Sedhain giới thiệu tại hội nghị WWW2015, là mô hình được coi là đầu tiên trong việc sử dụng kiến trúc mô hình "Auto-Encoder" để đưa ra gợi ý cho người dùng bằng cách huấn luyện mô hình để tái tạo lại dữ liệu tương tác của người dùng sau khi trích xuất đặc trưng ẩn từ dữ liệu tương tác của họ. "Collaborative denoising auto-encoders for top-n recommender systems" [12] (CDAE) được Wu cùng các cộng sự đề xuất nhằm hướng đến bài toán đưa ra gợi ý theo hướng xếp hạng và đưa ra "top-N sản phẩm" phù hợp nhất với người dùng hay nói cách khác là dự đoán tập các sản phẩm mà người dùng "thích" nhất. Mô hình này dã được xây dựng dựa trên "AutoRec" nhưng có các chỉnh sửa để phù hợp hơn với bài toán xây dựng hệ thống gợi ý sản phẩm khi mà ta quan tâm đến việc đưa ra xếp hạng các sản phẩm phù hợp với người dùng thay vì tái tạo lại tương tác của họ. Ngoài ra, việc thêm "nhiễu" vào dữ liệu huấn luyện của CDAE giúp mô hình "Auto-Encoder" phải học cách trích xuất các đặc trưng ẩn tốt và tạo ra gợi ý thay vì chỉ cố gắng tái tạo lại dữ liệu tương tác "giống" đầu vào nhất có thể, cũng như "nhiễu" sẽ giúp quá trình huấn luyện mạng

nơ-ron với hàm kích hoạt phi tuyến tránh được tình trạng "over-fitting" (tình trạng học "tủ" trên tập dữ liệu huấn luyện, dẫn đến mô hình đạt được kết quả thấp trên các tập dữ liệu kiểm định).

Một trong những phương pháp nổi bật nhất tới thời điểm hiện tại trong việc sử dụng kiến trúc mô hình "Auto-Encoder" để xây dựng mô hệ thống gợi ý sản phẩm là "Variational Autoencoder for Collaborative Filtering"[7] được giới thiệu bởi tác giả Liang cùng các cộng sự công bố tại hội nghị "International World Wide Web Conference Committee 2018". Đây là một phương pháp sử dụng mô hình "Variational Auto-encoder" (VAE) - một biến thể của mô hình "Auto-Encoder" cơ bản để có thể xây dựng một hệ thống gợi ý sản phẩm hiệu quả. "Variational Auto-encoder" đã đạt được một số thành công nhất định trong bài toán xây dựng hệ thống gợi ý. Theo nghiên cứu của Dacrema [3], mô hình được đề xuất trong [7] đạt được các kết quả tốt trong việc gợi ý và xếp hạng.

Với nền tảng của mô hình VAE dựa trên phương pháp "Variational Inference" trong lĩnh vực xác suất thống kê. "Variational Inference" dùng để suy diễn dữ liệu ẩn từ dữ liệu ta quan sát được, hay cụ thể trong bài toán này là suy diễn các "đặc trưng ẩn" dựa vào dữ liệu quan sát được là các tương tác của người dùng. Đặc điểm của phương pháp này là có thể áp dụng tốt cho dữ liệu thưa, có nghĩa là đối với "dữ liệu quan sát được" bị hạn chế thì việc "suy diễn" dữ liệu vẫn đạt được kết quả tốt. Dữ liệu thưa là dữ liệu mà đa phần các phần tử trong một điểm dữ liệu mang giá trị 0. Trong hệ thống gợi ý, tính chất của dữ liệu thường là thưa do mỗi người dùng chỉ tương tác với một lượng nhỏ sản phẩm trên toàn hệ thống, từ đó việc suy diễn trở nên hiệu quả trong hệ thống gợi ý.

Mục tiêu sau cùng của hệ thống gợi ý sản phẩm đó là tập sản phẩm được gợi ý sẽ phù hợp nhất đối với người dùng. Do đó, trong tập sản phẩm này thì cũng cần phải được sắp xếp theo thứ tự giảm dần theo thứ tự giảm dần về độ phù hợp. Để kết quả trả về từ mô hình có thể đạt được mục tiêu trên thì tác giả Liang đã giới thiệu "Multinomial log-likelihood" cho việc tính toán độ lỗi. Với tính chất trả về một giá trị xác suất cho mỗi sản

phẩm, và tổng giá trị xác suất trên toàn bộ sản phẩm là 1. Các sản phẩm sẽ phải "cạnh tranh" với nhau để có được xác suất cao hơn.

Mặc dù mô hình được tác giả Liang đề xuất ở bài báo [7] không phải là mô hình đạt được kết quả tốt nhất hiện nay trong việc xây dựng hệ thống gợi ý sản phẩm, tuy nhiên kiến thức nền tảng để xây dựng mô hình này bao phủ về lĩnh vực học máy cũng như là kiến thức về mô hình xác suất. Đó là lý do chúng tôi quyết định sẽ tập trung tìm hiểu mô hình này trong khóa luận.

Phần còn lại của khóa luận được trình bày như sau:

- Chương 2 trình bày sơ lược về mô hình "Auto-Encoder" và các kiến thức nền tảng của mô hình "Variational Auto-Encoder".
- Chương 3 trình bày về cách áp dụng mô hình "Variational Auto-Encoder" cùng với hàm lỗi "Multinomial log-likelihood" cho bài toán xây dựng hệ thống gợi ý. Bên cạnh đó, chương này cũng phân tích các hạn chế của mô hình đồng thời đề xuất phương pháp giúp giải quyết các hạn chế đó. Chương này là phần chính của khóa luận.
- Chương 4 trình bày về các thí nghiệm và các kết quả đạt được.
- Cuối cùng, tổng kết và hướng phát triển được trình bày ở chương 5.

Chương 2

Kiến thức nền tảng

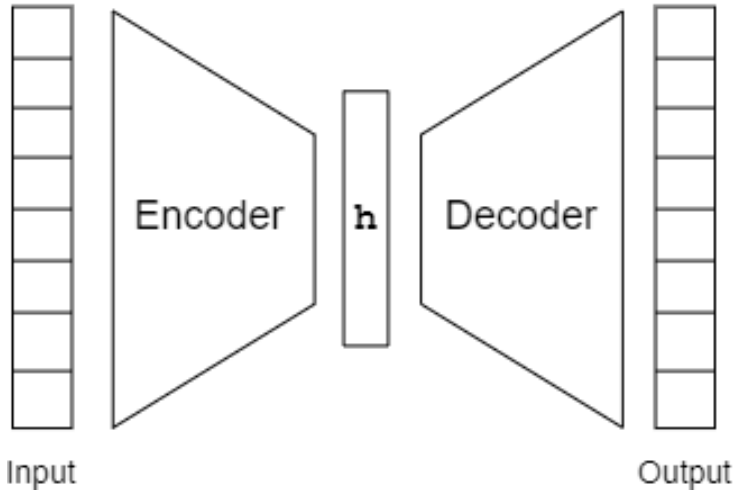
Trong chương này, đầu tiên chúng tôi sẽ trình bày về mô hình "Auto-Encoder" (AE), một mạng nơ-ron được dùng để học đặc trưng ẩn dựa trên phương pháp học không giám sát. Kế tiếp, chúng tôi giới thiệu và trình bày về mô hình "Variational Auto-Encoder" (VAE) và các nền tảng xác suất của nó. Các nền tảng xác suất bao gồm: Variational Inference - một phương pháp suy diễn dữ liệu hiệu quả - và Maximum Likelihood Estimation - một phương pháp để ước lượng một bộ tham số tốt cho mô hình.

2.1 Mô hình rút trích đặc trưng ẩn "Auto-Encoder"

Mô hình "Auto-Encoder" (AE) là một mạng nơ-ron truyền thẳng được huấn luyện để cố gắng sao chép đầu vào của nó thành đầu ra với mục đích trích xuất được các đặc trưng ẩn. Bên trong "Auto-Encoder" có một lớp ẩn $\boldsymbol{h}$ mô tả đặc trưng ẩn, gọi là véc-tơ biểu diễn ẩn đại diện cho đầu vào của nó.

Kiến trúc của một "Auto-Encoder" (được minh họa trong hình 2.1) bao gồm hai phần:

- Bộ mã hóa (encoder) ánh xạ véc-tơ đầu vào sang véc-tơ biểu diễn ẩn:



Hình 2.1: Minh họa "Auto-Encoder"

$$\boldsymbol{h} = f(x)$$

- Bộ giải mã (decoder) có nhiệm vụ cố gắng tái tạo lại véc-tơ đầu vào từ véc-tơ biểu diễn ẩn:

$$\hat{x} = g(\boldsymbol{h}) = g(f(x))$$

"Auto-Encoder" được huấn luyện bằng cách cực tiểu hóa hàm lỗi là độ sai lệch giữa dữ liệu được tái tạo với dữ liệu đầu vào.

$$L(x, g(f(x))) \tag{2.1}$$

Các hàm để tính độ lỗi thường được dùng là "Mean-square error" hoặc "Binary cross-entropy". Tương tự như các mạng nơ-ron khác, "Auto-Encoder" có thể được huấn luyện bằng phương pháp "Gradient-descent" với thuật toán lan truyền ngược ("back-propagation").

Khi thiết kế mô hình, kiến trúc của encoder, decoder và kích thước của véc-tơ $\boldsymbol{h}$ được xem như những siêu tham số của mô hình. Bằng các cách thiết lập khác nhau, mô hình sẽ có những tính chất khác nhau. "Auto-Encoder" với encoder và decoder là những hàm phi tuyến (cụ thể là mạng nơ-ron với hàm kích hoạt phi tuyến) với khả năng tính toán quá mạnh hay

trường hợp kích thước của véc-tơ $\boldsymbol{h}$ lớn hơn hoặc bằng so với véc-tơ đầu vào sẽ dẫn đến mô hình chỉ học cách sao chép thay vì trích xuất các đặc trưng ẩn từ dữ liệu.

Thông thường, một "Auto-Encoder" sao chép một cách "hoàn hảo" đầu vào thành đầu ra sẽ không có nhiều ý nghĩa. Thay vào đó, "Auto-Encoder" được thiết kế với các ràng buộc để không thể học cách sao chép "hoàn hảo" mà chỉ có thể sao chép gần đúng, từ đó ta hy vọng quá trình huấn luyện "Auto-Encoder" sẽ thu được véc-tơ biểu diễn ẩn có những thông tin hữu ích.

Từ véc-tơ biểu diễn ẩn thu được trong quá trình huấn luyện "Auto-Encoder", ta có thể áp dụng mô hình này như một mô hình trích xuất đặc trưng ẩn từ dữ liệu, làm đầu vào cho các tác vụ khác. Hoặc véc-tơ biểu diễn ẩn này có thể áp dụng được trong các tác vụ giảm chiều dữ liệu hỗ trợ cho các tác vụ lưu trữ, truy vấn, tìm kiếm.

2.1.1 "Undercomplete Auto-Encoder"

Như đã trình bày trước đó, việc sao chép đầu vào thành đầu ra của "Auto-Encoder" không mang nhiều ý nghĩa. Với mục đích thu được véc-tơ biểu diễn ẩn của dữ liệu thông qua quá trình huấn luyện, ta cần các ràng buộc để có được $\boldsymbol{h}$ nhận các thuộc tính hữu ích khi thiết kế mô hình.

Một cách ràng buộc để mô hình có thể học được các đặc trưng ẩn từ dữ liệu là giới hạn véc-tơ đặc trưng ẩn $\boldsymbol{h}$ có kích thước nhỏ hơn đáng kể so với véc-tơ đầu vào; tính chất này được gọi là "under-complete".

Mô hình "Auto-Encoder" với kích thước $\boldsymbol{h}$ nhỏ hơn đáng kể so với kích thước của véc-tơ đầu vào được gọi là "Undercomplete Auto-Encoder". Việc giới hạn này sẽ buộc mô hình phải nắm bắt các đặc trưng nổi bật nhất. Đây cũng là kiến trúc đa số các mô hình "Auto-Encoder" thường hay được sử dụng.

Quá trình huấn luyện "Undercomplete Auto-Encoder" cũng giống với mô hình "Auto-Encoder", ta cần cực tiểu hóa hàm lỗi (công thức 2.1) là độ

sai lệch giữa dữ liệu được tái tạo với dữ liệu đầu vào.

"Undercomplete Auto-Encoder" là mô hình tốt để sử dụng cho các tác vụ tiêu biểu của "Auto-Encoder" truyền thống như trích xuất đặc trưng, giảm chiều dữ liệu bởi vì tính chất "under-complete" của mô hình giúp dễ dàng thu được véc-tơ biểu diễn ẩn mang những thông tin hữu ích.

2.1.2 Biến thể của Auto-Encoder: "Denoising Auto-Encoder"

Hàm lỗi của một "Auto-Encoder" thông thường sẽ "phạt" một mức nhất định với các mẫu dữ liệu được tái tạo lại khác với dữ liệu đầu vào. Điều này vô hình chung khuyến khích việc $f \circ g$ là một hàm đồng nhất nếu khả năng tính toán của f và g cho phép. Nói đơn giản hơn, điều này là việc mô hình sao chép "hoàn hảo" đầu vào thành đầu ra của nó. Khi đó, véc-tơ biểu diễn ẩn sẽ không có các thông tin hữu ích.

Bằng cách thay đổi cách tính toán độ lỗi khi tái tạo lại, cụ thể là thêm nhiễu vào véc-tơ đầu vào, sau đó tính toán độ lỗi là đầu ra được mô hình tái tạo lại so với đầu vào ban đầu như sau:

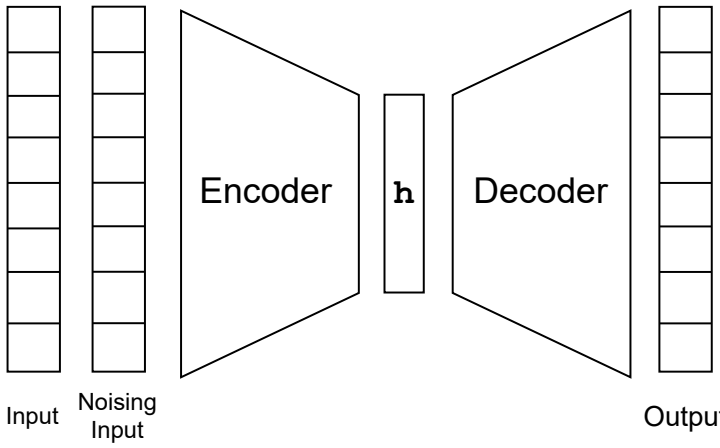
$$L(x, g(f(\tilde{x}))) \tag{2.2}$$

với $\tilde{x}$ là véc-tơ đầu vào x được thêm một độ nhiễu, ta có được mô hình "Denoising Auto-Encoder" (DAE) (hình 2.2).

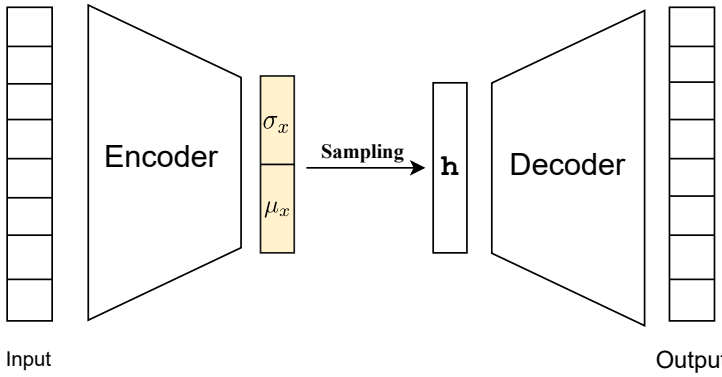
"Denoising Auto-Encoder" phải học cách khử độ nhiễu đã được thêm vào véc-tơ đầu vào, giảm khả năng sao chép của mô hình.

2.2 Mô hình phát sinh dữ liệu "Variational Auto-Encoder"

"Variational Auto-Encoder" (VAE) là một biến thể đặc biệt của "Auto-Encoder" cơ bản (được minh họa trong hình 2.3). VAE ngoài là một mô



Hình 2.2: Minh họa "Denoising Auto-Encoder"



Hình 2.3: Minh họa "Variational Auto-Encoder"

hình rút trích đặc trưng ẩn dựa trên phương pháp học không giám sát, còn là một mô hình phát sinh dữ liệu hiệu quả. Phát sinh dữ liệu là việc mô hình có khả năng tạo ra những điểm dữ liệu 'mới' dựa trên đặc trưng ẩn đã học được. Đây là một điểm khác biệt so với mô hình "Auto-Encoder" khi mà đặc trưng ẩn học từ "Auto-Encoder" cơ bản không thể được sử dụng để phát sinh. Điều tạo nên sự khác biệt này là bởi đặc trưng ẩn có được từ VAE là một phân bố chứ không phải là một điểm dữ liệu cụ thể. "Auto-Encoder" hay kể cả "Denosing Auto-Encoder", việc nhận dữ liệu đầu vào và trích xuất đặc trưng ẩn đều có thể được xem như là một phép chiếu dữ liệu ở chiều không gian cao lên một chiều không gian thấp hơn (thông thường thì kích thước của véc-tơ biểu diễn ẩn của một "Auto-Encoder" sẽ có tính chất "under-complete" như đã đề cập ở phần 2.1.1). Do đó, ta có thể xem đặc trưng ẩn này như là một điểm dữ liệu ở một chiều không gian khác với số chiều thấp hơn thể hiện cho dữ liệu ban đầu. Mặt khác, với VAE thì đặc trưng ẩn không còn là một điểm dữ liệu, thay vào đó sẽ là một "phân phối xác suất". Phân phối xác suất thể hiện cho phân bố của một đại lượng, một biến ngẫu nhiên nào đó.

Tuy nhiên, để làm rõ được sự hiệu quả của VAE trong việc phát sinh đặc trưng và phát sinh dữ liệu từ đặc trưng thì ta cần phải xét qua góc nhìn xác suất của mô hình này. Bản chất của một mô hình VAE là một mô hình đồ thị (graphical models) - là một mô hình dùng để giải thích các mối quan hệ giữa các biến ngẫu nhiên trong xác suất thống kê. Và nền tảng của mô hình là "Variation Inference" - là một phương pháp cũng thuộc lĩnh vực xác suất thống kê với mục đích có thể "giải thích" được dữ liệu mà ta không quan sát được từ những dữ liệu mà ta đã có. Tận dụng sức mạnh của mạng nơ-ron trong lĩnh vực học máy, các hàm số xác suất được thay thành các mạng nơ-ron. Và thông qua việc huấn luyện mô hình để tìm ra bộ trọng số tốt nhất để giải quyết bài toán được giả định mà mô hình cần giải quyết.

Do sự liên hệ chặt chẽ với lĩnh vực xác suất, ở mục này, chúng tôi sẽ trình bày về nền tảng xác suất liên quan với mô hình "Variational Auto-

Encoder", bao gồm các khái niệm, định lý trong lĩnh vực xác suất thống kê để có thể dễ dàng trình bày nội dung của VAE ở mục tiếp theo, cũng như là cách huấn luyện cho mô hình VAE.

2.2.1 Nền tảng xác suất của mô hình

Với sự tăng nhanh về số lượng dữ liệu có trên các nền tảng số thì nhu cầu cần một phương pháp có thể phân tích dữ liệu một cách tự động đang càng ngày càng tăng theo. Mục tiêu của học máy đó là phát triển các phương pháp mà có thể tự động phát hiện các mẫu "pattern" trong dữ liệu. Các mô hình học máy tìm được những "pattern" này thông qua việc tìm các bộ tham số phù hợp với mô hình. Sau đó sử dụng những "pattern" vừa khám phá được để có thể dự đoán dữ liệu trong tương lai hoặc để thực hiện các mục đích khác như đưa ra các quyết định. Lý thuyết xác suất ("probability theory") có thể được áp dụng cho bất kỳ vấn đề nào liên quan đến "những điều chưa chắc chắn". Trong máy học, "những điều chưa chắc chắn" có thể là những sự thay đổi trong dữ liệu mà chúng ta chưa biết trước được, hay là những bộ tham số của mô hình. Gọi là chưa chắc chắn bởi vì ta không biết trước được và cũng không quan sát được về những điều này và ta không thể khẳng định hoàn toàn về những điều đó. Mô hình sẽ hoạt động như thế nào khi dữ liệu thay đổi? Liệu bộ tham số hiện tại của mô hình đã đủ tốt? Do đó học máy có liên quan khá là gần gũi với lĩnh vực xác suất thống kê và khai thác dữ liệu.

Trên lý thuyết thì có ít nhất hai cách diễn giải của xác suất: "diễn giải tần suất" (frequentist interetation) và "diễn giải bayesian". Ở cách diễn giải thứ nhất thì xác suất được thể hiện thông qua việc thực hiện các thí nghiệm nhiều lần. Ví dụ như nếu ta thực hiện thí nghiệm tung đồng xu thì ta kì vọng rằng việc đồng xu xuất hiện mặt ngửa khoảng một nửa lần trong quá trình thực hiện. Còn ở cách diễn giải bayesian của xác suất thì thường được sử dụng để định lượng về "những điều chưa chắc chắn". Vậy nên ở góc nhìn này sẽ liên quan đến các thông tin hơn là việc lặp lại các

thí nghiệm. Một trong những ưu điểm của cách diễn giải này đó là nó có thể được sử dụng để mô hình "những điều chưa chắc chắn" của sự việc/sự kiện mà ta đang quan tâm đến mà không có tần suất xuất dài hạn. Ví dụ liên hệ với các bài toán trong lĩnh vực học máy như chúng ta nhận một email và ta quan tâm đến việc tính phân phôi xác suất mà email vừa nhận là spam; hay trong bài toán chúng ta nhận thấy được một vật thể thông qua màn hình radar và ta muốn tính phân phối xác suất theo vật thể vừa được phát hiện chính xác là gì? một con chim, hay máy bay? Trong những trường hợp trên thì ý tưởng việc lặp lại các thí nghiệm sẽ không giúp ích cho chúng ta trong việc giải quyết các vấn đề nhưng với Bayesian thì điều này khá là tự nhiên và có thể được áp dụng để giải quyết bất kỳ vấn đề nào liên quan tới những "điều không chắc chắn".

Định lý Bayes và ứng dụng trong lĩnh vực học máy

Trong lĩnh vực "máy học" và "thống kê Bayesian", chúng ta thường quan tâm đến việc thực hiện các phép suy diễn dữ liệu ẩn ta không quan sát được khi cho trước các dữ liệu ta quan đã quan sát được. Ví dụ như ứng dụng một mô hình học máy trong việc phát hiện sản phẩm lỗi, khi ta đã có ghi nhận lại một số lượng dữ liệu mô tả của sản phẩm và đã biết được sản phẩm nào có lỗi hay không, đây chính là dữ liệu mà ta đã quan sát được. Điều mà ta quan tâm đến đó là mô hình có thể phát hiện được những "pattern" hay những đặc trưng quyết định đến việc xác định một sản phẩm có được xem là sản phẩm lỗi hay không, thì những "pattern" hoặc những đặc trưng này sẽ được xem là những dữ liệu ẩn.

Giả sử rằng, ta có a là biến ngẫu nhiên thể hiện cho dữ liệu ẩn mà ta không có dữ liệu về nó, và b là biến ngẫu nhiên của dữ liệu mà ta có thể quan sát được. Theo đó, ta sẽ quan tâm đến việc tìm ra được giá trị a cụ thể khi cho trước giá trị b. Về xác suất, hay cụ thể ở đây, theo định lý Bayes, nếu ta có $p(a)$ là thông tin mà ta đã biết trước về biến ta không quan sát được a; và một số mẫu dữ liệu thể hiện mối quan hệ giữa a và b

được thể hiện bởi $p(b|a)$, theo công thức Bayes, ta có:

$$p(a|b) = \frac{p(b|a)p(a)}{p(b)} \tag{2.3}$$

Trong đó:

- $p(a)$ được gọi là "prior", prior thể hiện cho "kiến thức biết trước" theo góc nhìn chủ quan ban đầu của chúng ta trước khi ta có bất kỳ về thông tin nào về liệu mà ta quan sát được. prior có thể được thể hiện thông qua một phân phối xác suất theo biến ẩn, nó có thể là phân phối xác suất bất kỳ sao cho phù hợp với chúng ta, nhưng một điều chúng ta cần phải đảm bảo đó là phân phối prior phải là có giá trị khác không trên tất cả các giá trị có thể xuất hiện của a, kể cả khi giá trị đó rất hiếm khi xảy ra.

- $p(b|a)$ là "likelihood", mô tả mối quan hệ giữa a và b liên quan với nhau như thế nào, và cụ thể thì nó là khả năng của việc xảy ra giá trị b khi ta đã biết về dữ liệu ẩn a cụ thể.

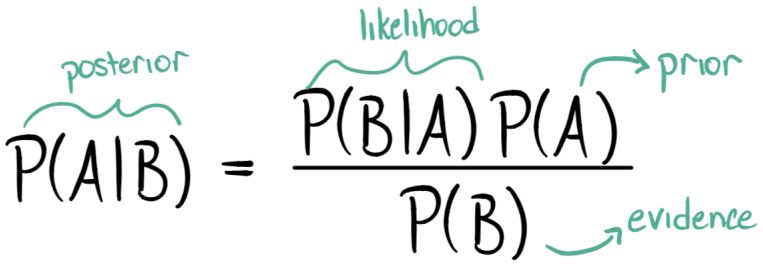
- Phân phối "posterior" $p(a|b)$ sẽ là giá trị mà ta quan tâm theo quan điểm của Bayes. Nó thể hiện rằng chúng ta có được thông tin gì về dữ liệu ẩn a không quan sát được khi ta có dữ liệu quan sát được là b.

- $p(b)$ là "evidence" hay cũng còn được biết đến là "marginal likelihood". Phân phối thể hiện cho khả năng xảy ra của một giá trị B cụ thể. Ngoài ra evidence độc lập với a nó còn có vai trò để chuẩn hoá cho posterior, có nghĩa là posterior sẽ có khoảng giá trị từ 0 đến 1.

Các đại lượng trong công thức 2.3 được chú thích trong hình 2.4

Khó khăn trong việc tính toán

Với góc nhìn này, "posterior" sẽ là giá trị mà chúng ta quan tâm đến, nó thể hiện mối quan hệ giữa "prior" của chúng ta và dữ liệu. Việc tính



Hình 2.4: Định lý "Bayes"

toán "posterior" sẽ giúp chúng ta giải quyết các vấn đề trong thực tế. Theo công thức 2.3, để tính toán "posterior" ta cần phải có: "prior", "likelihood" và "evidence". Hai giá trị ở trên tử số ("prior" và "likelihood") ta có thể dễ dàng xác định được trong hầu hết các trường hợp vì đó một phần là giả định của chúng ta về mô hình. Tuy nhiên, ở mẫu số ta cần tính:

$$p(b) = \int p(b|a)p(a)\mathrm{d}a = \mathbb{E}_a[p(b|a)] \tag{2.4}$$

theo đó ta thấy được để tính được "marginal likelihood" thì ta cần tính biểu thức với dấu tích phân. Để tính giá trị này với dữ liệu ở chiều không gian thấp có thể không gặp nhiều khó khăn, nhưng khi tính toán ở những chiều không gian cao thì nó có thể trở thành một vấn đề nan giải. Cụ thể ta thấy được rằng việc tính "marginal likelihood" sẽ thể hiện giá trị "likelihood" trung bình trên toàn bộ giá trị có thể xuất hiện của a, là những điều chưa chắc chắn trong mô hình, do đó a ở chiều không gian càng cao thì việc tính toán càng trở nên phức tạp hơn.

Chúng ta cần chú ý thêm một vài khó khăn khác có thể phải đối mặt khi tính toán "posterior" đó là việc lấy "tổ hợp" khi dữ liệu là rời rạc thay vì giá trị liên tục. Ở miền không gian liên tục thì ta có thể áp dụng hàm số trong lĩnh vực giải tích để tính toán, tuy nhiên trong những trường hợp mà chiều không gian của dữ liệu không liên lục, dữ liệu rời rạc thì việc tính toán sẽ còn phải xét thêm việc lấy tổ hợp dữ liệu.

Khi dữ liệu có số chiều lớn thì việc tính chính xác giá trị "posterior"

trong thực tiễn thường sẽ là một việc cực kỳ khó khăn và bất khả thi và ta cần một vài kĩ thuật xấp xỉ thường được dùng để giải quyết việc tính "posterior".

Bài toán Inference

Inference là một lớp bài toán để giải quyết vấn đề tìm hiểu về nhưng thứ mà ta biết được dựa trên những thứ mà ta đã biết. Nói một cách khác thì bài toán này là tiến trình để có thể dưa ra kết luận giá trị ước lượng, hay khoảng tin cậy hoặc xấp xỉ một phân phối cho một "biến ẩn" ("latent variable") thường được gọi là kết quả hay nhãn trong mẫu dữ liệu, dựa trên một vài các biến mà ta đã quan sát được thường được gọi là nguyên nhân hay dữ là dữ liệu đầu vào trong mẫu dữ liệu. Ví dụ như ta có dữ liệu là hình ảnh của các đối tượng trong tự nhiên và có nhãn đi kèm mỗi ảnh, bài toán inference sẽ trả lời câu hỏi rằng nếu một tấm ảnh mới không có trước đó thì liệu ta có biết được nhãn của đối tượng trong ảnh hay không?.

"Bayesian inferene" là việc giải quyết bài toán inference dựa trên "định lý Bayes". Phương pháp Bayesian inference là một phương pháp trong lĩnh vực xác suất thống kê mà ở đó kiến thức biết được biết trước "prior knowledge" được mô hình hoá bởi một phân phối xác suất và được cập nhật mỗi khi có một quan sát mới và những thứ mà ta không chắc chắn hay không quan sát được sẽ được mô hình bởi một phân phôi xác suất khác. Một ví dụ kinh diển là về các tham số của "Bayesian inference", giả định rằng một mô hình mà dữ liệu x được phát sinh từ một phân phối xác suất mà phân phối xác suất này được xác định bởi các tham số θ, tuy nhiên giá trị của θ thì ta chưa biết. Bên cạnh đó, ta giả định rằng, ta có một vài kiến thức được biết từ θ được gọi là "prior knowledge", nó có thể là phân phối xác suất $p(\theta)$. Sau đó, mỗi khi ta có một quan sát x mới, ta có thể cập nhật lại "prior knowledge" về tham số θ thông qua định lý Bayes theo công thức:

trong đó

Bayesian Inference là một vấn đề thường được phải giải quyết trong

các bài toán trong lĩnh vực xác suất thống kê tuy nhiên trong lĩnh vực học máy, nhiều phương pháp được xây dựng dựa trên việc giải quyết vấn đề Bayesian Inference. Ví dụ: "Gaussian mixture models" được dùng để giải quyết bài toán phân lớp, hay "Latent Dirichlet Allocation" để giải quyết bài toán phân loại chủ đề văn vản. Và cả hai mô hình kể trên đều được xây dựng dựa trên việc giải quyết bài toán Bayes Inference.

Variational inference

Variational inference (VI) là một phương pháp thường hay được sử dụng để giải quyết bài toán "Bayesian inference". Phương pháp này sử dụng hướng tiếp cận là tìm ra xấp xỉ tốt nhất cho một phân phối xác suất bằng cách tìm ra giá trị của bộ tham số tốt nhất định nghĩa cho một phân phối khác, sao cho phân phối này sẽ "gần" với phân phối mà ta quan tâm.

Với phương pháp VI, đầu tiên ta sẽ tìm một phân phối xác suất có cùng "họ" (family) với phân phối xác suất mà ta quan tâm. Một họ phân phối xác suất là tập các phân phối xác suất được định nghĩa bởi cùng một bộ tham số. Ví như họ phân phối Gaussian sẽ được định nghĩa bởi μ là giá trị kỳ vọng (mean) và σ là độ lệch chuẩn (standard deviation) Việc lựa chọn "họ" phân phối sẽ kiểm soát giữa độ phức tạp và độ chính xác của của phương pháp này. Nếu ta giả định rằng dữ liệu tuân theo một phân phối đơn giản thì kết quả suy diễn được sẽ không quá chính xác nhưng có thể dễ dàng tìm được nghiệm tối ưu. Ngược lại nếu ta lựa chọn "họ" phân phối phức tạp thì sẽ khó tìm đươc nghiệm tối ưu nhưng kết quả suy diễn sẽ có kết quả tốt hơn.

Sau khi xác định được "họ" phân phối xác suất dùng để xấp xỉ phân phối xác suất mà chúng ta quan tâm thì việc tiếp theo là làm sao để tìm ra được xấp xỉ tốt nhất.

Giả sử rằng chúng ta cần sấp xỉ phân phối p bởi một phân phối q cùng thuộc họ phân phối $\mathcal{F}$. Chúng ta xét độ lỗi $\mathbb{E}(q, p)$ giữa hai phân phối xác

suất p và q, việc tìm ra bộ tham số tốt nhất được thể hiện bởi:

$$q^* = \arg_{q \in \mathcal{F}} \min \mathbb{E}(q, p) \tag{2.5}$$

Vậy trong bài toán variational inference thì làm sao để xác định hai phân phối xác suất có "gần" nhau hay không hay làm sao để xác định độ lỗi $\mathbb{E}(q, p)$. Sự sai biệt Kullback-Leiber (KL) là một cách để tính mức độ lệch của một phân bố đối với một phân bố được chỉ định và thường được sử dụng để đo sự khác nhau giữa hai phân phối xác suất.

KL là môt thuật ngữ đến từ lĩnh lý thuyết thông tin, nó còn có tên gọi khác là entropy tương đối. Nói theo ngôn ngữ lý thuyết thông tin, nó đo lượng trung bình thông tin thêm vào nếu chúng ta mã hóa thông tin của phân bố q thay cho mã hóa thông tin phân bố p.

Nếu $p(x)$ và $q(x)$ là hai phân phối xác suất với x là biến ngẫu nhiên bất kỳ, thì sự sai biệt KL sẽ được định nghĩa như sau:

$$KL(q, p) = \mathbb{E}_q[\log q(x)] - \mathbb{E}_q[\log p(x)] \tag{2.6}$$

Sau khi xác định được một hàm lỗi để xấp xỉ phân phối xác suất mà chúng ta quan tâm p, bởi phân phối q thì bài toán sẽ trở thành việc tìm phân phối q^* như sau:

$$q^* = \arg_{q \in \mathcal{F}} \min \mathbb{E}_q[\log q(x)] - \mathbb{E}_q[\log p(x)]$$

Việc tìm ra phân phối xác suất tốt nhất này trở thành một bài toán tìm nghiệm tối ưu do đó phương pháp này có thể dễ dàng được áp dụng và mở rộng cho những trường hợp mà ta cần giải quyết một bài toán với quy mô dữ liệu lớn.

"Maximum Likelihood Estimation"

Trong lĩnh vực máy học, chúng ta sử dụng một mô hình để mô tả một tiến trình mà tổng hợp, phân tích tự động dữ liệu được thu thập để có thể

giải quyết các vấn đề như tìm ra các đặc trưng, đưa ra các dự đoán dựa trên dữ liệu quan sát được. Xét ví dụ bài toán *Hồi quy tuyến tính* ("Linear regression"), ta cần dự đoán giá trị y dựa trên giá trị của véc-tơ x theo công thức:

$$y = wx + b \tag{2.7}$$

Điều ta cần làm ở mô hình này là "ước lượng" giá trị tham số w và b để mô hình có thể dự đoán giá trị y một cách tốt nhất.

Với một mô hình máy học được mô tả bởi bộ tham số θ, ta cần thực hiện "ước lượng" bộ tham số θ sao cho mô hình có thể trả về kết quả tốt nhất. Thay vì dự đoán một hàm số nào đó có khả năng ước lượng tham số tốt, ta cần một nguyên tắc để suy ra các hàm số cụ thể cho các mô hình khác nhau. "Maximum likelihood Estimation" (MLE) là một công cụ phổ biến để thực hiện việc này.

Xét tập dữ liệu gồm m phần tử $\mathbb{X} = \{x_1, x_2, ..., x_m\}$ là độc lập với nhau và được phát sinh từ một phân phối xác suất $p_{data}(x)$. Giả sử mô hình được mô tả bởi bộ tham số θ. Khi đó, $p(x|\theta)$ là xác suất xảy ra sự kiện x khi ta biết θ. $p(x_1, x_2, ..., x_m|\theta)$ chính là xác suất các sự kiện $x_1, x_2, ..., x_m$ xảy ra đồng thời, xác suất đồng thời này được gọi là "likelihood". "Likelihood" của mô hình thể hiện khả năng mà bộ tham số của mô hình thể hiện mối quan hệ giữa dữ liệu mà ta có. Quá trình cực đại hóa "likelihood" là việc tối ưu khả năng mô hình thể hiện đúng nhất có thể mối quan hệ của dữ liệu. MLE là việc đi tìm bộ tham số θ sao cho likelihood là lớn nhất:

$$\theta = \max_{\theta} p(\mathbb{X}; \theta) \tag{2.8}$$

Vì các phần tử trong $\mathbb{X}$ là độc lập và cố định, do đó công thức 2.8 tương đương với:

$$\theta = \max_{\theta} \prod_{i=1}^{m} p(x_i; \theta) \tag{2.9}$$

Bài toán MLE là quá trình tối ưu công thức 2.9. Mà công thức này là

một tích, thường thì việc tối ưu hóa một tích sẽ gặp rất nhiều khó khăn trong việc tính toán. Thay vào đó, ta tối ưu hàm logarit của "likelihood" bởi vì:

- logarit là một hàm đồng biến, "likelihood" sẽ lớn nhất khi logarit của 'likelihood' là lớn nhất.
- logarit của một tích sẽ bằng tổng các logarit

Khi đó, bài toán MLE được đưa về bài toán "Maximum log-likelihood Estimation"

$$\theta = \max_{\theta} \sum_{i=1}^{m} \log(p(x_i; \theta)) \tag{2.10}$$

Gaussian likelihood ám chỉ "likelihood" của mô hình với việc phân phối xác suất $p_{data}(x)$ thuộc "họ" Gaussian. Tương tự, các hàm likelihood khác cũng thường dùng trong các bài toán máy học là: *Bernoulli likelihood, Multinomial likelihood.*

2.2.2 Mô hình "Variational Auto-Encoder"

Mô hình xác suất

Mô hình xác suất là một mô hình được dùng để mô tả một phân phối xác suất hợp của dữ liệu bằng cách sử dụng một đồ thị để mô tả các biến ngẫu nhiên tương tác với nhau trong phân phối xác suất. Ở đây chúng tôi sử dụng từ "đồ thị" là một định nghĩa về cấu trúc dữ liệu được mô tả trong lĩnh vực lý thuyết đồ thị. Đồ thị bao gồm các đỉnh được kết nối trực tiếp với nhau thông qua các cạnh. Vì cấu trúc của mô hình đươc mô tả bằng đồ thị cho nên những mô hình này còn được gọi với một tên gọi khác là "Graphical model". Một graphical model sẽ thể hiện phân phối hợp như sau:

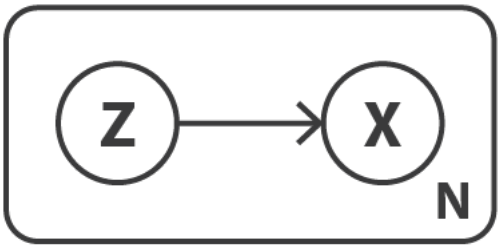
$$p(x_1, x_2, .., x_N)$$

trong đó $[x_1, x_2, .., x_N]$ là các đặc trưng dữ liệu, hoặc các tham số của mô hình,... Ví dụ như với bài toán phân loại thì một graphical model sẽ thể hiện cho phân phối $p(y, x, \theta)$ trong đó x là đặc trưng đầu vào, y là nhãn của dữ liệu và θ là trọng số của mô hình.

Graphical model cũng chính là một nhánh trong học máy, bằng cách thể hiện bài toán học máy dưới dạng một đồ thị. Sự kết hợp giữa xác suất vào một mô hình học máy sẽ giúp mô hình "giải thích" vấn đề thực tế một cách tốt hơn. Theo đó thì vấn đề cần quan tâm hoặc dữ liệu liên quan bởi các biến ngẫu nhiên. Và các biến ngẫu nhiên này chính là các đỉnh trong đồ thị và mối quan hệ giữa các đỉnh sẽ hình thành cạnh. Bằng cách thể hiện mối quan hệ giữa các biến dữ liệu bởi đồ thị ta dựa vào đó để có thể giải quyết các vấn đề mà chúng ta quan tâm. Ngoài ra bằng cách thể hiện bằng đồ thị, ta cũng thể hiện được tương quan giữa các biến dữ liệu, mà đa số các thuật toán máy học hiện nay thì tính tương quan giữa dữ liệu sẽ ảnh hưởng đến kết quả của mô hình. Dữ liệu có tính tương quan càng cao thì có thể sẽ ít đóng góp thêm "thông tin" cho mô hình thì có thể dẫn đến việc kết quả mô hình mang lại sẽ không cao. Do đó khi xây dựng thuật toán hay mô hình cho một vấn đề cụ thể, ta có thể áp dụng các kiến thức ta biết trước về lĩnh vực đó, về dữ liệu để có thể xác định các đặc trưng cho mô hình thông qua đồ thị. Ngoài ra việc sử dụng graphical cũng sẽ cung cấp một cái nhìn tổng quan về mô hình cũng như dữ liệu, từ đó việc phân tích, thiết kế và cài đặt cũng sẽ dễ dàng hơn.

Variational Auto-Encoder dưới góc nhìn xác suất

Với góc nhìn của một mô hình mạng nơ-ron thì "Variational Auto-Encoder" chỉ là một mạng nơ ron đơn giản với cấu trúc hai phần như mô hình "Auto-Encoder" tổng quát, gồm encoder và decoder. Encoder được dùng để trích xuất đặc trưng ẩn từ dữ liệu, điểm khác biệt so với "Auto-Encoder" cơ bản là encoder của VAE sẽ là một phân phối xác suất. Và tương tự thì decoder sẽ là mạng nơ-ron cố gắng để tái tạo lại dữ liệu ban đầu từ đặc trưng ẩn. Hình 2.3 thể hiện kiến trúc cơ bản của mô hình



Hình 2.5: Graphical model thể hiện cho mô hình "Variational Auto-Encoder". Dữ liệu quan sát được sẽ được giả định được phát sinh từ biến ẩn z

variational Auto-Encoder.

Tuy nhiên để hiểu rõ hơn về nền tảng toán học cũng như xác suất trong mô hình chúng tôi sẽ trình bày mô hình VAE dưới góc độ là một mô hình xác suất. "Variational Auto-enncoder" là một mô hình đồ thị có hướng mô tả mối quan hệ giữa dữ liệu quan sát được và đặc trưng ẩn. Một đồ thị có hướng là một graphical model mà các đỉnh được kết nối với nhau có thứ tự. Có nghĩa là để có học được mẫu ở nút "cha" thì trước đó ta cần mô hình hoá được dữ liệu ở nút con. Bên cạnh đó thì một graphical có hướng còn có tên gọi khác là "Bayesian network".

Xét graphical model thể hiện cho mô hình VAE trong hình 2.5: "Variational Auto-Encoder" bao gồm một biến x thể hiện cho dữ liệu, đây là biến dữ liệu quan sát được, và z là biến ẩn thể hiện cho đặc trưng ẩn của dữ liệu. Là một đồ thị có hướng do đó quá trình phát sinh dữ liệu của VAE được thực hiện qua các bước theo thứ tự như sau:

Với mỗi điểm dữ liệu:

- Đặc trưng ẩn z_i được lấy mẫu từ phân phối p(z)
- Điểm dữ liệu x_i được lấy mẫu từ phân phối p(x|z)

Cụ thể, biến đặc trưng ẩn z được chọn ra từ một phân phối "prior" $p(z)$ chính là những kiến thức ta biết trước ta biết trước hoặc là giả định của

chúng ta về z. Điểm dữ liệu x có một phân phối likelihood $p(x|z)$ thể hiện quan hệ giữa dữ liệu ta có với đặc trưng ẩn. Mô hình định nghĩa một phân phối hợp của dữ liệu và đặc trưng ẩn: $p(x, z)$. Với quy tắc nhân trong xác suất, chúng ta có thể phân tách phân phối hợp trên thành "prior" và "likelihood" như sau: $p(x, z) = p(x|z)p(z)$. Đây chính là mục tiêu chính khi ta xét "Variational Auto-Encoder" dưới góc độ của xác suất.

Mô hình này sử dụng phương pháp Variational inference được nhắc đến ở phần 2.2.1 để tìm ra đặc trưng ẩn, đây cũng là lý do dẫn đến cái tên "Variational Auto-Encoder". Và VAE có thể được huấn luyện dựa trên các thuật toán học dựa trên gradient truyền thống.

Tiếp theo, ta xét đến việc suy diễn (inference) trong mô hình này. Mục tiêu của việc suy diễn là tìm ra được một giá trị "tốt" thể hiện cho đặc trưng ẩn khi ta có các điểm dữ liệu, hay nói cách khác là ta tính "posterior". Theo công thức Bayes:

$$p(a|b) = \frac{p(b|a)p(a)}{p(b)} \tag{2.11}$$

theo như những gì đã trình bày ở những phần 2.2.1, phần mẫu số của công thức 2.11 được gọi là "marginal likelihood". Và "marginal likelihood" sẽ không dễ dàng tính toán được một cách chính xác khi đặc trưng ẩn ở không gian có số chiều cao.

Do đó, phương pháp Vairational inference (VI) được áp dụng để xấp xỉ phân phối "posterior" $p(z|x)$ này. VI xấp xỉ posterior thông qua một phân phối xác suất có cùng họ phân phối $q_\lambda(z|x)$. Trong đó λ thể hiện cho bộ tham số định nghĩa cho phân phối q, ví dụ nếu q là họ phân phối Gaussian thì $\lambda = (\mu; \sigma^2)$.

Tiếp đến, "Kullback-Leiber Devergence" được sử dụng để xấp xỉ "posterior" với:

$$\mathbb{KL}(q_\lambda(z|x)||p(z|x)) = \mathbb{E}_q[\log q_\lambda(z)] - \mathbb{E}_q[\log p(z|x)] \tag{2.12}$$

trong đó các kỳ vọng được lấy theo $q_\lambda(z)$.

Biến đổi xác suất có điều kiện $p(z|x)$ ở công thức 2.12, ta có:

$$\mathbb{KL}(q_\lambda(z|x)||p(z|x)) = \mathbb{E}_q[\log q_\lambda[(z|x)] - E_q[logp(x,z)] + \log p(x) \quad (2.13)$$

Mục tiêu của chúng ta là tìm ra bộ tham số λ sao cho tối thiểu được sự sai biệt trên. với q^* là phân phối q lý tưởng để xấp sỉ posterior thì ta có:

$$q^*_\lambda(z|x) = \arg\min_\lambda KL(q_\lambda(z|x)||p(x)) \quad (2.14)$$

Tuy nhiên ta vẫn chưa có thể tính được trực tiếp bởi trong công thức thì vân còn xuất hiện $p(x)$, cho nên "marginial likelihood" vẫn không thể tính một cách trực tiếp.

Vì không thể tính một cách trực tiếp ta xét một hàm số thay thế khác như sau:

$$ELBO(q_\lambda) = \mathbb{E}[\log p(z,x)] - \mathbb{E}[\log q_\lambda(z)] \quad (2.15)$$

Biến đổi phân phối hợp $p(z,x)$ trong 2.15, ta lại có:

$$\begin{aligned} ELBO(q_\lambda) &= \mathbb{E}[\log p(z)] + \mathbb{E}[\log p(x|z)] - \mathbb{E}[\log q_\lambda(z)] \\ &= \mathbb{E}[\log p(x|z)] - \mathbb{KL}(q_\lambda(z)||p(z)) \end{aligned} \quad (2.16)$$

Công thức 2.16 được gọi là "evidence lower bound" (ELBO). ELBO là trừ của sai biệt KL cộng với một lượng $\log p(x)$, bởi $\log p(x)$ là hằng số theo $q_\lambda(z)$. Bây giờ, việc cực đại ELBO sẽ tương đương với việc tối thiểu độ sai biệt KL.

Bên cạnh đó, sau khi biến đổi mở rộng ELBO thì ta thấy rằng ELBO được tính từ 2 phần đó là kỳ vọng của likelihood và trừ KL giữa prior và phân phối sấp xỉ $q_\lambda(z)$. Với kỳ vọng của likelihood, nó thể hiện khả năng mô hình hoá dữ liệu còn sai biệt KL thì đảm bảo việc phân phối được xấp xỉ sẽ gần với prior của dữ liệu. Đây chính là sự đánh đổi thường gặp trong những bài toán Bayesian inference, đánh đổi giữa khả năng mô hình hoá

dữ liệu và việc đảm bảo phân phối được xấp sỉ hay cụ thể là đặc trưng ẩn sẽ gần với prior của chúng ta.

Sau khi nắm được lý thuyết nền tảng về xác suất của mô hình VAE, tiếp theo ta sẽ liên hệ với mạng nơ ron để thể hiện cho mô hình. Mục tiêu cuối cùng của chúng ta là tìm ra bộ tham số có thể xấp xỉ được posterior $p(z|x, \lambda)$ bằng $q_\theta(z|x, \lambda)$ thông qua một mạng nơ ron thường được gọi là "inference network" hay chính là encoder. Mạng này nhận đầu vào là dữ liệu x và trả về kết quả là bộ tham số λ thể hiện phân phối xác suất của đặc trưng ẩn. Bên cạnh đó, sẽ có một mạng nơ ron được gọi là "genetive network" hay còn được biết đến là decoder thể hiện cho likelihood $p_\phi(x|z)$ của mô hình. Tiếp theo đặc trưng ẩn sẽ được lấy mẫu từ phân phối xác suất trả ra từ encoder, đây chính là bước đầu tiên trong mô hình xác suất đã được nói ở trên. Dữ liệu đầu vào của decoder là đặc trưng ẩn được lấy mẫu dựa trên phân phối của đặc trưng ẩn, và decoder sẽ cố gắng tái tạo lại dữ liệu ban đầu từ đặc trưng ẩn z, đây chính là bước thứ hai trong mô hình xác suất.

Inference network và generative network sẽ có bộ tham số tương ứng là θ và ϕ. Thông thường bộ tham số là các trọng số và bias trong một mạng nơ ron thông thường. Mục tiêu của mô hình sẽ là cực đại hàm ELBO tương ứng như sau:

$$ELBO(\theta, \phi) = \mathbb{E}_{q_\theta}(z|x)[\log p_\phi(x|z)] - \mathbb{KL}(q_\theta(z)||p(z)) \tag{2.17}$$

Để huấn luyện mô hình, ta có thể sử dụng những thuật toán học như "gradient descent" để có thể tìm ra bộ trọng số θ, ϕ.

Chương 3

Xây dựng hệ thống gợi ý sản phẩm dựa trên mô hình "Auto-Encoder"

Chương này trình bày về cách áp dụng mô hình "Variational Auto-Encoder" để giải quyết bài toán xây dựng hệ thống gợi ý sản phẩm [7]; đây là cách giải quyết bài toán mà chúng tôi tập trung tìm hiểu trong khóa luận. Đầu tiên, chúng tôi phân tích hai loại dữ liệu phản hồi được dùng trong bài toán này từ người dùng là: "explicit feedback" và "implicit feedback" và lý do chúng tôi đặc biệt quan tâm xây dựng hệ thống gợi ý với dữ liệu "implicit feedback". Sau đó, chúng tôi trình bày về cách áp dụng mô hình "Auto-Encoder" để có thể xây dựng được một hệ thống gợi ý sản phẩm ở mức cơ bản. Cuối cùng, chúng tôi trình bày các phần tinh chỉnh để có được một mô hình gợi ý sản phẩm tốt hơn, bao gồm: áp dụng kỹ thuật "drop-out" khi huấn luyện mô hình, thay mô hình "Auto-Encoder" bằng một biến thể của nó là "Variational Auto-Encoder" và thay đổi hàm mục tiêu để phù hợp hơn với bài toán xây dựng hệ thống gợi ý.

3.1 Dữ liệu phản hồi của người dùng trong bài toán xây dựng hệ thống gợi ý sản phẩm

Như đã trình bày ở phần 1, để xây dựng một hệ thống gợi ý theo hướng tiếp cận "Collaborative filtering" ta chỉ cần dữ liệu là ma trận tương tác của người dùng. Tương tác ở đây có nghĩa là các phản hồi của người dùng dành cho sản phẩm, và các phản hồi này bao gồm hai loại:

- Phản hồi cụ thể ("explicit feedback")
- Phản hồi ngầm ("implicit feedback")

Trong phần này, chúng tôi sẽ làm rõ về tính chất của hai loại dữ liệu phản hồi cũng như ảnh hưởng của chúng đến hệ thống gợi ý.

3.1.1 Dữ liệu "explicit feedback"

Dữ liệu "explicit feedback" được hiểu là những phản hồi của khách hàng về sản phẩm một cách tường minh và cụ thể, ví dụ như: số điểm đánh giá, bình luận, ... "Explicit feedback" có thể thể hiện rõ về mức độ thích/không thích của người dùng về sản phẩm; ví dụ người dùng có thể thể hiện sự yêu thích của họ từ 1 đến 5 sao cho một sản phẩm (một cách đánh giá thông dụng), sản phẩm được đánh giá 5 sao chứng tỏ nó được thích hơn so với sản phẩm được đánh giá 4 sao. Trong thực tế, dữ liệu "explicit feedback" thường khó để thu thập cũng như gặp trở ngại về tính tin cậy. Thu thập loại dữ liệu này gặp khó khăn vì không phải người dùng nào cũng sẵn sàng phản hồi về sản phẩm. Sự miễn cưỡng của người dùng cũng như những tác động khi họ phản hồi có thể dẫn đến sự thiếu khách quan, làm sai lệch kết quả của hệ thống gợi ý. Thêm nữa, vì phản hồi của người dùng thể hiện mức độ thích/không thích của người dùng, mà người dùng thì chỉ tương tác với một lượng sản phẩm nhỏ trên toàn hệ thống, những sản phẩm còn

lại sẽ rơi vào trường hợp thiếu dữ liệu ("missing data"), gây khó khăn cho việc xử lí. Ngày nay, số lượng sản phẩm trong hệ thống là rất lớn, "explicit feedback" sẽ gặp khó khăn rất lớn khi có quá nhiều trường hợp thiếu dữ liệu, tác động đáng kể đến hiệu quả của hệ thống. Mặt khác, "collaborative filtering" sẽ có cơ sở đánh giá nhóm người dùng "tương đồng" với nhau một cách khắt khe hơn, giúp các gợi ý là những sản phẩm "tốt" hơn, tuy nhiên đôi lúc làm cho các gợi ý không được đa dạng.

3.1.2 Dữ liệu "implicit feedback"

Dữ liệu "implicit feedback" là dữ liệu được suy ra từ hành động của người dùng, nếu họ xem một bộ phim thì ta có thể hiểu là họ "thích" bộ phim đó. "Implicit feedback" cũng có thể được suy ra từ "tín hiệu ngầm" ("implicit signal"), xét ví dụ người dùng đánh giá một sản phẩm là 4 sao (trên thang đánh giá từ 1 đến 5 sao), từ "tín hiệu ngầm" dựa trên số sao họ đánh giá, ta có thể suy ra họ "thích" sản phẩm đó. "Implicit feedback" chỉ thể hiện rõ về sự "thích" cũng như chỉ thể hiện một cách tương đối mức độ yêu thích của người dùng. Cụ thể, người dùng không xem một bộ phim không có nghĩa là họ không thích bộ phim đó, có thể là họ chưa xem hoặc không biết nó có trên hệ thống. Cũng như họ xem một bài hát 10 lần chứng tỏ họ thích hơn so với một bài hát họ chỉ nghe 2 lần, và "implicit feedback" không thể thể hiện được rõ điều này. Trong thực tế, lượng dữ liệu phản hồi ngầm rất lớn và dễ dàng thu thập được, quá trình "phản hồi" của người dùng là bị động nên không bị ảnh hưởng bởi các yếu tố ngoại cảnh khác.

Ma trận tương tác của người dùng với dữ liệu phản hồi ngầm sẽ có dạng là một ma trận nhị phân, với giá trị **1** thể hiện người dùng "thích" sản phẩm đó, giá trị **0** thể hiện hệ thống chưa có cơ sở để xác định người dùng "thích" sản phẩm đó.

Với dữ liệu phản hồi ẩn, "collaborative filtering" sẽ xác định nhóm người dùng "tương đồng" với nhau rộng hơn do chỉ quan tâm đến các sản phẩm

họ thích. Điều này sẽ giúp các gợi ý của hệ thống đa dạng hơn, tuy nhiên các sản phẩm mà người dùng không thích cũng có thể sẽ được gợi ý.

Trong giới hạn của khóa luận này, chúng tôi chỉ tìm hiểu về một hệ thống gợi ý với dữ liệu phản hồi ngầm do tính khách quan cũng như giải quyết được các khó khăn của "explicit feedback".

3.2 Áp dụng mô hình "Auto-Encoder" để xây dựng hệ thống gợi ý sản phẩm ở mức cơ bản

Như đã trình bày ở chương 2, mô hình "Auto-Encoder" là một mạng nơ-ron thường được sử dụng trong tác vụ trích xuất đặc trưng ẩn thông qua phương pháp học không giám sát. Ở phần này, chúng tôi sẽ trình bày việc xây dựng một hệ thống gợi ý sản phẩm cơ bản dựa trên mô hình "Auto-Encoder".

3.2.1 Kiến trúc mô hình

Phương pháp xây dựng mô hình gợi ý sản phẩm mà chúng tôi tìm hiểu đó là sử dụng kiến trúc "Auto-Encoder", một mạng nơ-ron nhận input đầu vào là tương tác của người dùng trong lịch sử và mô hình được huấn luyện để tái tạo lại tương tác của người dùng. Mô hình sẽ đưa ra gợi ý các tương tác cho người dùng dựa trên những tương tác được tái tạo lại. Mô hình này là một mạng nơ-ron bao gồm hai thành phần:

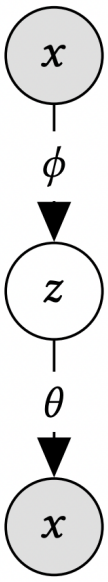
- Mạng nơ-ron encoder có chức năng rút trích đặc trưng ẩn từ những tương tác của người dùng trong lịch sử.
- Mạng nơ-ron decoder có chức năng tái tạo lại tương tác của người dùng từ đặc trưng ẩn.

Tập dữ liệu được sử dụng để huấn luyện bao gồm U người dùng $\mathcal{U} = [u_1, u_2, .., u_U]$ và mục tiêu là xây dựng mô hình đưa ra gợi ý trên tập I sản phẩm $\mathcal{I} = [i_1, i_2, ..., i_I]$. Bên cạnh đó, dữ liệu tương tác của các người dùng với các sản phẩm sẽ được thể hiện bởi một ma trận tương tác $X \in \mathbb{N}^{U \times I}$. Tương tác của một người dùng sẽ là một véc-tơ $x_u = [x_{u1}, x_{u2}, ..., x_{uI}] \in \mathbb{N}^I$ với $u \in \mathcal{U}$.

Với giả định rằng mỗi người dùng sẽ có tồn tại một đặc trưng ẩn nào đó, đặc trưng ẩn này là yếu tố sẽ quyết định đến việc tương tác của người dùng lên các sản phẩm. Do đó, bằng cách dựa vào lịch sử tương tác của người dùng ta sẽ cố gắng để có thể trích xuất được đặc trưng ẩn của họ. Mô hình sẽ nhận đầu vào x_u là dữ liệu tương tác của người dùng u, sau đó qua mạng encoder ta có được đặc trưng ẩn z_u. Từ đặc trưng ẩn z_u có được, mạng decoder sẽ cố gắng tái tạo lại dữ liệu tương tác ban đầu. Điều đó sẽ giúp ta có được một đặc trưng ẩn thể hiện được những yếu tố quyết định đến tương tác của người dùng. Hay nói cách khác, ta hy vọng đặc trưng ẩn được trích xuất sẽ thể hiện cho sở thích của người dùng.

"Auto-Encoder" là một mạng nơ-ron thường được sử dụng để trích xuất đặc trưng ẩn bằng phương pháp học không giám sát. Một cách đơn giản để huấn luyện mô hình "Auto-Encoder" cho bài toán xây dựng hệ thống gợi ý là ta sẽ cố gắng tái tạo lại tương tác của người dùng. Dựa vào kết quả tương tác được tái tạo để gợi ý tập sản phẩm "tốt" nhất mà người dùng chưa tương tác trước đó. Hoặc ta có thể xem như là một bài toán hồi quy, khi mà dữ liệu đầu vào là một con số thể hiện cho tương tác của người dùng. Kết quả của mô hình cũng là một véc tơ số "gần" với dữ liệu tương tác ban đầu. Để áp dụng hướng tiếp cận cơ bản này, chúng tôi sẽ trình bày về kiến trúc của một mô hình Auto-Encoder để xây dựng hệ thống gợi ý cơ bản.

Mô hình được cấu thành từ hai mạng nơ-ron tách biệt được gọi là encoder và decoder. Ta có thể xem rằng mạng nơ-ron encoder là một hàm phi tuyến ánh xạ dữ liệu đầu vào ở chiều không gian cao, $x_u \in R^I$ sang một không gian thấp hơn để biểu diễn cho đặc trưng ẩn $z_u \in R^k$ với k là



Hình 3.1: Minh họa kiến trúc mô hình Auto-Encoder cho bài toán gợi ý sản phẩm

số chiều của không gian đặc trưng ẩn. Thường thì k sẽ rất nhỏ so với I. Cụ thể thì ta có:

$$z_u = g_\phi(x_u) = g(W^T x_u + b_1) \tag{3.1}$$

trong đó $g(.)$ là hàm kích hoạt phi tuyến và $\phi = (W, b_1)$, với $W \in R^{I \times k}$, $b_1 \in R^k$ tương ứng sẽ là trọng số và hệ số "bias" của mạng nơ-ron encoder.

Sau khi có được đặc trưng ẩn, z_u sẽ được truyền thẳng qua mạng nơ-ron decoder nhằm mục đích xây dựng lại tương tác ban đầu từ z_u

$$\widehat{x}_u = f_\theta(z_u) = f(V^T z_u + b_2) \tag{3.2}$$

Tương tự với encoder, $\theta = (V, b_2)$, với $V \in R^{k \times I}$, $b_2 \in R^k$ sẽ tương ứng là các trọng số và "bias" của mạng nơ ron và $f(.)$ chính là hàm kích hoạt phi tuyến của mạng nơ-ron. Hình 3.1 minh họa cho kiến trúc Auto-Encoder cho bài toán gợi ý sản phẩm

3.2.2 Quá trình huấn luyện và đưa ra gợi ý

Để huấn luyện một mô hình "Auto-Encoder" cho bài toán xây dựng gợi ý sản phẩm, ta cần cực tiểu hóa hàm lỗi giữa tương tác được mô hình tái tạo lại và dữ liệu tương tác đầu vào sau:

$$\mathcal{L}(x;\phi,\theta) = \frac{1}{U}\sum_{x_u \in X}(||x_u - h_{\phi,\theta}(x_u)||_2^2) \tag{3.3}$$

trong đó $h_{\phi,\theta}(x_u)$ là tương tác được tái tạo lại với dữ liệu đầu vào là tương tác của người dùng ban đầu $u \in \mathcal{U}$ và $x_u \in R^I$.

$$h(x_u;\theta) = f(V^T z_u + b_2) = f(V^T g(W^T x_u + b_1) + b_2) \tag{3.4}$$

với $f(.)$ và $g(.)$ sẽ là các hàm kích hoạt phi tuyến của mạng nơ-ron.

Với hàm lỗi 3.3 mô hình hy vọng sẽ có thể tự động trích xuất được những đặc trưng ẩn có thể được dùng để hình thành nên các tương tác của người dùng. Mô hình sẽ được huấn luyện và tìm ra bộ tham số θ^* sao cho tối thiểu được độ lỗi 3.3:

$$\theta^* = \arg_\theta \min \mathcal{L}(x;\theta) \tag{3.5}$$

Tuy nhiên, để tránh tình trạng "overfitting" sẽ thường xảy ra khi huấn luyện một mạng nơ-ron phi tuyến, gây ảnh hưởng đến độ chính xác của mô hình. Chúng ta cần phải thêm một lượng "regularization" cho bộ tham số (ϕ,θ). "Regularization" nói một cách đơn giản là thay đổi mô hình một chút để tránh "overfitting" trong khi vẫn giữ được tính tổng quát của nó (tính tổng quát là tính mô tả được nhiều dữ liệu, trong cả tập dữ liệu huấn luyện và dữ liệu kiểm định). Cụ thể chúng tôi dùng "L2 regularization" cho mô hình. Vậy nên để huấn luyện mô hình "Auto-Encoder" với bài toán xây dựng hệ thống gợi ý sản phẩm ta sẽ huấn luyện mô hình để tìm được

bộ tham số cho mô hình như sau:

$$\theta^* = \arg_\theta \min \mathcal{L}(x;\theta) + \frac{\lambda}{2} \times (||W||_2^2 + ||V||2_2) \tag{3.6}$$

trong đó λ là hệ số "regularization".

Với kiến trúc trên, mô hình xây dựng cho hệ thống gợi ý với số lượng sản phẩm là I và đặc trưng ẩn với số chiều là k thì số lượng tham số của mô hình sẽ là $2Ik + I + k$.

Sau khi huấn luyện mô hình với hàm mục tiêu trên, ta tìm được được bô tham số θ^*, thì để dự đoán tương tác của người dùng u với sản phẩm i cụ thể sẽ là:

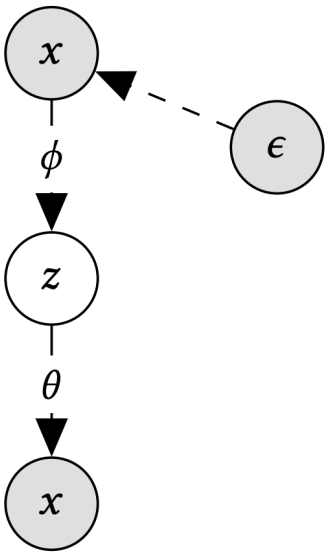
$$\widehat{x}_{ui} = h(x_{ui}, \theta^*)$$

Với tương tác được dự đoán, cụ thể là những tương tác lên những sản phẩm chưa được người dùng thực hiện trước đó, ta chọn ra những sản phẩm có kết quả trả về cao nhất để có thể đưa ra gợi ý cho người dùng.

3.2.3 Thay đổi "input" và "output" trong quá trình huấn luyện mô hình để phù hợp hơn với bài toán gợi ý sản phẩm

Như đã thảo luận ở phần 3.2.1 trên, chúng ta đã xây dựng một kiến trúc ở mức cơ bản nhất trong việc xây dựng một hệ thống gợi ý sản phẩm dựa trên mô hình "Auto-Encoder". Tuy nhiên, mô hình trên vẫn còn nhiều hạn chế:

- Thực tế thì mô hình vẫn đang làm tác vụ là tái tạo lại tương tác của người dùng chứ chưa thực sự được huấn luyện để dự đoán những sản phẩm mà người dùng sẽ tương tác dựa lịch sử tương tác của họ.
- Một hạn chế khác đó là tình trạng "overfiting" khi huấn luyện mạng nơ-ron. Với việc thêm hệ số "regularization" để huấn luyện nhưng với dữ liệu thưa thì việc chính quy hóa sẽ không thực sự hiệu quả.

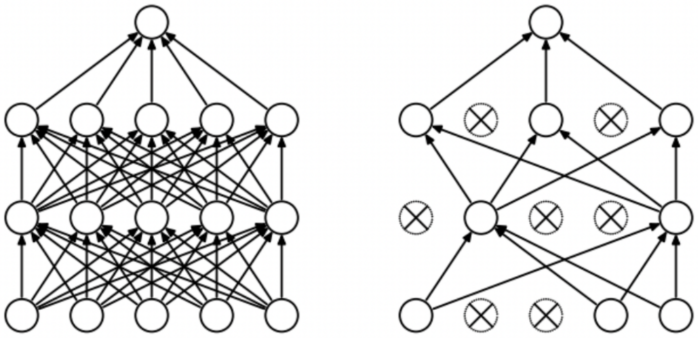


Hình 3.2: Minh họa kiến trúc mô hình "Denoising Auto-Encoder" cho bài toán gợi ý sản phẩm

Một phương pháp mà chúng tôi sử dụng để có thể giải quyết những vấn đề trên đó là chúng tôi "che" đi một số tương tác trước khi dữ liệu được truyền thẳng qua mạng encoder. Nói cách khác là chúng tôi áp dụng kỹ thuật "dropout" cho véc-tơ đầu vào, "dropout" là một phương pháp "regularization" thường được áp dụng khi huấn luyện các mạng học nơ-ron [1]. "Dropout" huấn luyện một tập các mạng con mà các mạng con này được định nghĩa bằng cách bỏ đi một số nơ-ron (không phải nơ-ron đầu ra) của mạng ban đầu. Trong thực tế, để bỏ đi một số nơ-ron ta chỉ nhân cho giá trị đầu ra của nó cho 0. "Dropout" sẽ ép mạng nơ-ron tìm ra được những đặc trưng quan trọng hơn, hay là tìm ra những sản phẩm ảnh hưởng lớn đến đặc trưng ẩn của người dùng. Hình 3.3 minh họa cho kĩ thuật "Dropout" khi huấn luyện mạng nơ-ron.

Mô hình "Auto-Encoder" với đầu vào được "che" đi trong quá trình huấn luyện chính là "Denoising Auto-Encoder" như đã trình bày trong phần 2.1.2.

Gọi $\tilde{x}$ sẽ là dữ liệu sau khi đã "che" đi một số các tương tác, $\tilde{x}$ sẽ được



Hình 3.3: Minh họa áp dụng "Dropout" cho mạng nơ-ron truyền thẳng. Hình a) minh họa một mạng nơ-ron truyền thẳng có hai tầng ẩn. Hình b) minh họa một mạng nơ-ron con được tạo thành từ việc áp dụng "Dropout" cho mạng nơ-ron ban đầu; lưu ý, ta không áp dụng "Dropout" cho tầng đầu ra

truyền thẳng qua mạng encoder để trích xuất được đặc trưng ẩn:

$$z_u = g(W^T \tilde{x}_u + b_1) \tag{3.7}$$

Với việc sử dụng thêm "nhiễu" vào dữ liệu tương tác ban đầu của người dùng sẽ mang lại cho mô hình "Auto-Encoder" phù hợp hơn và hiệu quả hơn khi huấn luyện với dữ liệu tương tác của người dùng. Nói cách khác đó là sau khi thực hiện thêm nhiễu, mô hình sẽ thực sự là dự đoán những tương tác bị che từ những tương tác có trước đó trong lịch sử của người dùng thay vì chỉ để tái tạo lại tương tác của người dùng trước đó.

Bên cạnh đó, "dropout" là một kỹ thuật khi huấn luyện mạng nơ-ron để ngăn tình trạng "overfitting" hiệu quả. Hình 3.2 thể hiện cho ý tưởng sử dụng kiến trúc Denosing Auto-Encoder cho bài toán gợi ý sản phẩm.

3.3 Tinh chỉnh cách áp dụng mô hình "Auto-Encoder" để có được hệ thống gợi ý sản phẩm hoạt động tốt hơn

Trong lĩnh vực trí tuệ nhân tạo, dữ liệu đóng vài trò cực kỳ quan trọng. Đặc biệt là lĩnh vực máy học, việc hiểu được dữ liệu sẽ góp phần không nhỏ đến việc xây dựng một mô hình hiệu quả. Đối với bài toán xây dựng hệ thống gợi ý sản phẩm, trong thực tế mỗi người dùng sẽ chỉ tương tác với một lượng nhỏ số lượng sản phẩm trên toàn bộ hệ thống. Điều này dẫn đến dữ liệu tương tác của người dùng sẽ là dữ liệu thưa. Với đặc điểm này, tác giả Liang trong bài báo [7] mà chúng tôi tìm hiểu trong khóa luận đã đề xuất việc sử dụng mô hình "Variational Auto-Encoder" (VAE), một biến thể đặc biệt của "Auto-Encoder" cơ bản để xây dựng hệ thống gợi ý sản phẩm. Bên cạnh đó, tác giả Liang cũng đã có một số tinh chỉnh về hàm chi phí giúp mô hình hoạt động tốt hơn.

Trong phần này chúng tôi sẽ trình bày về kiến trúc mô hình VAE cũng như là những cải thiện cho mô hình trong tác vụ gợi ý sản phẩm cho người dùng được đề xuất trong bài báo [7].

3.3.1 Thay "Auto-Encoder" bằng "Variational Auto-Encoder"

Dữ liệu tương tác trong hệ thống gợi ý sản phẩm thường là dữ liệu thưa, có nghĩa là các phần tử trong véc-tơ input đầu vào sẽ đa phần sẽ mang giá trị 0. Ngoài ra, mục tiêu của một hệ thống gợi ý sản phẩm chính là việc tăng thêm số lượng tương tác của người dùng lên hệ thống, đặc biệt là khi người dùng chưa tương tác nhiều. Do đó, trong trường hợp này nếu áp dụng các mô hình "Auto-Encoder" truyền thống sẽ dẫn đến tình trạng "overfitting", nghĩa là không sinh ra được gợi ý cho người dùng.

Dựa vào hạn chế này, bài báo đã đề xuất việc sử dụng mô hình "Vari-

ational Auto-Encoder". Đây là một mô hình mạng nơ-ron nhưng với nền tảng xác suất, cụ thể là phương pháp Variational Inference - một phương pháp suy diễn dữ liệu trong lĩnh vực xác suất thống kê. Như đã trình bày ở phần 2.2, đây là một biến thể đặc biệt của "Auto-Encoder" cơ bản, với việc đặc trưng ẩn được rút trích là một phân phối xác suất, mô hình đã mang lại ý nghĩa xác suất cho đặc trưng ẩn. Nói cách khác, đặc trưng giờ đây không chỉ thể hiện cho tương tác của người dùng, ngoài ra, là một phân phối xác suất thì đặc trưng ẩn còn có thể thể hiện được sự không chắc chắn ở trong đó, và sự không chắc chắn này có thể phần nào giúp giảm hiện tượng "overfitting".

Kiến trúc mô hình

Để áp dụng "Variational Auto-Encoder" (VAE) cho việc xây dựng một hệ thống gợi ý sản phẩm, ta sẽ xét VAE với 2 tiến trình riêng biệt.

Đầu tiên là tiến trình suy diễn, tiến trình này suy diễn đặc trưng ẩn của dữ liệu từ tương tác của người dùng. Có nghĩa là tìm ra phân phối bố của đặc trưng ẩn dựa trên dữ liệu tương tác của người dùng. Tiến trình này được thực hiện thông qua một mạng nơ-ron được gọi là encoder hay còn được gọi là mạng inference.

Tiếp theo là tiến tình phát sinh, tiến trình này có mục tiêu là phát sinh tương tác của ngời dùng từ đặc trưng ẩn. Cụ thể, từ phân phối có được từ tiến trình suy diễn, một đặc trưng ẩn sẽ được lấy mẫu từ phân phối trên. Đặc trưng ẩn này sẽ tiếp tục đường truyền thẳng thông qua một mạng nơ-ron được gọi là decoder hay còn được gọi là mạng generative. Kết quả trả về sẽ là 1 véc-tơ thể hiện cho tương tác của người dùng được phát sinh từ đặc trưng ẩn.

Encoder được dùng để xấp xỉ được "posterior" $p(z_u|x_u)$ thông qua phương pháp Variational Inference. Như đã trình bày ở phần 2.2.1, phương pháp này xấp xỉ "posterior" thông qua một phân phối q cùng "họ" phân phối và đơn giản hơn để tính toán. Giả định rằng $p(z_u) = \mathcal{N}(0, I)$ là một phân phối chuẩn, thể hiện phân phối xác suất trên không gian của đặc trưng ẩn.

Do đó, $p(z_u|x_u)$ và $q(z_u|x_u)$ sẽ thuộc cùng họ phân phối "Gaussian". Khi đó $q(z_u|x_u) = \mathcal{N}(\mu_u, \sigma_u^2)$, bằng cách tối thiểu $\mathbb{KL}(q(z_u||x_u)||p(z_u|x_u))$ tìm ra bộ tham số định nghĩa cho phân phối q sao cho $q(z_u|x_u)$ xấp xỉ được $p(z_u|x_u)$.

Tuy nhiên với phương pháp này, thì số lượng tham số sẽ tỉ lệ với số lượng người dùng và số lượng sản phẩm. Với số lượng tham số như vậy thì nó có thể sẽ dẫn đến các khó khăn khi xây dựng hệ thống gợi ý sản phẩm. Do đó ta sẽ sử dụng phương pháp "amortized inference", phương pháp này thay vì ta tìm trọng số mỗi người dùng độc lập với nhau thì các người dùng sẽ được chia sẻ chung bộ trọng số. Có nghĩa là phân phối đặc trưng ẩn của toàn bộ người dùng, $\mu_u \in R^k$ và $\sigma_u \in R^k$ sẽ được định nghĩa bởi mạng nơ-ron:

$$[\mu_u, \sigma_u] = g(x_u; \phi) \in R^{2k} \tag{3.8}$$

với k là số chiều của đặc trưng ẩn, [.] là phép nối 2 véc-tơ, $f(.)$ sẽ là mạng nơ-ron với hàm kích hoạt phi tuyến với bộ trọng số ϕ được học qua quá trình lan truyền ngược. Lúc này, encoder sẽ là một mạng nơ-ron định nghĩa cho phân phối:

$$q_\phi(z_u|x_u) = \mathcal{N}(\mu_\phi(x_u), \sigma_\phi^2(x_u)) \tag{3.9}$$

Với x_u là dữ liệu tương tác của người dùng, tương tự như kiến trúc mô hình đã được tình bày trước đó ở mục 3.2.1. Trước khi dữ liệu được truyền thẳng qua mạng encoder thì x_u sẽ được thêm nhiễu thông qua một tầng "drop-out". Sau khi drop-out ta có được $\tilde{x}_u$. Lúc này $\tilde{x_u}$ sẽ được truyền qua mạng encoder. Sau khi $\tilde{x}_u$ được truyền qua mạng encoder 3.9, thì ta có được z_u.

$$z_u \sim q_\phi(z_u|\tilde{x}_u) \tag{3.10}$$

Tương tự với "Auto-Encoder" thì z_u sẽ được truyền thẳng qua một mạng nơ-ron phi tuyến là decoder. Như đã trình bày ở phần 2.2.1 thì mô hình decoder sẽ định nghĩa cho likelihood của mô hình, có nghĩa là:

$$p_\theta(x_u|z_u) = f(z_u; \theta) \tag{3.11}$$

Kết hợp encoder 3.9 và decoder 3.11 lại ta sẽ có một kiến trúc cơ bản của mô hình "Variational Auto-Encoder" cho bài toán xây dựng hệ thống gợi ý sản phẩm.

Quá trình huấn luyện

Để huấn luyện một mô hình Variational Auto-Encoder thì ta sẽ có hàm mục tiêu là hàm evidence lower bound (ELBO) như sau:

$$\mathcal{L}(x_u; \phi, \theta) = E_{q_\phi(z_u|x_u)}[\log p_\theta(x_u|z_u)] - \mathbb{KL}(q_\phi(z_u|x_u)||p(z_u)) \tag{3.12}$$

trong đó $p_\theta(x_u|z_u)$ được thể hiện bởi một mạng nơ ron, qua công thức 3.11 và $q_\phi(z_u|x_u)$ là một mạng nơ-ron, được thể hiện ở công thức 3.9.

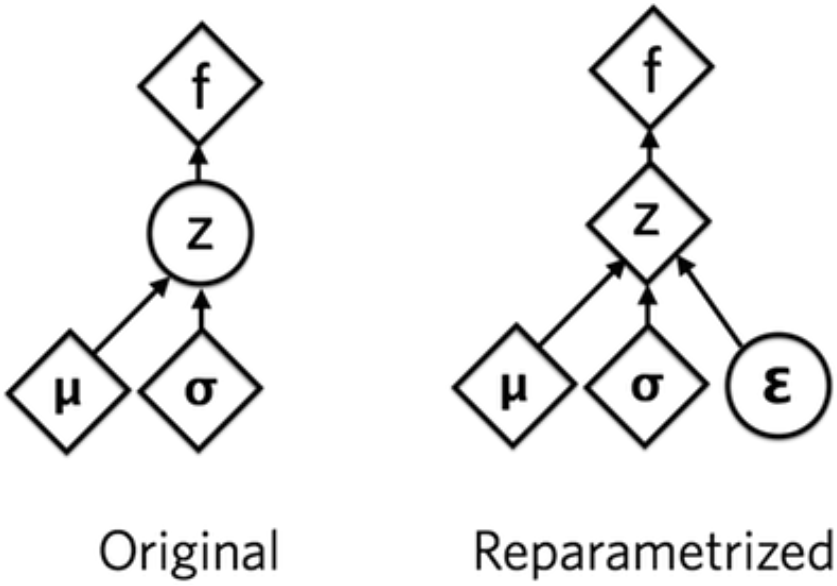
Để tìm được ϕ và θ ta sẽ thực hiện cực đại hóa, hay tối thiểu hóa giá trị âm của ELBO, sau đó thực hiện thuật toán lan truyền ngược để tìm được bộ trọng số cho mô hình.

Tuy nhiên, ta cần chú ý một điều là z_u ở công thức 3.10, là một đại lượng ngẫu nhiên được lấy mẫu từ một phân phối xác suất. Do vậy, giá z_u theo đó không còn phụ thuộc cứng vào ϕ. Mà mặc khác, khi lan truyền ngược để cập nhật trọng số, thì ta cần trọng số đó đóng góp trực tiếp vào giá trị của một nơ-ron. Điều này dẫn đến ta không thể cập nhật trọng số ϕ.

"Reparametrization trick" là một phương pháp được sử dụng trong mô hình Variational Auto-Encoder để có thể cập nhật trọng số ϕ mà z_u vẫn đảm bảo là được lấy mẫu từ phân phối xác suất $q_\phi(z_u|\tilde{x_u})$. Bằng cách ta lấy mẫu $\epsilon \sim \mathcal{N}(0, I_K)$ và khi đó ta tính z_u thông qua công thức:

$$z_u = \mu_\phi(\tilde{x}_u) + \epsilon \odot \sigma_\phi(\tilde{x}_u) \tag{3.13}$$

theo đó thì z_u thì vẫn là một giá trị được lấy mẫu ngẫu nhiên từ phân phối xác suất trả về từ encoder, nhưng z_u vẫn được tính thông qua phép tính với dấu bằng '='.



Hình 3.4: Minh họa "Reparametrization trick". Hình bên trái thể hiện cách bình thường, hình bên phải thể hiện Reparametrization trick. Reparametrization trick giúp ta loại bỏ tính ngẫu nhiên khi lấy mẫu dữ liệu từ phân phối xác suất. Nút hình thoi thể hiện biến cố định, nút hình tròn thể hiện cho biến ngẫu nhiên.

Dựa vào "Reparametrization trick" ta đã có thể cập nhật trọng số cho mô hình variational Auto-Encoder thông qua thuật toán lan truyền ngược. Hình 3.4 thể hiện cho ý tưởng của "Reparametrization trick".

Quá trình huấn luyện mô hình chúng ta sẽ tìm được bộ tham số (θ^*, ϕ^*) sao cho tối thiểu được hàm lỗi 3.12:

$$(\phi^*, \theta^*) = \arg_{\phi,\theta)}\min\mathcal{L}(x_u; \phi, \theta) \tag{3.14}$$

Quá trình đưa ra gợi ý sản phẩm cho người dùng

Sau quá trình huấn luyện mô hình, chúng ta có được bộ tham số tốt nhất với bộ dữ liệu đã được huấn luyện. Để có thể đưa ra được gợi ý cho người dùng, chúng ta đơn giản chỉ truyền thẳng dữ liệu qua mạng, với dữ liệu đầu vào tương tác của người dùng. Ở quá trình huấn luyện ta áp dụng kỹ thuật "Dropout" tuy nhiên ở quá đưa ra gợi ý thì chúng ta không áp dụng "Dropout" lên dữ liệu đầu vào. Khi dữ liệu tương tác của người

dùng được truyền qua mạng encoder, thì ta lấy kết quả trả về từ $\mu(x)$ để làm đặc trưng ẩn. Tiếp theo đó đặc trưng ẩn này truyền thẳng qua mạng decoder để có được tương tác của người dùng được dự đoán từ mô hình.

Ta gọi:

$$f_\theta(z_u) \equiv [f_{u1}, f_{u2}, \ldots, f_{uI}] \tag{3.15}$$

là kết quả trả về từ decoder, với đầu vào là đặc trưng ẩn $z_u = \mu x_u$. Từ kết quả được dự đoán, ta thực hiện sắp xếp lại kết quả (bỏ qua các tương tác trong lịch sử của người dùng). Tập sản phẩm được gợi ý sẽ là tập sản phẩm có kết quả tương tác trả về cao nhất.

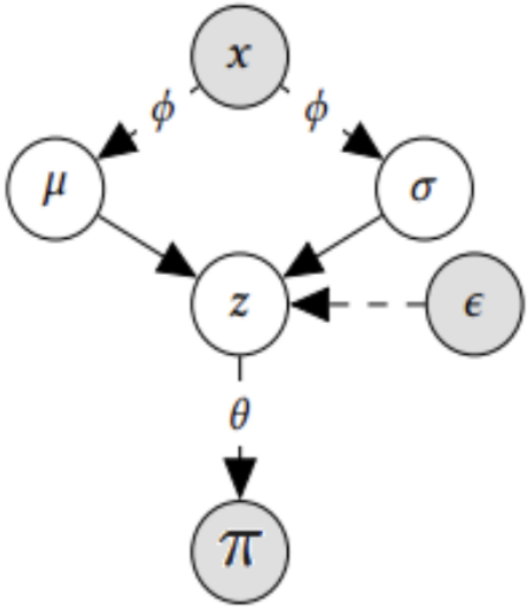
Lý do "Variational Auto-Encoder" phù hợp với bài toán xây dựng hệ thống gợi ý sản phẩm

"Variational Auto-Encoder" đã đạt được một số thành công nhất định trong bài toán xây dựng hệ thống gợi ý. Theo nghiên cứu của Dacrema [3], mô hình được đề xuất trong [7] đạt được các kết quả tốt trong việc gợi ý và xếp hạng.

Lý do mà chúng tôi cho rằng "Variational Auto-Encoder" phù hợp với bài toán xây dựng hệ thống gợi ý sản phẩm là vì:

- Đầu tiên, "Variational Auto-Encoder" có tính tổng quát hóa cao. Có nghĩa là chúng ta có thể có được những mô hình cơ bản hơn như "Matrix Factorization", hay "Auto-Encoder" cơ bản bằng cách sửa các thiết lập của mô hình.

- Bên cạnh đó, nền tảng của mô hình là "Variational Inference", là một phương pháp suy diễn dữ liệu hiệu quả. Dựa vào phương pháp này, ta có được đặc trưng ẩn là một phân phối xác suất. Mô hình này ước lượng phân bố xác xuất của véc-tơ đặc trưng ẩn thay vì ước lượng một véc-tơ đặc trưng ẩn cố định; nghĩa là có sự không chắc chắn ở trong đó, và sự không chắc chắn này có thể phần nào giúp chống overfitting.



Hình 3.5: Minh họa kiến trúc mô hình "Variational Auto-Encoder" cho bài toán gợi ý sản phẩm

- Cuối cùng, như trình bày ở mục 3.3.1 thì mô hình dựa trên "Amortized inference". Khi đó, mọi người dùng sẽ chia sẻ chung bộ tham số trong quá trình huấn luyện và đưa ra gợi ý. Điều này có liên hệ với ý tưởng của hướng tiếp cận "Collaborative filttering": đó là phân tích tương tác của các người dùng trong lịch sử và tìm ra những mẫu "pattern" chung giữa các người dùng để dựa vào đó đưa ra gợi ý cho người dùng khác.

3.3.2 Dùng hàm chi phí trong quá trình huấn luyện phù hợp hơn với bài toán gợi ý sản phẩm

"Multinomial likelihood"

Trong bất kỳ mô hình học máy nào, hàm lỗi luôn đóng một vai trò cực kỳ quan trọng trong việc quyết định đến độ hiệu quả của mô hình. Hàm lỗi sẽ định nghĩa mục tiêu mà mô hình cần đạt được thông qua việc ước

lượng sự khác biệt giữa giá trị được dự đoán của mô hình trả về so giá trị thực tế của nó. Ngoài ra, việc tìm được bộ tham số tốt nhất của mô hình cũng dựa cách tìm ra bộ tham số để tối thiểu hàm này. Do đó việc lựa chọn một hàm lỗi phù hợp sẽ ảnh hưởng ít nhiều đến kết quả của mô hình

Sau khi xác định được hàm lỗi thì ta sẽ có thể dễ dàng sử dụng phương pháp "Maximum Likelihood Estimation" như đã được trình bày trong phần 2.2.1.

Nhưng khác với "Auto-Encoder", hay cả "Variational Auto-Encoder" thông thường, tác giả trong bài báo [7] đã giả định rằng

$$x \sim \text{Mul}(N_u, \pi(z_u)) \tag{3.16}$$

có nghĩa rằng, x_u sẽ được phát sinh từ một phân phối xác suất, cụ thể là phân phối đa thức ("Multinomial Distribution"). Với $\pi(z_u)$ là một véc-tơ xác suất thể hiện cho xác suất được chọn của tập N_u sản phẩm với N_u là số lượng sản phẩm mà người dùng u đã tương tác. Ta gọi $f_\theta(z_u) \equiv [f_{u1}, f_{u2}, \ldots, f_{uI}]$ là kết quả trả về từ decoder, với đầu vào là đặc trưng ẩn z_u. Theo đó π_{ui} được tính như sau:

$$\pi_{ui} = \frac{\exp(f_{ui})}{\sum_j (\exp(f_{uj}))} \tag{3.17}$$

trong đó, $u \in \mathcal{U}$ và $i \in \mathcal{I}$, thể hiện cho xác suất người dùng u sẽ tương tác với sản phẩm i so với toàn bộ tập sản phẩm $\mathcal{I}$.

Với giả định này thì sau khi decoder nhận đầu vào z_u sẽ trả về véc-tơ $\pi(z_u)$ trên toàn bộ tập sản phẩm trong hệ thống. Hình 3.5 thể hiện kiến trúc mà tác giả đã sử dụng. Theo đó hàm lỗi được tác giả sử dụng trong trường hợp bài toán xây dựng hệ thống gợi ý sản phẩm là "Multinomial log-likelihood" (logarit của hàm "Multinomial likelihood" để thuận tiện cho việc tối ưu hóa được trình bày ở phần 2.2.1).

$$\log p_\theta(x_u | z_u) = \sum_i x_{ui} \log(\pi_i(z_u)) \tag{3.18}$$

Hàm lỗi này thì thường được sử dụng trong những mô hình ngôn ngữ như "Latent Dirichlet allocation", và kinh tế "Multinomial logit choice model". "Multinomial likelihood" thường được sử dụng với những bài toán liên quan đến dữ liệu chuỗi thời gian (time-series) hay dữ liệu dạng chuỗi (sequence) tuy nhiên lại ít được tập trung, nghiên cứu sử dụng trong lĩnh vực mô hình đặc trưng ẩn.

Ở mô hình này, thì tác giả trong bài báo [7] đã sử dụng hàm lỗi này cho bài toán gợi ý sản phẩm dựa trên giả định 3.16. Lý do mà tác giả đã sử dụng hàm này bởi vì "Multinomial log-likelihood" sẽ đặt giá trị thể hiện độ lớn xác suất lên tập sản phẩm được gợi ý mà mô hình trả về. Vì tổng độ lớn xác suất sẽ phải bằng 1, do đó các sản phẩm được mô hình gợi ý sẽ phải "cạnh tranh" với nhau để có được giá trị xác suất cao hơn. Do đó, mô hình sẽ cố gắng đạt được mục tiêu là các sản phẩm trả về sẽ được xếp hạng đúng theo mức độ phù hợp với người dùng.

Một góc nhìn khác của hàm ELBO

Theo hàm mục tiêu được định nghĩa ở công thức 3.12, ở một góc nhìn khác ta có thể phân tích 2 thành phần cấu thành ELBO với 2 mục tiêu khác nhau. Ở thành phần đầu tiên, sẽ thể hiện cho độ lỗi tái tạo lại tương tác của người dùng, hay là tập sản phẩm được gợi ý có đúng với nhu cầu người dùng hay không. Thành phần thứ hai, thì ta có thể xem như là một "regularization" cho hàm lỗi. Góc nhìn này, ta đã được thảo luận qua ở mục 2.2.2 thì đây cũng chính là việc đánh đổi giữa độ lỗi trong việc mô hình hóa dữ liệu và chuẩn hóa cho đặc trưng ẩn gần với phân phối chuẩn trong mô hình variational Auto-Encoder.

Với một bài toán phát sinh dữ liệu mới, thì đánh đổi này là cần thiết bởi nó sẽ đảm bảo cho dữ liệu mới được phát sinh được chuẩn hóa. Nhưng trong bài gợi ý sản phẩm thì ta quan tâm hơn việc mô hình hóa dữ liệu hơn là việc đảm bảo các tính chất xác suất của đặc trưng ẩn. Do đó tác giả đã đề xuất sử dụng một siêu tham số β để có thể dễ dàng kiểm soát được đánh đổi đã được nói ở trên. Siêu tham số là một hệ số không được

học từ mô hình mà được chỉ định trước cho mô hình để có thể huấn luyện.

Theo đó, hàm mục tiêu lúc này của mô hình sẽ là:

$$\mathcal{L}(x_u; \phi, \theta) = E_{q_\phi(z_u|x_u)}[\log p_\theta(x_u|z_u)] - \beta \times \mathbb{KL}(q_\phi(z_u|x_u)||p(z_u)) \quad (3.19)$$

Với đề xuất này, thì việc huấn luyện mô hình sẽ phải bao gồm thêm việc lựa chọn siêu tham số cho mô hình. Trong nhiều trường hợp thì việc lựa chọn siêu tham số là một bước mà tốn kém thời gian khi ta phải đánh giá mô hình trên nhiều giá trị siêu tham số khác nhau. Do đó tác giả cũng đề xuất thêm một phương pháp được gọi là "KL-annealing". Phương pháp này là một phương pháp "heuristic" để lựa chọn siêu tham số cho mô hình.

Ý tưởng của phương pháp là ban đầu ta sẽ khởi tạo giá trị cho $\beta = 0$. Và tăng dần giá trị β cho đến một giá trị cố định nào đó sau mỗi lần cập nhật trọng số của mô hình. Ở bài báo [7] thì tác giả tăng dần β từ 0 cho đến 1 thì dừng cập nhật cho β.

Lý do ta chỉ tăng β lên đến giá trị tối đa là 1 bởi vì ta quan tâm đến việc mô hình hóa nhiều hơn là việc phát sinh dữ liệu mới. Vậy nên, ban đầu β được gán giá trị rất nhỏ và được tăng dần, điều này sẽ giúp giai đoạn đầu mô hình sẽ cố gắng tái tạo lại dữ liệu tương tác của người dùng một cách hiệu quả nhất.

Khi huấn luyện mô hình với phương pháp này thì sau mỗi lần cập nhật mô hình thì ta sẽ đánh giá lại mô hình và lưu lại giá trị β sau mỗi lần mà mô hình đạt kết quả tốt hơn trên tập kiểm định. Tiếp theo đó thì để mô hình có thể đạt được một kết quả tốt hơn thì ta sẽ thực hiện huấn luyện lại mô hình một lần nữa với giá trị β tốt nhất đã được lưu lại trước đó. Ngoài ra, nếu chi phí tài nguyên không cho phép, ta chỉ có thể huấn luyện mô hình chỉ trong một lần thì ta có thể dừng cập nhật lại giá trị β sau khi mô hình có dấu hiệu giảm độ hiệu quả trên tập kiểm định.

Chương 4

Thí nghiệm

Trong chương này, chúng tôi trình bày các kết quả thí nghiệm nhằm đánh giá những nội dung trình bày ở chương 3. Bộ dữ liệu được dùng để tiến hành các thí nghiệm là bộ MovieLens-20M [4] và Million Song Dataset [2]; bên cạnh đó, độ đo được sử dụng để đánh giá khả năng đưa ra gợi ý của mô hình là Normalized Discounted Cumulative Gain (NDCG) và Recall. Kết quả thí nghiệm cho thấy mô hình do chúng tôi cài đặt đạt được kết quả xấp xỉ so với kết quả được công bố trong bài báo [7]. Bên cạnh đó, các kết quả thí nghiệm cũng chứng tỏ mô hình "Variational Auto-Encoder" có khả đưa ra các gợi ý sản phẩm phù hợp hơn so với mô hình "Auto-Encoder" cơ bản, đặc biệt là trong trường hợp dữ liệu tương tác của người dùng thưa. Ngoài ra, các kết quả thí nghiệm cũng cho thấy việc thay đổi hàm mục tiêu trong quá trình huấn luyện giúp mô hình đưa ra kết quả tốt hơn cho bài toán gợi ý sản phẩm.

4.1 Tập dữ liệu sử dụng

Chúng tôi tiến hành thí nghiệm trên các tập dữ liệu vừa và lớn ở các lĩnh vực khác nhau là MovieLens-20M [4] và Million Song Dataset (MSD) [2]; đây là các tập dữ liệu thường được dùng cho bài toán xây dựng hệ thống gợi ý với hướng tiếp cận "Collaborative Filtering".

	MovieLens	**MSD**
Số lượng người dùng	136,677	571,355
Số lượng sản phẩm	20,108	41,140
Số lượng tương tác	10.0M	33.6M
% tương tác	0.36	0.14
Số người dùng trong tập "held-out"	**10,000**	**50,000**

Bảng 4.1: Thống kê số lượng người dùng, số lượng sản phẩm, số lượng tương tác trong các tập dữ liệu

- Tập dữ liệu MovieLens-20M bao gồm dữ liệu đánh giá của 138,000 người dùng với khoảng 27,000 bộ phim, với 20 triệu đánh giá.
- Tập dữ liệu Million Song Dataset bao gồm dữ liệu tương tác (số lượt nghe) của khoảng 1 triệu người dùng và hơn 300,000 bài hát, với hơn 48 triệu tương tác.

Để có thể đánh giá kết quả đạt được so với kết quả trong bài báo [7] một cách tốt nhất, chúng tôi thực hiện các bước tiền xử lí được các tác giả mô tả trong bài báo [7].

- Tập dữ liệu MovieLens-20M: chúng tôi giữ lại những người dùng đã đánh giá ít nhất 5 bộ phim trở lên.
- Tập dữ liệu Million Song Dataset: chúng tôi chỉ giữ lại những bài hát được nghe bởi ít nhất 200 người dùng, và những người dùng nghe ít nhất 20 bài hát trong số còn lại.

Sau khi lọc dữ liệu theo điều kiện đã nói, số lượng người dùng, số lượng sản phẩm, số lượng tương tác, tỉ lệ tương tác giữa người dùng và sản phẩm được mô tả trong bảng 4.1.

Chúng tôi tiến hành "chuẩn hóa" dữ liệu về dạng "implicit feedback". Đối với tập dữ liệu MovieLens, chúng tôi chúng tôi chuyển dữ liệu thành dạng ma trận tương tác nhị phân, với các đánh giá của người dùng từ 4 trở lên (nghĩa là người dùng thích bộ phim đó) mang giá trị 1 trong ma trận

nhị phân. Với tập dữ liệu MSD, chúng tôi chuyển các giá trị tương tác của người dùng (số lượt nghe) thành giá trị nhị phân, có nghĩa là nếu người dùng đã nghe bài hát đó thì sẽ mang giá trị 1 trong ma trận nhị phân.

Với mỗi tập dữ liệu, chúng tôi lần lượt lấy ra một số lượng người dùng bằng nhau cho tập đánh giá (validation) và tập kiểm tra (testing), gọi chung đây là các tập "held-out". Số lượng người dùng cho mỗi tập "held-out" được thống kê lại trong 4.1. Phần còn lại sẽ là tập dữ liệu huấn luyện (training). Ngoài ra mỗi tập trên đều sẽ được chia làm 2 phần với mục đích tính toán độ đo để đánh giá mô hình. Cụ thể thì mỗi tập sẽ được chia thành hai phần "fold-in" và "fold-out". "fold-in" là phần sẽ là đầu vào của mô hình thể hiện cho dữ liệu tương tác trong qua khứ của người dùng. Phần còn lại dùng để đánh giá kết quả của mô hình. Tỉ lệ giữa "fold-in" và "fold-out" mặc định là 8:2 khi đánh giá mô hình.

4.2 Các thiết lập thí nghiệm

Để đánh giá khả năng tổng quát hóa của mô hình, chúng tôi chia dữ liệu thành 3 phần như đã trình bày trong phần 4.1, bao gồm các tập huấn luyện, đánh giá, kiểm tra. Với số người dùng của các tập "held-out" (tập dữ liệu đánh giá và kiểm tra) lần lượt là 10,000 và 50,000 cho hai bộ dữ liệu MovieLens và MSD.

Để huấn luyện mô hình, chúng tôi sử dụng toàn bộ dữ liệu tương tác của người dùng trong tập huấn luyện. Để đánh giá mô hình, chúng tôi dùng phần "fold-in" trong tập "held-out" làm đầu vào của mô hình. Sau đó đánh giá các sản phẩm được dự đoán bởi mô hình dựa trên tập "fold-out" với hai độ đo "Normalized Discounted Cumulative Gain" (NDCG) và "Recall" (chi tiết về độ đo đánh giá hệ thống gợi ý sản phẩm được trình bày trong phụ lục A.1).

Chúng tôi sử dụng một kiến trúc giống nhau cho hai mô hình "Auto-encoder" và "Variational Auto-encoder" (VAE). Với "encoder" và "decoder" là mạng nơ-ron nhiều lớp ("Multi-layer Perceptron" hay MLP) có kiến trúc

đối xứng nhau, kích thước véc-tơ biểu diễn ẩn của mô hình là 200. Kiến trúc của các mô hình trên với MLP có 1 tầng ẩn kích thước 600 có thể mô tả ngắn gọn là $[I \to 600 \to 200 \to 600 \to I]$, với I là số lượng sản phẩm. Áp dụng hàm kích hoạt phi tuyến là *tanh* giữa các lớp, tuy nhiên, do đầu ra của "encoder" trong mô hình VAE là một phân phối xác suất, do đó chúng tôi không áp dụng hàm kích hoạt tầng đó.

Ngoài ra, như đã trình bày ở phần 3.2.3, chúng tôi áp dụng kỹ thuật "dropout" cho véc-tơ đầu vào để tăng khả năng sinh ra gợi ý của mô hình cũng như tránh tình trạng "overfitting".

Như đề xuất của tác giả Liang trong bài báo [7], chúng tôi sử dụng "Multinomal log-likelihood" làm hàm tính độ lỗi cho cả hai mô hình. Chúng tôi gọi mô hình "Variational Auto-encoder" với hàm lỗi này là Mult-VAE, mô hình "Auto-encoder" với hàm lỗi này và phần "dropout" là Mult-DAE.

Chúng tôi điều chỉnh siêu tham số β của mô hình Mult-VAE với 200,000 bước cập nhật theo phương pháp "KL-annealing" được trình bày ở phần 3.3.2. Và áp dụng một lượng "weight-decay" 0.01 cho Mult-DAE.

Chúng tôi lần lượt huấn luyện hai mô hình Mult-VAE và Mult-DAE bằng thuật toán Adam [1] với hệ số học ("learning rate") là *0.001* trên các tập dữ liệu MovieLens [4] với 200 "epoch" và Million Song Dataset [2] với 100 "epoch" (mỗi "epoch" là một lần mô hình duyệt qua hết dữ liệu huấn luyện).

4.3 Các kết quả thí nghiệm

4.3.1 Kết quả cài đặt của khóa luận so với bài báo

Trong phần này, chúng tôi sẽ trình bày kết quả của mô hình mà chúng tôi cài đặt. Cụ thể, chúng tôi so sánh kết quả mô hình Mult-VAE và Mult-DAE chúng tôi cài đặt so với kết quả trong bài báo. Các mô hình được đánh giá trên tập kiểm tra gồm 10,000 người dùng trong tập dữ liệu MovieLens và 50,000 người dùng trong tập Million Song Dataset. Chúng

	Recall@20	**Recall@50**	**NDCG@100**
Mult-VAE	0.395	0.537	0.429
Mult-VAE (cài đặt của tác giả)	0.395	0.537	0.426
Mult-DAE	0.398	0.535	0.423
Mult-DAE (cài đặt của tác giả)	0.387	0.524	0.419

Bảng 4.2: Kết quả cài đặt trên tập MovieLens so với kết quả trong bài báo [7]

tôi dùng các độ đo giống với mô tả trong bài báo là:

- NDCG@100: độ đo NDCG trên tối đa 100 sản phẩm được mô hình xếp hạng cao nhất.
- Recall@50: độ đo Recall trên tối đa 50 sản phẩm được mô hình xếp hạng cao nhất.
- Recall@20: độ đo Recall trên tối đa 20 sản phẩm được mô hình xếp hạng cao nhất.

Từ kết quả trên bảng 4.2, ở tập dữ liệu MovieLens, cả hai mô hình chúng tôi cài đặt đều đạt được kết quả tương đồng so với kết quả trong bài báo [7].

Với các kết quả tập dữ liệu MSD được thể hiện trong bảng 4.3, cả hai mô hình chúng tôi cài đặt cho kết quả có sai lệch so với kết quả trong bài báo. Nhìn chung, kết quả của Mult-VAE và Mult-DAE ở cả 3 độ đo đều tương quan so với kết quả của bài báo.

Với mô hình Mult-VAE, cài đặt của tác giả Lobel và các cộng sự trong bài báo "RACT: Towards amortized ranking-critical trainning for collaborative filtering" [8], mô hình của tác giả cài đặt cũng cho kết quả tương đồng với mô hình chúng tôi cài đặt, với sai số nhỏ. Theo đó, kết quả trên tập dữ liệu MovieLens tương đồng với bài báo gốc [7] và kết quả trên tập dữ liệu MSD (được trình bày trong bảng 4.4) thấp hơn bài báo gốc, tuy nhiên sai số không quá lớn.

	Recall@20	Recall@50	NDCG@100
Mult-VAE	0.257	0.353	0.308
Mult-VAE (cài đặt của tác giả)	0.266	0.364	0.316
Mult-DAE	0.256	0.351	0.306
Mult-DAE (cài đặt của tác giả)	0.266	0.363	0.313

Bảng 4.3: Kết quả cài đặt trên tập MSD so với kết quả trong bài báo [7]

	Recall@20	Recall@50	NDCG@100
Mult-VAE	0.257	0.353	0.308
Mult-VAE (cài đặt của tác giả Lobel)	0.260	0.356	0.310
Mult-VAE (cài đặt của tác giả Liang)	0.266	0.364	0.316

Bảng 4.4: Kết quả cài đặt của khoá luận so với kết quả cài đặt trong bài báo của tác giả Lobel [8] và bài báo của tác giả Liang [7] trên tập dữ liệu MSD

Qua các kết quả trên, chúng tôi rút ra nhận xét như sau. Mult-VAE cho kết quả tốt hơn Mult-DAE ở cả 3 độ đo trên tập dữ liệu MovieLens. Với tập dữ liệu có kích thước lớn hơn là MSD, cả Mult-VAE và Mult-DAE đều cho kết quả khá gần nhau ở cả 3 độ đo, tương đồng với kết quả của hai mô hình này trong bài báo gốc [7].

4.3.2 Phân tích "Multinomial Likelihood"

Ở phần này, chúng tôi sẽ trình bày thí nghiệm để kiểm chứng hàm lỗi Multinomial log-likelihood so với các hàm lỗi thường được dùng khác. Cụ thể chúng tôi sẽ so sánh Multinomial log-likelihood so với Gaussian log-likelihood và Logistic log-likelihood.

Hàm "Multinomaial log-likelihood" như đã được trình bày ở phần 3.3.2 sẽ giả định dữ liệu ban đầu sẽ được phát sinh bởi một phân phối đa thức ("Multinomial Distribution"). Và kết quả trả về của mô hình Mult-VAE là

một véc-tơ thể hiện xác suất được chọn của từng sản phẩm trong hệ thống (véc-tơ này có tổng bằng 1).

Về Gaussian log-likelihood, cũng chính là độ lỗi "Mean Square Error", là một độ lỗi thường gặp trong những bài toán hồi quy (regression). Với độ lỗi này thì dữ liệu tương tác của người dùng được giả định sẽ tuân theo phân phối Gaussian. Theo đó, chúng tôi dựa vào bài báo [6] của tác giả Yifan Hu, đã áp dụng Gaussian log-likelihood cho bài toán gợi ý sản phẩm với dữ liệu "implicit feedback". Theo bài báo thì Gaussian log-likelihood cho người dùng u sẽ là:

$$\log p_\theta(x_u|z_u) = -\sum_i \frac{c_{ui}}{2}(x_{ui} - f_{ui})^2 \tag{4.1}$$

trong đó, c_{ui} là hệ số "confidence" của tương tác được thực hiện bởi người dùng u với sản phẩm i và f_{ui} là kết quả tái tạo lại tương tác của người dùng u với sản phẩm i. Hệ số "confidence" c_{ui} kiểm soát sự ảnh hưởng giữa những sản phẩm được tương tác với những sản phẩm không được tương tác. Cụ thể, những sản phẩm được tương tác sẽ có giá trị hệ số "confidence" cao hơn so với những sản phẩm không tương tác. Có nghĩa là $c_1 > c_0$, với c_1, c_0 lần lượt là hệ số "confidence" của sản phẩm được tương và sản phẩm không được tương tác.

Bên cạnh đó, Logistic log-likelihood, hay còn được biết đến là độ lỗi "Binary cross entropy", là một độ lỗi thường được dùng trong những bài toán phân loại (classification). Về lý thuyết xác suất, sử dụng mô hình này ta sẽ giả định rằng dữ liệu của chúng ta sẽ được phát sinh từ một phân phối Bernoulli. Với việc sử dụng, Logistic log-likelihood thì bài toán xây dựng hệ thống gợi ý sản phẩm hay cụ thể việc xác định tương tác của một người dùng với một sản phẩm sẽ được xem như bài toán phân loại 2 lớp. Logistic log-likelihood của người dùng u sẽ được tính như sau:

$$\log p_\theta(x_u|z_u) = -\sum_i x_{ui} \log \sigma(f_{xui}) + (1 - x_{ui}) \log(1 - \sigma(f_{ui})) \tag{4.2}$$

	Multinomial	**Gaussian**	**Logistics**
VAE	**0.429**	0.422	0.419
DAE	**0.423**	0.409	0.412

Bảng 4.5: Độ đo NDCG@100 trên các mô hình VAE và DAE với các hàm lỗi khác nhau trên tập dữ liệu MovieLens

Trong đó $\sigma(x) = \frac{1}{1+\exp(-x)}$ là hàm logistic và f_{ui} là kết quả tái tạo lại tương tác của người dùng u với sản phẩm i. Để thể hiện sự hiệu quả của multinomial log-likelihood so với hai hàm lỗi còn lại trong bài toán gợi ý sản phẩm với sản phầm phản hồi ngầm, chúng tôi lần lượng thực hiện huấn luyện mô hình với các độ lỗi trên và ghi lại kết quả tốt nhất. Trong thí nghiệm này việc huấn luyện mô hình chỉ thay đổi độ lỗi và các thiết lập khác sẽ được giữ nguyên. Cụ thể, chúng tôi tiến hành thay đổi hàm lỗi "Multinomial log-likelihood" thành các hàm lỗi "Gaussian likelihood" và "Logistic log-likelihood" trên hai mô hình "Variational Auto-encoder" và "Auto-encoder" với các thiết lập giống phần 4.2. Và chúng tôi thực hiện "fine-tune" siêu tham số tách biệt giữa các độ lỗi với nhau. Vì tính chất của "Multinomial log-likelihood" phục vụ mục đích xếp hạng các sản phẩm "phù hợp" với người dùng, do đó chúng tôi dùng độ đo Normalized Discounted Cumulative Gain (NDCG) để đánh giá chất lượng xếp hạng của mô hình.

Kết quả thực nghiệm được ghi nhận lại ở bảng 4.5 thể hiện độ đo NDCG@100 với các hàm lỗi và mô hình tương ứng. Từ kết quả cho ta thấy ảnh hưởng của "Multinomial log-likelihood" lên chất lượng xếp hạng của mô hình (đánh giá thông qua độ đo NDCG@100). Điều này chứng tỏ được rằng việc lựa chọn hàm lỗi thực sự phụ thuộc vào dữ liệu của mô hình. Và đối với bài toán gợi ý sản phẩm, cụ thể là với dữ liệu phản hồi ngầm thì multinomial log-likelihood mang lại kết quả ấn tượng. Từ đó cho thấy "Multinomial log-likelihood" là một hàm lỗi phù hợp với hệ thống gợi ý, khi chúng ta muốn "xếp hạng" các sản phẩm, từ đó đưa ra các gợi ý tốt nhất cho người dùng.

4.3.3 "Variational Auto-encoder" và "Auto-encoder" đối với trường hợp dữ liệu thưa

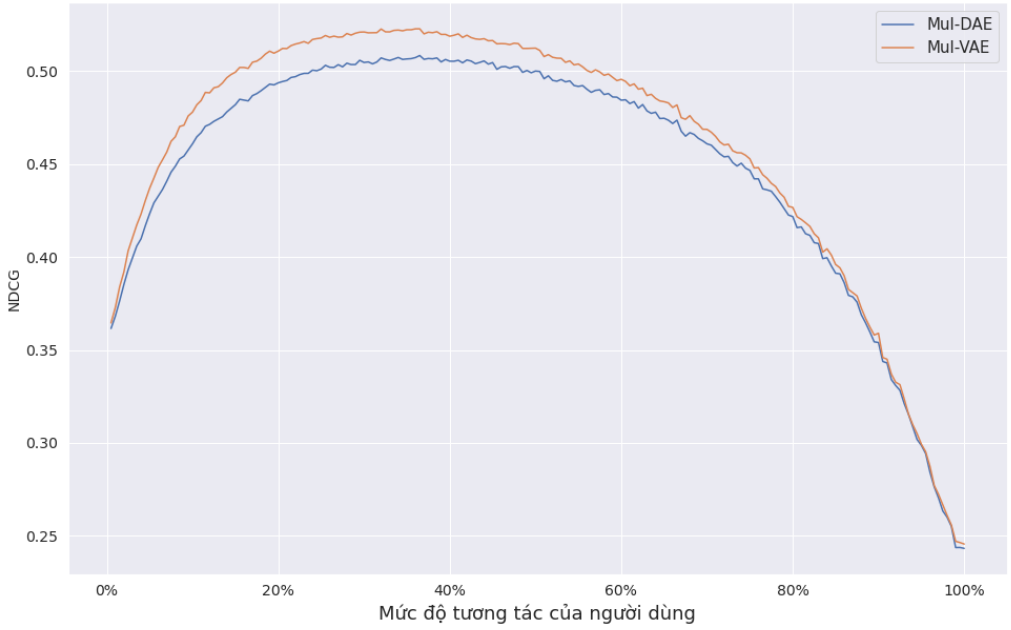
Trong phần này, mục tiêu của chúng tôi là kiểm chứng về sự phù hợp của "Variational Auto-encoder" so với "Auto-encoder", cụ thể là so sánh Mult-VAE và Mult-DAE. Theo góc nhìn khác thì ta có thể xem như đây là thí nghiệm so sánh giữa phương pháp "Bayesian inference" cho đặc trưng ẩn là một phân bố so với hướng tiếp cận xấp xỉ đặc trưng ẩn là một điểm xác định.

Như đã trình bày ở mục 3.3.1, thì mô hình Mult-VAE được xây dựng dựa trên các giả định của chúng tôi về dữ liệu, cụ thể là khi dữ liệu thưa. Do đó chúng tôi sẽ đánh giá hai mô hình Mult-VAE với tình trạng dữ liệu thưa.

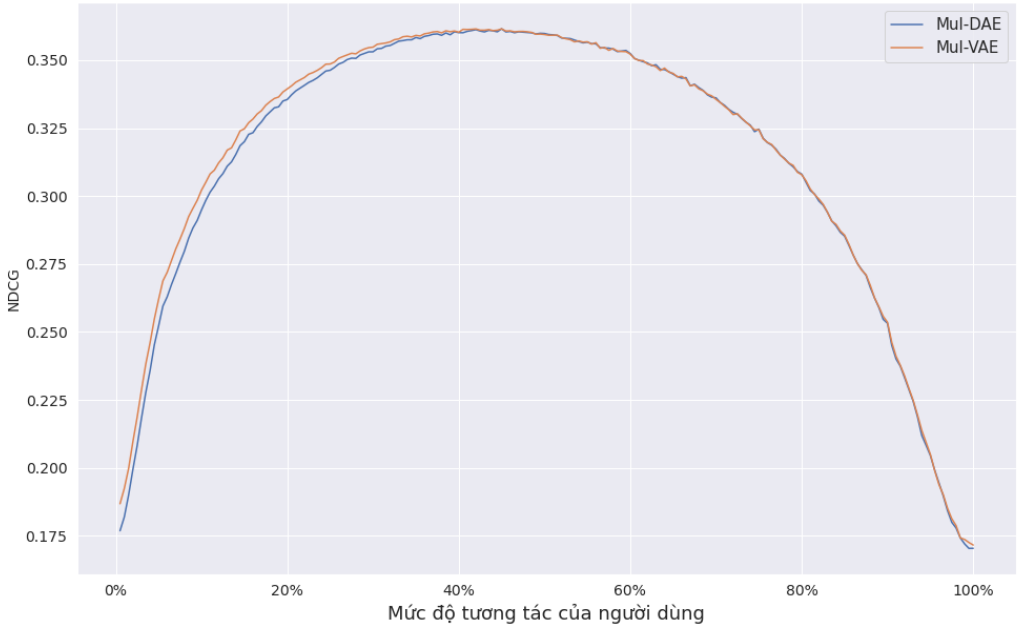
Theo thí nghiệm ở mục 4.3.1 thì ta có thể thấy được ở tập MovieLens thì cách biệt giữa Mult-VAE và Mult-DAE là lớn hơn so với tập MSD. Để kiểm chứng hiệu quả của Mult-VAE tốt hơn Mult-DAE ở điều kiện dữ liệu thưa, ở thí nghiệm này, cụ thể thì trong tập kiểm tra chúng tôi sẽ lần lượt thí hiện giảm số lượng tương tác người dùng trong tập "fold-in" (ở đánh giá trước các đánh giá dựa trên tập "fold-in" gồm 80% tương tác của người dùng). Và đánh giá kết quả của mô hình dựa trên số lượng tương tác còn lại.

Hình 4.1 thể hiện giá trị của độ đo NDCG@100 trên tập MovieLens. Với tập dữ liệu này, ta dễ dàng nhận thấy Mult-VAE vượt trội hơn so với Mult-DAE, đặc biệt là khi dữ liệu tương tác người dùng thấp. Cụ thể trong khoảng mức độ tương tác của người dùng từ 20% đến 40% thì Mult-DAE hoàn toàn bị bỏ xa bởi Mult-VAE. Hình 4.2 là kết quả thực hiện thí nghiệm trên tập MSD. Tập dữ liệu này cũng có xu hướng tương tự như trên tập MovieLens, tuy nhiên sự cách biệt giữa Mult-DAE và Mult-VAE không quá rõ ràng. Song, ta vẫn dễ dàng nhận thấy được rằng với mức độ tương tác thấp Mult-VAE vẫn cho kết quả nhỉnh hơn so với Mult-DAE.

Kết quả thí nghiệm giúp ta đã giúp khẳng định lại được độ hiệu quả



Hình 4.1: So sánh Mult-VAE và Mult-DAE ở các mức độ tương tác khác nhau trên tập kiểm tra trên tập MovieLens



Hình 4.2: So sánh Mult-VAE và Mult-DAE ở các mức độ tương tác khác nhau trên tập kiểm tra trên tập Million Song Datasets

của mô hình Mult-VAE được đề xuất trong bài báo [7]. Tuy trên tổng quan thì Mult-VAE không quá vượt trội so với Mult-DAE nhưng ở trường hợp dữ liệu tương tác người dùng ở mức độ thấp thì Mult-VAE mang lại kết quả cực kỳ ấn tượng.

Chương 5

Kết luận và hướng phát triển

5.1 Kết luận

5.2 Hướng phát triển

Tài liệu tham khảo

[1] Ian Goodfellow Yoshua Bengio and Aaron Courville. "Deep Learning". Book in preparation for MIT Press. 2016. URL: `httpwww.deeplearningbo org`.

[2] Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere. "The Million Song Dataset". In: *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*. 2011.

[3] Maurizio Ferrari Dacrema, Paolo Cremonesi, and Dietmar Jannach. "Are we really making much progress? A worrying analysis of recent neural recommendation approaches". In: *Proceedings of the 13th ACM Conference on Recommender Systems*. 2019, pp. 101–109.

[4] F Maxwell Harper and Joseph A Konstan. "The movielens datasets: History and context". In: *Acm transactions on interactive intelligent systems (tiis)* 5.4 (2015), pp. 1–19.

[5] Yifan Hu, Yehuda Koren, and Chris Volinsky. "Collaborative filtering for implicit feedback datasets". In: *2008 Eighth IEEE International Conference on Data Mining*. Ieee. 2008, pp. 263–272.

[6] Yifan Hu, Yehuda Koren, and Chris Volinsky. "Collaborative Filtering for Implicit Feedback Datasets". In: *2008 Eighth IEEE International Conference on Data Mining*. 2008, pp. 263–272.

[7] Dawen Liang, Rahul G Krishnan, Matthew D Hoffman, and Tony Jebara. "Variational autoencoders for collaborative filtering". In: *Proceedings of the 2018 world wide web conference*. 2018, pp. 689–698.

[8] Sam Lobel, Chunyuan Li, Jianfeng Gao, and Lawrence Carin. "RaCT: Toward Amortized Ranking-Critical Training For Collaborative Filtering". In: *International Conference on Learning Representations*. 2019.

[9] Ivy Pro School. *Movie Recommendations : How Netflix does it? | Data Science Workshop*. URL: `https://www.youtube.com/watch?v=7lPOL74eG5I` (visited on 06/17/2021).

[10] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. "Autorec: Autoencoders meet collaborative filtering". In: *Proceedings of the 24th international conference on World Wide Web*. 2015, pp. 111–112.

[11] Harald Steck. "Gaussian ranking by matrix factorization". In: *Proceedings of the 9th ACM Conference on Recommender Systems*. 2015, pp. 115–122.

[12] Yao Wu, Christopher DuBois, Alice X Zheng, and Martin Ester. "Collaborative denoising auto-encoders for top-n recommender systems". In: *Proceedings of the ninth ACM international conference on web search and data mining*. 2016, pp. 153–162.

Phụ lục A

PHỤC LỤC

A.1 Các độ đo đánh giá hệ thống gợi ý

A.1.1 Độ đo "Normalized Discounted Cumulative Gain"

Thông thường, các hệ thống gợi ý sản phẩm sẽ đánh giá một số điểm thể hiện mức độ phù hợp giữa sản phẩm được gợi ý và người dùng. Để đánh giá chất lượng của sản phẩm được gợi ý, độ đo "Cumulative Gain" được sử dụng. "Cumulative Gain" (CG) là tổng mức độ phù hợp của tất cả các kết quả trong danh sách kết quả tìm kiếm.

$$CG_p = \sum_{i=1}^{p} rel_i \tag{A.1}$$

Trong đó, rel_i là độ phù hợp của sản phẩm thứ i trong danh sách kết quả trả về của hệ thống, p là số sản phẩm được gợi ý.

Khi đánh giá bằng độ đo CG, CG không xem xét đến xếp hạng của các sản phẩm được gợi ý. Với ý tưởng các sản phẩm có mức độ phù hợp với người dùng cao khi được gợi ý sớm hơn sẽ hữu ích hơn. Khi đó, độ đo "Discounted Cumulative Gain" sẽ được sử dụng. "Discounted Cumulative Gain" (DCG) là một độ đo thường được sử dụng để đánh giá chất lượng của các thuật toán xếp hạng. Bằng các phạt các sản phẩm có mức độ phù

hợp cao nhưng được xếp hạng thấp khi gợi ý, DCG được tính bởi công thức:

$$DCG = \sum_{i=1}^{N} \frac{2^{rel_i} - 1}{\log_2(i+1)} \tag{A.2}$$

Trong đó, rel_i là mức độ phù hợp của sản phẩm với người dùng, N là số lượng sản phẩm được gợi ý.

Trong bài toán xây dựng hệ thống gợi ý, với mục tiêu đưa ra "top-k" sản phẩm thay vì đánh giá mức độ sản phẩm phù hợp với người dùng, độ đo DCG cho "top-k" sản phẩm sẽ được định nghĩa như sau:

$$DCG@k = \sum_{i=1}^{k} \frac{2^{\mathbb{I}[\omega(i) \in I_u]} - 1}{\log_2(i+1)} \tag{A.3}$$

Trong đó, $\omega(i)$ là sản phẩm được gợi ý thứ i, I_u là tập sản phẩm người dùng tương tác. $\mathbb{I}[.]$ là hàm chỉ thị ("indicator function"), hàm này sẽ trả về 1 nếu sản phẩm được gợi ý thứ i có trong tương tác của người dùng.

Độ đo DCG là một tổng, do đó khi k tăng sẽ dẫn đến DCG cũng tăng theo. Khi muốn so sánh hai hệ thống gợi ý, ta cần một độ đo được chuẩn hóa để thuận tiện cho việc so sánh. "Normalized Discounted Cumulative Gain" (NDCG) là độ đo DCG được chuẩn hóa, được tính bằng công thức:

$$NDCG = \frac{DCG}{IDCG} \tag{A.4}$$

Với IDCG là DCG trong trường hợp lý tưởng. Nghĩa là hàm $\mathbb{I}[.]$ trả về giá trị 1 tại mọi i.

NDCG sẽ có miền giá trị thuộc $[0, 1]$.

A.1.2 Độ đo "Recall"

Độ đo "Recall" cho k sản phẩm trong hệ thống gợi ý là độ đo đánh giá tỉ lệ sản phẩm được gợi ý phù hợp với người dùng trong k sản phẩm được

gợi ý. Độ đo "Recall" cho k sản phẩm được tính bằng công thức:

$$Recall@k = \frac{\sum_{i=1}^{k} \mathbb{I}[\omega(i) \in I_u]}{min(k, |I_u|)} \tag{A.5}$$

Trong đó, $\omega(i)$ là sản phẩm được gợi ý thứ i, I_u là tập sản phẩm người dùng tương tác. $\mathbb{I}[.]$ là hàm chỉ thị ("indicator function"), hàm này sẽ trả về 1 nếu sản phẩm được gợi ý thứ i có trong tương tác của người dùng. $|I_u|$ là số lượng sản phẩm người dùng tương tác.