

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

Lê Hoàng Phương Nhi - Lê Thị Như Quỳnh

MÔ HÌNH HỌC SÂU CHO BÀI TOÁN
HỌC CÓ GIÁM SÁT VỚI DỮ LIỆU
DẠNG BẢNG

KHÓA LUẬN TỐT NGHIỆP CỬ NHÂN
CHƯƠNG TRÌNH CHÍNH QUY

Tp. Hồ Chí Minh, tháng 06/2022

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

Lê Hoàng Phương Nhi - 18120496

Lê Thị Như Quỳnh - 18120530

**MÔ HÌNH HỌC SÂU CHO BÀI TOÁN
HỌC CÓ GIÁM SÁT VỚI DỮ LIỆU
DẠNG BẢNG**

KHÓA LUẬN TỐT NGHIỆP CỬ NHÂN
CHƯƠNG TRÌNH CHÍNH QUY

GIÁO VIÊN HƯỚNG DẪN

Thầy Trần Trung Kiên

Tp. Hồ Chí Minh, tháng 06/2022

Lời cảm ơn

Lời đầu tiên, chúng em xin trân trọng cảm ơn Thầy Trần Trung Kiên. Thầy đã rất tận tình giúp đỡ, hướng dẫn và chỉ bảo chúng em trong suốt quá trình thực hiện khóa luận. Những góp ý, định hướng cách tư duy và cách làm việc khoa học của thầy không chỉ giúp chúng em trong quá trình thực hiện khóa luận mà còn là hành trang tiếp bước cho chúng em trong quá trình học tập và làm việc sau này.

Chúng em xin chân thành cảm ơn các Thầy, Cô khoa Công nghệ thông tin - trường Đại học Khoa Học Tự Nhiên thành phố Hồ Chí Minh, những người đã tận tình giảng dạy cho chúng em trong quá trình học tập.

Cuối cùng, chúng em xin cảm ơn gia đình, người thân, bạn bè, những người đã luôn bên cạnh, giúp đỡ, ủng hộ, động viên chúng em trong học tập và cuộc sống.

TP.Hồ Chí Minh, 06/2022

Lê Hoàng Phương Nhi

Lê Thị Như Quỳnh

Mục lục

Lời cảm ơn	i
Tóm tắt	vi
1 Giới thiệu	1
2 Kiến thức nền tảng	8
2.1 Mạng nơ-ron sâu (Deep Neural Network)	8
2.1.1 Định nghĩa	8
2.1.2 “Hàm kích hoạt” (Activation Function)	10
2.1.3 “Gradient Descent”	13
2.2 “XGBoost”	16
2.2.1 Các đặc trưng của “XGBoost” (XGBoost Features)	17
2.2.2 Các thuật toán tìm kiếm sự phân tách (Split Finding Algorithms)	20
3 Mô hình “TabNet”	23
3.1 Kiến trúc chung	24
3.2 Attentive Transformer	26
3.3 Feature Transformer	28
3.4 Decision Steps	31
3.5 Khả năng diễn giải	34
3.6 Cách xử lý dữ liệu dạng categorical	35

4	Thí nghiệm	37
4.1	Tập dữ liệu sử dụng	37
4.2	Các thiết lập thí nghiệm	39
4.3	Các kết quả thí nghiệm	41
4.3.1	Kết quả cài đặt của khóa luận so với bài báo	41
4.3.2	“TabNet” so với “XGBoost”	42
4.3.3	Ảnh hưởng của các siêu tham số trong “TabNet” . .	50
5	Tổng kết và hướng phát triển	54
5.1	Tổng kết	54
5.2	Hướng phát triển	56
	TÀI LIỆU THAM KHẢO	57
A	Các độ đo đánh giá hiệu suất mô hình	59
A.1	Độ đo “Area Under the Receiving Op-erating Characteristic” (AUC)	59
A.2	Độ đo “Mean Squared Error” (MSE)	61
A.3	Độ đo “Root Mean Square” (RMSE)	61
B	Thí nghiệm thêm	63
B.1	Kết quả so với bài báo gốc [1]	63
B.2	Ảnh hưởng của các siêu tham số	64

Danh sách hình

1.1	Minh họa phương pháp học có giám sát giải quyết bài toán phân lớp, gồm có ba lớp (hình vuông, hình tròn, hình tam giác).	2
1.2	Minh họa lựa chọn đặc trưng của “TabNet” cho tập dữ liệu điều tra dân số và thu nhập (adult census income).	6
2.1	Hình mô phỏng một mạng nơ-ron nhân tạo đơn giản	9
2.2	Hình mô phỏng một mạng nơ-ron sâu với ba tầng ẩn	10
2.3	Một số hàm kích hoạt thông dụng (Nguồn: Wikipedia) . . .	11
2.4	Mô phỏng thuật toán “Gradient ” (Nguồn: rasbt.github.io)	14
3.1	Kiến trúc chung của “TabNet”	25
3.2	Khối Attentive Transformer	26
3.3	Cấu trúc khối Gated Linear Units (GLU).	30
3.4	Khối Feature Transformer	31
3.5	Cấu trúc Decision Step (Step 1)	33
3.6	Cấu trúc Decision Step (Step i)	34
4.1	Tầm quan trọng của các đặc trưng trên tập Syn2 (trái: “TabNet”, phải: “XGBoost”). Ở hai hình, trục tung thể hiện cho các đặc trưng từ X_1 đến X_{11} , trục hoành thể hiện cho mức độ quan trọng của đặc trưng.	44

4.2	Tầm quan trọng của các đặc trưng trên tập Syn4 (trái: “TabNet”, phải: “XGBoost”). Ở hai hình, trục tung thể hiện cho các đặc trưng từ X_1 đến X_{11} , trục hoành thể hiện cho mức độ quan trọng của đặc trưng.	45
4.3	Tầm quan trọng của các đặc trưng trên tập Rossmann Store Sales (trái: “TabNet”, phải: “XGBoost”). Ở hai hình, trục tung thể hiện cho các đặc trưng từ X_1 đến X_{11} , trục hoành thể hiện cho mức độ quan trọng của đặc trưng.	45
4.4	Tầm quan trọng trên cục bộ của các đặc trưng đối với tập Syn4 biểu diễn bằng “TabNet”. Hình thể hiện cho tầm quan trọng của 11 đặc trưng trên 15 mẫu đầu tiên của tập test. Trục tung biểu diễn cho 15 mẫu, trục hoành biểu diễn cho 11 đặc trưng.	46
4.5	Tầm quan trọng trên cục bộ của các đặc trưng đối với tập Syn4 biểu diễn bằng “XGBoost”. Hình thể hiện cho tầm quan trọng của 11 đặc trưng trên 15 mẫu đầu tiên của tập test. Trục tung biểu diễn cho 15 mẫu, trục hoành biểu diễn cho 11 đặc trưng.	47
A.1	Ví dụ đường cong ROC curve (Nguồn: scikit-learn.org) . .	60
B.1	Độ chính xác trên tập train và tập validation khi n_d rất nhỏ (màu xanh: train, màu cam: val)	65
B.2	Độ chính xác trên tập train và tập validation khi n_d rất lớn (màu xanh: train, màu cam: val)	65
B.3	Độ chính xác trên tập train và tập validation khi n_a rất nhỏ (màu xanh: train, màu cam: val)	66
B.4	Độ chính xác trên tập train và tập validation khi n_a rất lớn (màu xanh: train, màu cam: val)	66

Danh sách bảng

4.1	Kết quả cài đặt trên tập Synthetic so với kết quả trong bài báo [1]	41
4.2	Kết quả cài đặt trên tập dữ liệu Rossmann Store Sales so với kết quả trong bài báo [1]	42
4.3	Kết quả trên tập Synthetic của “TabNet” và “XGBoost” . .	42
4.4	Kết quả trên tập Sarcos của “TabNet” và “XGBoost” . . .	48
4.5	Thời gian chạy trên tập Sarcos của “TabNet” và “XGBoost” khi thay đổi số lượng output	48
4.6	Kết quả trên tập Rossmann Store Sales của “TabNet” và “XGBoost”	49
4.7	Kết quả AUC và loss ở epoch 199 trên tập train và validation của Syn1 đối với các giá trị λ_{sparse} khác nhau	51
4.8	Trung bình số lượng đặc trưng được chọn ở mỗi bước trên tập train của Syn1 đối với các giá trị λ_{sparse} khác nhau . .	52
4.9	Kết quả AUC và loss ở epoch 199 trên tập train và validation của Syn1 đối với các giá trị n_d và n_a khác nhau	53
B.1	Kết quả cài đặt trên tập dữ liệu Poker Hand so với kết quả trong bài báo [1]	64

Tóm tắt

Hiện nay, sự phát triển của công nghệ kéo theo đó là sự bùng nổ dữ liệu trên Internet. Các nhà nghiên cứu nhận thấy được những tiềm năng trong việc khai thác dữ liệu và tìm hiểu dữ liệu, điều này sẽ mang lại rất nhiều giá trị cho các tổ chức, doanh nghiệp. Vì vậy, việc nghiên cứu, tạo ra các thuật toán, mô hình để xử lý các dữ liệu đã và đang là một trong những vấn đề được cộng đồng nghiên cứu khoa học quan tâm.

Trên thực tế, dữ liệu có thể tồn tại ở nhiều dạng khác nhau như hình ảnh, âm thanh, văn bản hay dạng bảng, ... Và dạng dữ liệu phổ biến nhất hiện nay đó là dữ liệu dạng bảng. Đây cũng chính là lý do khóa luận tập trung tìm hiểu, nghiên cứu bài toán học có giám sát với dữ liệu dạng bảng.

Thêm vào đó, mô hình học sâu đã đạt được rất nhiều thành công đối với dữ liệu dạng hình ảnh, văn bản hay âm thanh, còn đối với dữ liệu dạng bảng thì chưa đạt được thành công đó. Tuy nhiên, các nhà nghiên cứu nhận ra được tiềm năng của việc sử dụng mô hình học sâu cho bài toán học có giám sát với dữ liệu dạng bảng. Điều này sẽ mang lại một số lợi ích nhất định, nổi bật trong số đó là giảm thiểu nhu cầu về “thiết kế đặc trưng thủ công” (feature engineering). Một bài báo đạt được hiệu quả cao trong việc sử dụng mô hình học sâu cho bài toán học có giám sát với dữ liệu dạng bảng là “TabNet: Attentive Interpretable Tabular Learning” [1] được giới thiệu bởi nhóm tác giả của Google tại hội nghị AAAI 2021, mô hình được bài báo đưa ra đó là “TabNet” - một kiến trúc học sâu mới dành cho dữ liệu dạng bảng. “TabNet” dựa trên “sự chú ý tuần tự” (sequential attention) để chọn các đặc trưng cần xử lý ở mỗi bước quyết định. Ưu

điểm của “TabNet” là: (1) mô hình cho phép hiểu về dự đoán mà mô hình đưa ra trên cả tổng thể và từng đầu vào cụ thể, (2) dễ dàng điều chỉnh, mở rộng kiến trúc mạng, (3) do dùng “Stochastic Gradient Descent” nên có thể hoạt động tốt với ngữ cảnh học “online” và học với dữ liệu lớn.

Kết quả đạt được của khóa luận là tìm hiểu và cài đặt lại được mô hình với một số kết quả tương đương bài báo gốc [1]. Khóa luận cũng tiến hành thêm một số thí nghiệm nhằm đánh giá rõ hơn về hiệu quả và tính chất của mô hình.

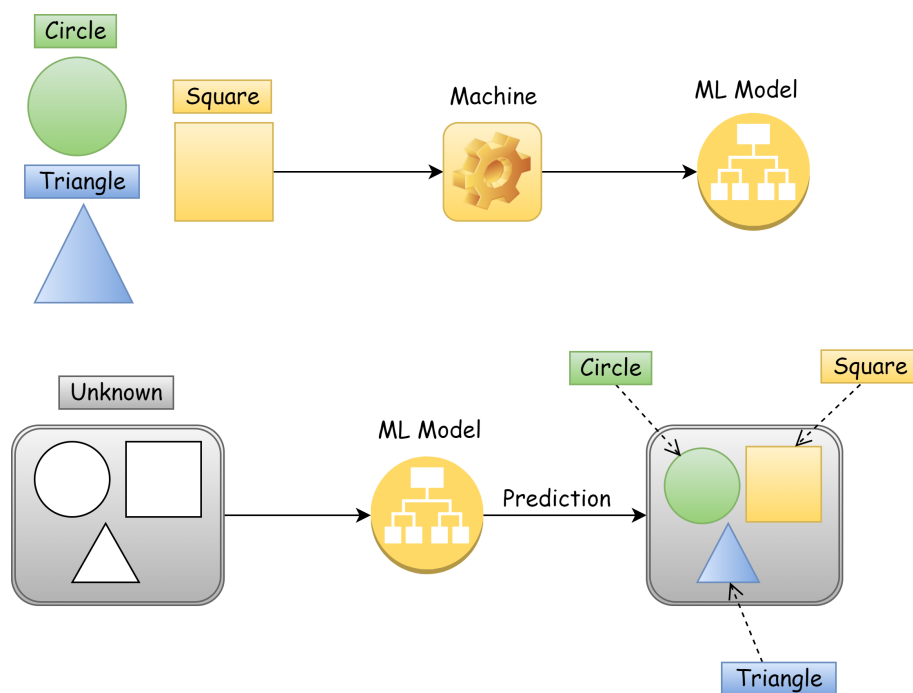
Chương 1

Giới thiệu

Hiện nay, sự phát triển của công nghệ kéo theo là sự bùng nổ dữ liệu. Theo thống kê, vào năm 2020, một người có thể tạo ra khoảng 1.7 MB dữ liệu trong một giây. Mỗi ngày có khoảng 306.4 triệu lá thư điện tử (email) được gửi đi, 5 triệu bài đăng được tạo ra trên twitter. Việc khai thác dữ liệu mang lại nhiều lợi ích cho doanh nghiệp, ví dụ cá nhân hóa trải nghiệm người dùng dựa theo lịch sử mua hàng, ... Chính vì vậy các nhà nghiên cứu tạo ra những kỹ thuật nhằm rút trích giá trị có trong dữ liệu.

Để có thể triển khai các kỹ thuật khai thác dữ liệu cần phải tổng hợp dữ liệu với số lượng đủ lớn. Nhiều công ty sử dụng nguồn dữ liệu được tạo ra trong quá trình làm việc của công ty. Nhiều công ty hay tổ chức khác sử dụng nguồn dữ liệu được thu thập từ internet. Dữ liệu sau khi được thu thập và làm sạch sẽ được đưa vào làm đầu vào cho mô hình học máy (machine learning). Trong lĩnh vực học máy, học có giám sát là một trong những phương pháp đạt nhiều thành công trong giới doanh nghiệp. Học có giám sát dùng để giải quyết nhiều bài toán trong nhiều lĩnh vực khác nhau; một số bài toán được giải quyết bởi mô hình học có giám sát như “nhận dạng ảnh và đối tượng” (image-and object-recognition), “phân tích dự đoán” (predictive analytics), “phân tích tâm lý khách hàng” (customer sentiment analysis), “phát hiện thư rác” (spam detection). Bài toán học có giám sát được phát biểu như sau (hình 1.1):

- Cho tập dữ liệu quan sát được (còn gọi là tập dữ liệu huấn luyện) gồm các cặp input và output đúng tương ứng. Ví dụ: {(ảnh thứ nhất, “ảnh hình vuông”), (ảnh thứ hai, “ảnh hình tròn”), (ảnh thứ ba, “ảnh hình tam giác”), ... }.
- Yêu cầu: tìm ra một mô hình (một công thức) để tính output từ input, sao cho mô hình này không chỉ đúng với dữ liệu quan sát được mà còn đúng với dữ liệu không quan sát được ở ngoài kia.



Hình 1.1: Minh họa phương pháp học có giám sát giải quyết bài toán phân lớp, gồm có ba lớp (hình vuông, hình tròn, hình tam giác).

Trong tập dữ liệu quan sát được thì các input có thể là các tấm ảnh, hoặc các đoạn âm thanh, hoặc các văn bản, hoặc các bộ gồm d thông tin về một loại đối tượng nào đó (ví dụ, input là bộ gồm 3 thông tin về sinh viên là MSSV, họ tên, điểm trung bình, ...). Trường hợp các input là các bộ gồm d thông tin (có khi còn gọi là thuộc tính hoặc đặc trưng) về một loại đối tượng nào đó là một trường hợp thường gặp. Dữ liệu dạng này còn được gọi là dữ liệu dạng bảng, vì ta có thể xem các input là các dòng

của bảng, và d thông tin là d cột của bảng. Trong khóa luận này, chúng em sẽ tập trung tìm hiểu bài toán học có giám sát cho dữ liệu dạng bảng.

Dữ liệu dạng bảng là kiểu dữ liệu chiếm phần lớn trong thế giới thực. Và nó có những thách thức lớn sau đây:

- **Dữ liệu bị nhiễu hoặc thiếu:** Dữ liệu dạng bảng thường được thu thập bằng cách thực hiện các phiếu khảo sát, điều này phụ thuộc hoàn toàn vào người tham gia thực hiện khảo sát. Và không ít trường hợp họ điền thông tin sai lệch hoặc có những thông tin cá nhân họ không muốn cung cấp. Điều này dẫn đến việc dữ liệu dạng bảng bị nhiễu hoặc thiếu.
- **Nhiều dữ liệu dạng categorical:** Như chúng ta đã biết, các mô hình học máy, đặc biệt là các mô hình học sâu, hoạt động tốt trên dữ liệu dạng số và liên tục. Trong khi đó, dữ liệu dạng bảng ngoài dữ liệu dạng số và liên tục còn có dạng categorical. Những đặc trưng dạng này cũng có thể đem lại rất nhiều giá trị, tuy nhiên để khai thác được điều đó là không hề dễ dàng.
- **Dữ liệu dạng categorical có nhiều phần tử riêng biệt:** Đối với dữ liệu dạng categorical cách xử lý thông thường đó là mã hóa “one-hot”, nhưng khi dữ liệu dạng categorical có quá nhiều phần tử riêng biệt thì phương pháp này có các hạn chế: làm tăng số lượng đặc trưng lên rất lớn khiến mô hình dễ bị “quá khớp” (overfitting) và quá trình huấn luyện có thể sẽ bị tràn RAM.

Các biến thể của cây quyết định (DT - Decision Tree) rất thành công với dữ liệu dạng bảng. Trên Kaggle, các biến thể của cây quyết định hiện đang là đề tài được thảo luận nhiều nhất. Đặc biệt “XGBoost” và “LightGBM” là hai mô hình cây quyết định tổng hợp đang thống trị hầu hết các cuộc thi máy học hiện nay. Có nhiều lý do giúp các mô hình cây quyết định tổng hợp thành công trên dữ liệu dạng bảng như vậy:

- Các phương pháp cây quyết định tổng hợp có khả năng diễn giải cục bộ và toàn cục.
- “XGBoost” và “LightGBM” được rất nhiều người quan tâm và trên các cuộc thi hai mô hình này gần như mang lại độ chính xác cao nhất cho tập thử nghiệm (test).
- Thực hiện quá trình huấn luyện nhanh.
- Có thể xử lý dữ liệu bị thiếu và dữ liệu dạng categorical.

Tuy có nhiều ưu điểm, nhưng cây quyết định tổng hợp vẫn tồn tại một số khó khăn. Trong những trường hợp dữ liệu có multi-output, ở mỗi bước “XGBoost” sẽ phải xây dựng nhiều cây, mỗi cây tương ứng với một output. Nhưng phương pháp học sâu có thể linh hoạt trong những trường hợp như vậy. Ý tưởng dùng mô hình học sâu cho dữ liệu dạng bảng là một hướng tiếp cận mới có nhiều tiềm năng. Hiện nay, mô hình học sâu đạt được nhiều thành công với dữ liệu dạng ảnh, văn bản và âm thanh [1]. Đối với dữ liệu dạng kể trên, mô hình học sâu có khả năng chuyển đổi dữ liệu từ dạng thô ban đầu sang dạng biểu diễn có ý nghĩa hơn. Còn đối với dữ liệu dạng bảng, phương pháp học sâu chưa thực sự hoạt động tốt bởi vì một số lý do như sau:

- Việc sử dụng phương pháp học sâu cho dữ liệu dạng bảng được cho là không tốt, khi mô hình học sâu cần phải tính toán một lượng lớn các trọng số. Đối với mô hình học sâu mà có nhiều tầng ẩn cần tốn nhiều thời gian để huấn luyện mô hình.
- Đầu vào của mô hình học sâu thường yêu cầu lượng lớn dữ liệu. Những tập dữ liệu dạng bảng có số lượng mẫu nhỏ khi đưa vào mô hình học sâu sẽ cho kết quả độ chính xác không cao.

Ngoài những thách thức thì mô hình học sâu có những ưu điểm có thể xem xét để sử dụng cho dữ liệu dạng bảng như sau:

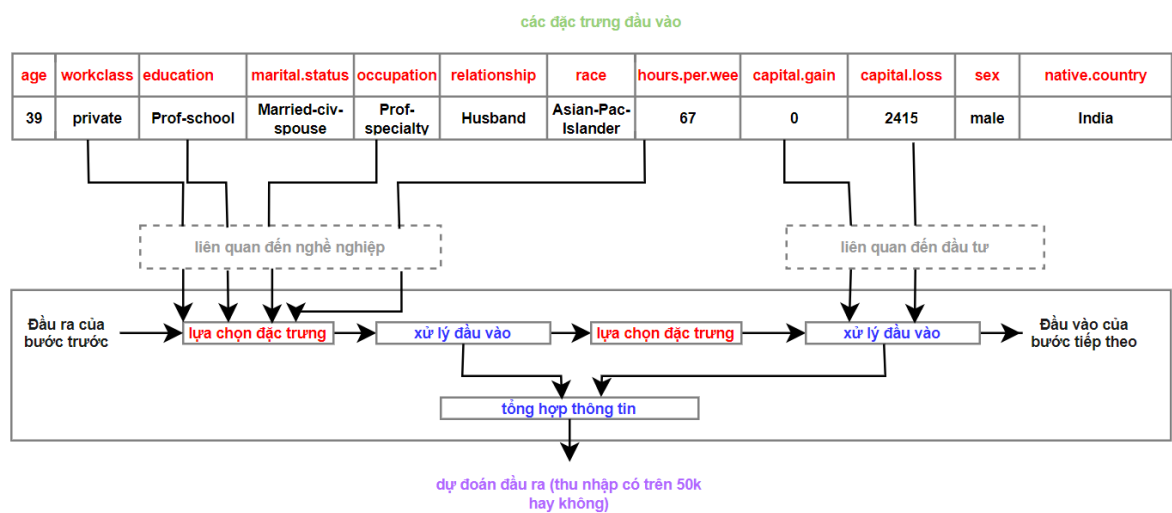
- Dễ dàng điều chỉnh/mở rộng kiến trúc mạng (ví dụ, để có thể dự đoán nhiều output thì chỉ cần thêm các nơ-ron ở tầng output).
- Trong học máy, thường thì dữ liệu sẽ được thu thập trong một thời gian dài để đủ lớn sau đó được đưa vào mô hình. Một điểm mạnh và quan trọng của mô hình học sâu đó là có thể học với dữ liệu trực tuyến (streaming) do dùng “Stochastic Gradient Descent”, tức là tại một thời điểm chỉ có một mẫu dữ liệu để cập nhật mô hình, và học với dữ liệu lớn khi chỉ có thể dùng một tập con để cập nhật mô hình do không đủ bộ nhớ để load toàn bộ tập dữ liệu.

Tại hội nghị “Proceedings of the AAAI Conference on Artificial Intelligence” năm 2021, nhóm tác giả Google đã công bố bài báo “TabNet: Attentive Interpretable Tabular Learning” [1]. Bài báo giới thiệu một kiến trúc học sâu mới - “TabNet”. “TabNet” có thể khắc phục một số hạn chế của mô hình học sâu cổ điển:

- Đầu vào của mô hình “TabNet” là dữ liệu thô chưa được qua tiền xử lý (preprocessing). “TabNet” sử dụng quy trình đầu cuối (end-to-end) để huấn luyện với “Stochastic Gradient Descent”.
- Trong cấu trúc của “TabNet” sẽ được chia thành các bước quyết định (decision step), ở mỗi bước quyết định “TabNet” sử dụng chú ý tuần tự (sequential attention) để chọn đặc trưng (features) cho mỗi bước quyết định (hình 1.2). Trong hình 1.2, đầu vào là tập dữ liệu điều tra dân số và thu nhập (adult census income). Ở mỗi bước quyết định, “TabNet” chỉ sẽ chọn một nhóm các đặc trưng ví dụ nhóm đặc trưng liên quan đến nghề nghiệp (workclass, education, occupation, hours.per.week) được chọn ở bước quyết định đầu và nhóm đặc trưng liên quan đến đầu tư (capital.gain, capital.loss) được chọn ở bước quyết định tiếp theo. “TabNet” sẽ chú trọng vào những đặc trưng quan trọng nhất giúp “TabNet” học tốt hơn. Phương pháp chọn đặc

trưng của “TabNet” là chọn các đặc trưng khác nhau đối với mỗi đầu vào (instance-wise).

- “TabNet” có thể phù hợp cho bài toán học phân lớp hoặc bài toán hồi quy. Ngoài ra, “TabNet” cho phép diễn giải. Cách thứ nhất là diễn giải cục bộ (local interpretability) bằng cách trực quan mức độ quan trọng của đặc trưng theo mẫu. Cách diễn giải thứ hai là diễn giải toàn cục (global interpretability) định lượng mức độ quan trọng của đặc trưng trên toàn bộ tập dữ liệu.



Hình 1.2: Minh họa lựa chọn đặc trưng của “TabNet” cho tập dữ liệu điều tra dân số và thu nhập (adult census income).

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

- Chương 2: Trình bày các kiến thức nền tảng về mô hình học sâu và các thuật toán cây quyết định tổng hợp - cụ thể là “XGBoost”.
- Chương 3: Trình bày về kiến thức của mô hình “TabNet”. Đây là phần chính của khóa luận.
- Chương 4: Trình bày các thí nghiệm và kết quả đạt được.

- Chương 5: Tổng kết và hướng phát triển.

Chương 2

Kiến thức nền tảng

Trong chương này, đầu tiên chúng em trình bày về mạng nơ-ron sâu (Deep neural network). Sau đó, chúng em trình bày về thuật toán “XGBoost” - một thuật toán học máy mạnh mẽ, dựa trên cây quyết định, đạt được rất nhiều thành công, chiến thắng nhiều cuộc thi trên Kaggle trong bài toán học có giám sát với dữ liệu dạng bảng. Chương này sẽ cung cấp những kiến thức nền tảng để có thể hiểu rõ hơn về mô hình mà chúng em tập trung tìm hiểu (sẽ được trình bày ở chương kế tiếp); cũng như những điểm nổi trội, khác biệt của mô hình so với các thuật toán học máy dựa trên cây quyết định.

2.1 Mạng nơ-ron sâu (Deep Neural Network)

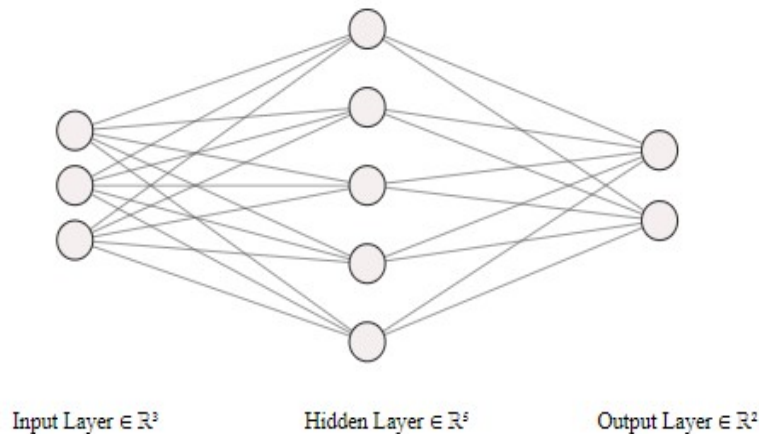
2.1.1 Định nghĩa

Mạng nơ-ron nhân tạo (neural network) (hình 2.1) là mạng sử dụng các mô hình toán học để xử lý thông tin, nó bắt chước cách thức hoạt động trong não bộ con người. Nó là nền tảng của học sâu (deep learning) - một lĩnh vực con của học máy. Các thành phần của mạng nơ-ron nhân tạo:

- Tầng vào (input layer): Tầng này nằm bên trái cùng của mạng, thể

hiện cho các đầu vào của mạng.

- Tầng ra (output layer): Tầng này nằm bên phải cùng của mạng, thể hiện cho các đầu ra của mạng.
- Tầng ẩn (hidden layer): Tầng này nằm ở giữa tầng vào và tầng ra, thể hiện cho quá trình suy luận logic của mạng. Đây là nơi các nơ-ron nhân tạo nhận tập hợp các đầu vào có trọng số và tạo ra đầu ra thông qua một hàm kích hoạt (chi tiết về hàm kích hoạt sẽ được trình bày ở phần kế tiếp).

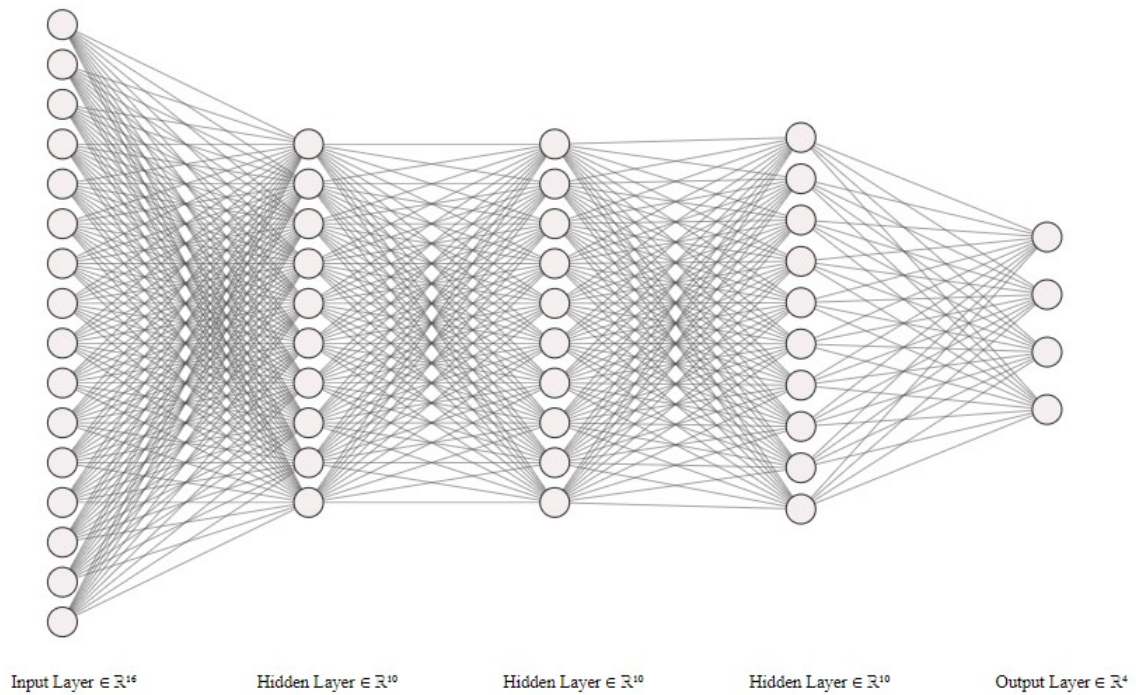


Hình 2.1: Hình mô phỏng một mạng nơ-ron nhân tạo đơn giản

Mạng nơ-ron nhân tạo dựa trên dữ liệu huấn luyện để tìm hiểu và cải thiện độ chính xác theo thời gian. Và khi các thuật toán này được tối ưu hoá về độ chính xác, nó sẽ trở thành công cụ mạnh mẽ trong khoa học máy tính và trí tuệ nhân tạo, cho phép ta có thể phân loại và phân cụm dữ liệu với tốc độ cao. Các tác vụ trong nhận dạng giọng nói hoặc nhận dạng hình ảnh có thể mất vài phút, trong khi việc nhận dạng thủ công mất đến hàng giờ đồng hồ. Một trong những mạng nơ-ron được biết đến

nhieu nhất là thuật toán tìm kiếm của Google.

Một mạng nơ-ron nhân tạo chỉ có một tầng vào và một tầng ra nhưng có thể có nhiều tầng ẩn. Và mạng nơ-ron sâu (hình 2.2) là mạng nơ-ron nhân tạo bao gồm nhiều tầng ẩn (từ 3 tầng ẩn trở lên).











Hình 2.2: Hình mô phỏng một mạng nơ-ron sâu với ba tầng ẩn

2.1.2 “Hàm kích hoạt” (Activation Function)

“Hàm kích hoạt” là những hàm được sử dụng để xác định đầu ra của mạng nơ-ron. Đây là thành phần rất quan trọng đối với mạng nơ-ron, nếu không có các phép biến đổi phi tuyến để xác định đầu ra của mạng nơ-ron thì dù ta có xây dựng một mạng nơ-ron sâu đến mức nào đi nữa, nó cũng chỉ có hiệu quả như một lớp tuyến tính. Vì vậy, nếu không có các hàm kích hoạt, khả năng dự đoán của mạng nơ-ron sẽ giảm đi rất nhiều, sẽ không khám phá được các mối quan hệ phi tuyến tiềm ẩn trong dữ liệu. Về cơ

bản các hàm kích hoạt được chia thành 2 loại: Hàm kích hoạt tuyến tính và hàm kích hoạt phi tuyến. Tầm quan trọng của hàm kích hoạt phi tuyến đã được nêu ở trên, vậy tại sao lại có các hàm kích hoạt tuyến tính? Trên thực tế, vẫn có nhiều trường hợp sử dụng hàm kích hoạt tuyến tính. Ví dụ, trong bài toán hồi quy, kết quả đầu ra y là số thực, ta có thể sử dụng hàm kích hoạt tuyến tính ngay trước đầu ra y , tuy nhiên các hàm kích hoạt ở các tầng ẩn khác bắt buộc phải có phi tuyến. Sau đây là một số hàm kích hoạt thường được sử dụng (hình 2.3):

Name	Plot	Function, $g(x)$	Derivative of g , $g'(x)$	Range	Order of continuity
Identity		x	1	$(-\infty, \infty)$	C^∞
Binary step		$\begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x \geq 0 \end{cases}$	$\begin{cases} 0 & \text{if } x \neq 0 \\ \text{undefined} & \text{if } x = 0 \end{cases}$	$\{0, 1\}$	C^{-1}
Logistic, sigmoid, or soft step		$\sigma(x) = \frac{1}{1 + e^{-x}}$	$g(x)(1 - g(x))$	$(0, 1)$	C^∞
Hyperbolic tangent (tanh)		$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$	$1 - g(x)^2$	$(-1, 1)$	C^∞
Rectified linear unit (ReLU) ^[11]		$\begin{cases} 0 & \text{if } x \leq 0 \\ x & \text{if } x > 0 \end{cases}$ $= \max\{0, x\} = x \mathbf{1}_{x>0}$	$\begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x > 0 \\ \text{undefined} & \text{if } x = 0 \end{cases}$	$[0, \infty)$	C^0
Gaussian Error Linear Unit (GELU) ^[2]		$\frac{1}{2}x \left(1 + \operatorname{erf}\left(\frac{x}{\sqrt{2}}\right)\right)$ $= x\Phi(x)$	$\Phi(x) + x\phi(x)$	$(-0.17\dots, \infty)$	C^∞
Softplus ^[12]		$\ln(1 + e^x)$	$\frac{1}{1 + e^{-x}}$	$(0, \infty)$	C^∞
Exponential linear unit (ELU) ^[13]		$\begin{cases} \alpha(e^x - 1) & \text{if } x \leq 0 \\ x & \text{if } x > 0 \end{cases}$ with parameter α	$\begin{cases} \alpha e^x & \text{if } x < 0 \\ 1 & \text{if } x > 0 \\ 1 & \text{if } x = 0 \text{ and } \alpha = 1 \end{cases}$	$(-\alpha, \infty)$	$\begin{cases} C^1 & \text{if } \alpha = 1 \\ C^0 & \text{otherwise} \end{cases}$
Scaled exponential linear unit (SELU) ^[14]		$\lambda \begin{cases} \alpha(e^x - 1) & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases}$ with parameters $\lambda = 1.0507$ and $\alpha = 1.67326$	$\lambda \begin{cases} \alpha e^x & \text{if } x < 0 \\ 1 & \text{if } x \geq 0 \end{cases}$	$(-\lambda\alpha, \infty)$	C^0

Hình 2.3: Một số hàm kích hoạt thông dụng (Nguồn: Wikipedia)

Như hình trên có thể thấy có rất nhiều hàm kích hoạt có thể được sử dụng, dưới đây là sơ lược về một số hàm kích hoạt liên quan đến mô hình chính:

- **ReLU:** Đây là hàm kích hoạt được sử dụng rất nhiều trong các mô hình học sâu hiện nay, bởi vì nó đơn giản nhưng lại mang đến hiệu quả rất cao. ReLU là một phép biến đổi phi tuyến đơn giản. Về cơ bản, hàm ReLU chỉ giữ lại các phần tử có giá trị dương và loại bỏ tất

cả các phần tử có giá trị âm, đưa các phần tử đó về bằng 0. Công thức:

$$ReLU(z) = \max(z, 0), \text{ với } z \text{ là các phần tử đầu vào.} \quad (2.1)$$

- **Softmax:** Là một hàm kích hoạt phi tuyến tính. Nó sẽ ánh xạ kết quả đầu ra thành các giá trị nằm trong khoảng $(0,1]$ và có tổng bằng 1. Vì vậy, đầu ra của hàm kích hoạt softmax là một phân phối xác suất. Hàm softmax thường được sử dụng trong các mô hình phân loại. Công thức:

$$\text{softmax}(z_i) = \frac{\exp(z_i)}{\sum_j \exp(z_j)} \quad (2.2)$$

trong đó, z_i là các giá trị đầu vào của hàm softmax (là đầu ra ban đầu của các mô hình phân loại); $\exp(z_i)$ là hàm mũ cho mỗi giá trị đầu vào của hàm softmax, hàm mũ hoạt động như một hàm phi tuyến tính, nó sẽ đưa ra một giá trị dương lớn hơn 0, giá trị này sẽ rất lớn nếu giá trị đầu vào dương và rất nhỏ nếu giá trị đầu vào âm. Sau đó, các giá trị này sẽ được chia cho tổng tất cả các giá trị của hàm mũ để chuẩn hóa.

- **Sparsemax:** Đây là hàm kích hoạt được xây dựng dựa trên hàm kích hoạt softmax. Tuy nhiên, nó khác softmax ở một chỗ đó là có thể tạo ra các xác suất thừa thớt, nó sẽ đưa các xác suất nhỏ hơn một ngưỡng cụ thể về bằng 0. Công thức:

$$\text{sparsemax}_i(z) = [z_i - \tau(z)]_+ \quad (2.3)$$

Giá trị của biểu thức này sẽ nằm trong khoảng $(0,1)$ và có tổng bằng 1. Trong đó:

- z_i là các phần tử đầu vào.

– $\tau(z)$ là ngưỡng để xem xét đưa xác suất về bằng 0.

- **GLU (gated linear unit):** Đây là một hàm kích hoạt phi tuyến. Nó được tính bằng cách lấy một phép biến đổi tuyến tính nhân với sigmoid của một phép biến đổi tuyến tính khác. Công thức:

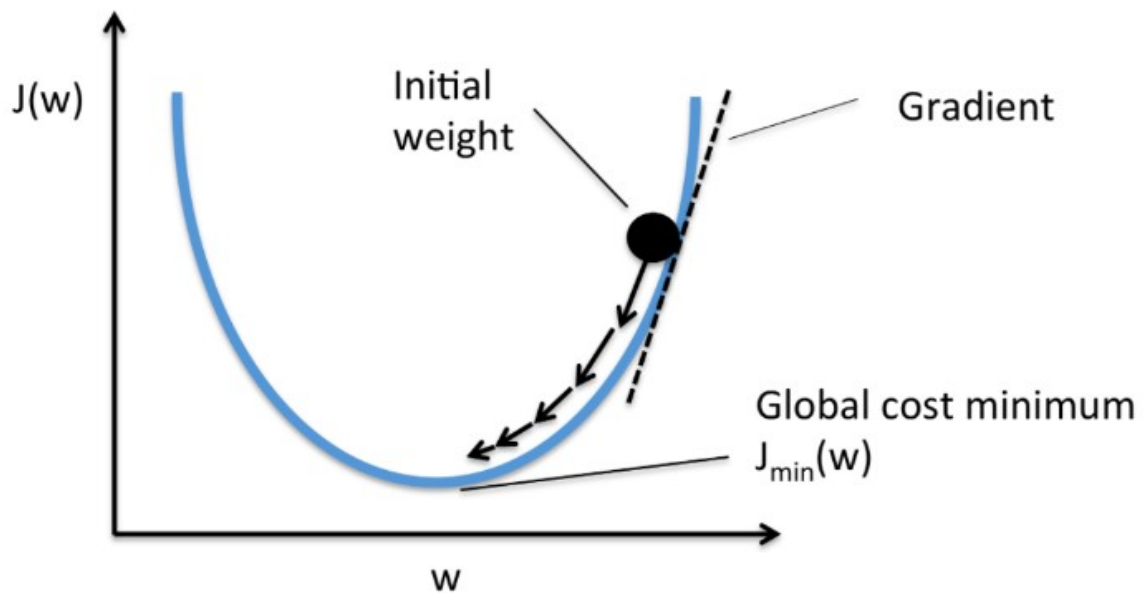
$$h_l(X) = (X * W + b) \otimes \sigma(X * V + c) \quad (2.4)$$

trong đó, X là input của lớp h_l , W và V là các trọng số, σ là hàm sigmoid.

2.1.3 “Gradient Descent”

Như đã nói ở phần trước, một mô hình học máy nói chung, hay mạng nơ-ron nhân tạo nói riêng sẽ trở thành công cụ mạnh mẽ cho các ứng dụng trí tuệ nhân tạo và khoa học máy tính, nếu như chúng được tối ưu hóa về độ chính xác. Vì vậy, mục đích chính của việc huấn luyện mô hình là tìm ra các tham số để cho hàm mất mát (hàm chi phí) của mô hình đạt giá trị nhỏ nhất. Để thực hiện điều này ta có thể sử dụng các thuật toán tối ưu hóa. Hiện nay, có rất nhiều thuật toán tối ưu hóa đã được ra đời và hoạt động rất hiệu quả, một trong số đó là thuật toán “Gradient Descent”.

“Gradient Descent” là một thuật toán tối ưu lặp, nó sẽ lặp đi lặp lại việc điều chỉnh giá trị các tham số thông qua mỗi dữ liệu huấn luyện để giảm thiểu hàm chi phí. Nói cách khác, tại mỗi điểm của hàm chi phí, “Gradient Descent” sẽ xác định đạo hàm (gradient), sau đó đi ngược lại với hướng của đạo hàm cho đến khi hàm số đạt cực tiểu. Hình 2.4 dưới đây mô phỏng sơ lược về hoạt động của thuật toán “Gradient Descent”:



Hình 2.4: Mô phỏng thuật toán “Gradient ” (Nguồn: [rasbt.github.io](https://github.com/rasbt))

Một cách cụ thể, xem xét hàm chi phí trên toàn bộ tập huấn luyện của một mô hình học nào đó:

$$J(\theta) = \frac{1}{N} \sum_{i=1}^N J^{(i)}(\theta) \quad (2.5)$$

Trong đó:

- θ là các tham số của mô hình học.
- $J^{(i)}(\theta)$ là chi phí của mẫu huấn luyện thứ i trong tập huấn luyện.
- N là tổng số mẫu huấn luyện.

Mục tiêu là tìm θ để $J(\theta)$ đạt cực tiểu.

Về cơ bản thì quá trình tối ưu hóa của “Gradient Descent” sẽ được thực thi như sau:

- Khởi tạo ngẫu nhiên cho θ .

- Tính toán gradient (đạo hàm riêng) của hàm chi phí bằng cách sử dụng đạo hàm từng phần: $G = \nabla_{\theta} J(\theta) = \frac{\partial J(\theta)}{\partial \theta}$, khi đó $\nabla_{\theta} J(\theta)$ là véc-tơ chứa các đạo hàm riêng của J theo θ .
- Cập nhật các tham số: $\theta = \theta - \eta G$, với η là tốc độ học (learning rate) xác định độ dài của một bước đi (bước nhảy).
- Lặp lại cho đến khi $J(\theta)$ ngừng giảm hoặc thỏa mãn các điều kiện dừng khác.

Hiện nay, có ba biến thể của “Gradient Descent”, chúng sẽ khác nhau về lượng dữ liệu sử dụng để tính toán véc-tơ đạo hàm riêng (véc-tơ gradient) của hàm chi phí, đó là:

- “Batch Gradient Descent” (BGD) còn được gọi là “Vanilla Gradient Descent”, nó sẽ duyệt qua toàn bộ tập dữ liệu huấn luyện để tính các véc-tơ đạo hàm riêng của hàm chi phí của mỗi mẫu huấn luyện rồi lấy trung bình của các véc-tơ đạo hàm riêng này để có được $\nabla_{\theta} J(\theta)$.
- “Stochastic Gradient Descent” (SGD): Khi tập huấn luyện lớn thì việc tính các véc-tơ đạo hàm riêng (gradient) trên từng mẫu một của BGD sẽ tốn thời gian và chạy rất chậm. SGD khắc phục được nhược điểm này của BGD bằng cách: thay vì thực hiện tính toán véc-tơ đạo hàm riêng dựa trên toàn bộ tập dữ liệu huấn luyện trong mỗi lần lặp thì SGD sẽ chọn ngẫu nhiên một mẫu huấn luyện để tính toán véc-tơ đạo hàm riêng và thực hiện cập nhật các tham số trong mỗi lần lặp. Tuy nhiên, vì sự ngẫu nhiên này mà làm cho thuật toán thiếu tính ổn định.
- “Mini-batch Gradient Descent”: Đây là thuật toán kết hợp giữa BGD và SGD. Nó chỉ đơn giản là chia ngẫu nhiên tập dữ liệu huấn luyện thành các mini-batch và thực hiện tính toán véc-tơ đạo hàm riêng cho từng mini-batch.

2.2 “XGBoost”

Học kết hợp (ensemble method) là một phương pháp với ý tưởng: Thay vì cố gắng xây dựng một mô hình tốt, có độ chính xác cao thì ta sẽ đi xây dựng các mô hình yếu hơn, có độ chính xác kém hơn một chút (weak learner) khi đi riêng lẻ, nhưng khi kết hợp các mô hình lại sẽ thu được một mô hình vượt trội hơn. Người ta có thể chia các thuật toán ensemble thành nhiều nhóm khác nhau, trong đó hai nhóm nổi trội nhất đó là:

- “Bagging” (ensemble sử dụng bootstrap): Là viết tắt của “Bootstrap Aggregating”. Nó xây dựng một lượng lớn các mô hình trên những tập con khác nhau từ tập dữ liệu huấn luyện (lấy mẫu ngẫu nhiên có thay thế). Những mô hình này sẽ được huấn luyện độc lập và song song với nhau và khi dự đoán thì đầu ra của các mô hình sẽ được tổng hợp, lấy trung bình cộng để đưa ra dự đoán cuối cùng.
- “Boosting” (ensemble sử dụng boosting): Xây dựng một lượng lớn các mô hình. Mô hình sau sẽ cố gắng sửa lỗi của mô hình trước (dữ liệu mà mô hình trước dự đoán sai). Khi một mô hình mới được thêm vào, trọng số của dữ liệu sẽ được điều chỉnh lại, cụ thể là trọng số của những dữ liệu dự đoán đúng sẽ không đổi, còn trọng số của những dữ liệu dự đoán sai sẽ tăng lên. Điều này giúp cho các mô hình mới có thể tập trung hơn vào các mẫu dữ liệu đang bị dự đoán sai. Chúng ta sẽ lấy kết quả của mô hình cuối cùng trong chuỗi mô hình làm kết quả trả về.

“Gradient Boosting” là một thuật toán dựa trên “boosting”, nó tái định nghĩa lại boosting là một bài toán tối ưu hóa, trong đó mục tiêu là giảm thiểu hàm mất mát của mô hình bằng cách thêm các mô hình sử dụng “gradient descent”. Thay vì gán trọng số cho các mẫu dữ liệu cụ thể thì nó xây dựng một lượng lớn các mô hình, mỗi mô hình sau sẽ dự đoán lỗi (lỗi phần dư residuals error) do mô hình trước mắc phải thay vì dự đoán trực

tiếp mục tiêu. Trong số các thuật toán học máy được sử dụng trong thực tế, tăng cường độ dốc (Gradient Boosting) là thuật toán đạt được nhiều thành công trong các ứng dụng. “XGBoost” là một phiên bản có thể mở rộng và được cải tiến của thuật toán “Gradient Boosting”. Nó được nhóm tác giả của trường đại học Washington công bố vào năm 2016 tại hội nghị quốc tế ACM SIGKDD lần thứ 22 [3]. “XGBoost” là viết tắt của “Extreme Gradient Boosting”, đây là một thuật toán mạnh mẽ nhằm giải quyết bài toán học có giám sát cho độ chính xác cao và đã chiến thắng nhiều cuộc thi trên Kaggle. Yếu tố quan trọng nhất tạo nên sự thành công này đó là khả năng mở rộng của nó trong mọi tình huống. So với các thuật toán trước đó, “XGBoost” có những đổi mới nổi bật:

- Là một thuật toán dựa trên cây mới cho phép xử lý dữ liệu thưa thớt.
- Tính toán song song trên CPU/GPU và tính toán phân tán trên nhiều server giúp học tập nhanh hơn, cho phép triển khai và khám phá mô hình tốt hơn.
- “XGBoost” khai thác việc tính toán ngoài lõi (out-of-core) cho phép người dùng có thể xử lý hàng trăm triệu ví dụ.
- Tạo ra một hệ thống học từ đầu đến cuối (end-to-end) có quy mô dữ liệu lớn và khả năng mở rộng cao.
- Có khả năng xử lý dữ liệu bị thiếu (missing value data).

2.2.1 Các đặc trưng của “XGBoost” (XGBoost Features)

Quy ước: ¹

- n : số lượng mẫu huấn luyện.

¹XGBoost: thuật toán giành chiến thắng tại nhiều cuộc thi Kaggle - Ông Xuân Hồng

- m : số lượng đặc trưng.
- $\mathcal{D} = \{(x_i, y_i)\}$ là tập dữ liệu huấn luyện với $|\mathcal{D}| = n, x_i \in \mathbb{R}^m, y_i \in \mathbb{R}$.
- q : cấu trúc của một cây.
- T : số lượng nút lá của cây.
- f_k : cấu trúc các cây k độc lập của mô hình.
- w_i : trọng số của nút lá thứ i .
- $\hat{y}_i^{(t)}$: giá trị dự đoán của mẫu thứ i tại vòng lặp thứ t .
- $f_t^2(x_i)$: đạo hàm bậc 2 của hàm f .
- $I_j = \{i | q(x_i) = j\}$: tập các giá trị tại nút lá j .
- I_L : tập các giá trị tại nút lá bên trái.
- I_R : tập các giá trị tại nút lá bên phải.
- $I = I_L \cup I_R$.

a. Học chính quy (Regularized Learning)

Mô hình kết hợp các cây quyết định đưa ra dự đoán đầu ra bằng cách:

$$\hat{y}_i = \phi(x_i) = \sum_{k=1}^K f_k(x_i), \quad f_k \in \mathcal{F}, \quad (2.6)$$

trong đó, $\mathcal{F} = \{f(x) = w_{q(x)}\} (q : \mathbb{R}^m \rightarrow T, w \in \mathbb{R}^T)$

Hàm học:

$$\mathcal{L}(\phi) = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k) \quad (2.7)$$

trong đó, $\Omega(f) = \gamma T + \frac{1}{2} \lambda \|w\|^2$

Ở đây, l là một hàm mất mát để đo lường sự khác biệt giữa \hat{y}_i (dự đoán) và y_i (mục tiêu). Ω là tham số phạt độ phức tạp của mô hình. Chính quy giúp làm mượt các trọng số đã học cuối cùng để tránh “quá khớp dữ liệu” (overfitting).

b. Gradient Tree Boosting

Hàm học trong “XGBoost” không thể được tối ưu hóa bằng các phương pháp tối ưu hóa trong không gian Euclide như truyền thống. Nó sẽ được huấn luyện một cách bổ sung, thêm $f(t)$ để tối ưu hóa mục tiêu:

$$\mathcal{L}^{(t)} = \sum_{i=1}^n l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \Omega(f_t) \quad (2.8)$$

Sử dụng thêm xấp xỉ bậc hai để tối ưu hóa mục tiêu ²:

$$\mathcal{L}^{(t)} \simeq \sum_{i=1}^n [l(y_i, \hat{y}^{(t-1)}) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + \Omega(f_t) \quad (2.9)$$

trong đó, $g_i = \partial_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$, $h_i = \partial_{\hat{y}^{(t-1)}}^2 l(y_i, \hat{y}^{(t-1)})$

Bỏ hằng số đi để thu được mục tiêu tối ưu hóa ở bước t :

$$\tilde{\mathcal{L}}^{(t)} = \sum_{i=1}^n [g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + \Omega(f_t) \quad (2.10)$$

Thay thế phép tính của Ω vào:

$$\begin{aligned} \tilde{\mathcal{L}}^{(t)} &= \sum_{i=1}^n [g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + \gamma T + \frac{1}{2} \lambda \sum_{j=1}^T w_j^2 \\ &= \sum_{j=1}^T [(\sum_{i \in I_j} g_i) w_j + \frac{1}{2} (\sum_{i \in I_j} h_i + \lambda) w_j^2] + \gamma T \end{aligned} \quad (2.11)$$

²J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, 28(2):337–407, 2000.

Trọng số tối ưu tại mỗi nút lá:

$$w_j^* = -\frac{\sum_{i \in I_j} g_i}{\sum_{i \in I_j} h_i + \lambda} \quad (2.12)$$

Hàm lỗi tính trên toàn bộ cây:

$$\tilde{\mathcal{L}}^{(t)}(q) = -\frac{1}{2} \sum_{j=1}^T \frac{(\sum_{i \in I_j} g_i)^2}{\sum_{i \in I_j} h_i + \lambda} + \gamma T \quad (2.13)$$

Tiêu chí để phân chia tốt hơn (để đánh giá các thành phần đã phân chia):

$$\mathcal{L}_{split} = \frac{1}{2} \left[\frac{(\sum_{i \in I_L} g_i)^2}{\sum_{i \in I_L} h_i + \lambda} + \frac{(\sum_{i \in I_R} g_i)^2}{\sum_{i \in I_R} h_i + \lambda} - \frac{(\sum_{i \in I} g_i)^2}{\sum_{i \in I} h_i + \lambda} \right] - \gamma \quad (2.14)$$

c. Shrinkage và Column Subsampling

Để tránh bị “quá khớp dữ liệu” (overfitting) thì bên cạnh việc sử dụng chính quy hóa thì “XGBoost” còn sử dụng thêm hai kỹ thuật khác:

- Kỹ thuật Shrinkage, sẽ chia tỷ lệ của trọng số mới được thêm vào theo hệ số η sau mỗi bước. Điều này sẽ giúp giảm ảnh hưởng của từng cây riêng lẻ để có thể cải thiện mô hình dựa vào các cây tiếp theo.
- Lấy mẫu con theo cột (đặc trưng), điều này sẽ ngăn ngừa việc quá khớp dữ liệu hơn so với lấy mẫu con như truyền thống (theo hàng).

2.2.2 Các thuật toán tìm kiếm sự phân tách (Split Finding Algorithms)

a. Thuật toán tham lam chính xác (Basic Exact Greedy Algorithm)

Đối với các thuật toán dựa trên cây quyết định thì việc tìm ra sự phân tách tốt nhất rất quan trọng. Và “XGBoost” sẽ sử dụng thuật toán tham lam chính xác để thực hiện điều này. Thuật toán này sẽ liệt kê tất cả các phân tách có thể có trên tất cả các đặc trưng sau đó chọn ra phân tách tốt nhất. Tổng thể thuật toán tham lam chính xác được trình bày ở thuật toán 1.

Algorithm 1: Exact Greedy Algorithm for Split Finding

Input: I , instance set of current node
Input: d , feature dimension
 $gain \leftarrow 0$
 $G \leftarrow \sum_{i \in I} g_i, H \leftarrow \sum_{i \in I} h_i$
for $k = 1$ **to** m **do**
 $G_L \leftarrow 0, H_L \leftarrow 0$
 for j *in* $sorted(I, \text{by } x_{jk})$ **do**
 $G_L \leftarrow G_L + g_j, H_L \leftarrow H_L + h_j$
 $G_R \leftarrow G - G_L, H_R \leftarrow H - H_L$
 $score \leftarrow \max(score, \frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{G^2}{H + \lambda})$
 end
end
Output: Split with max score

b. Thuật toán gần đúng (Approximate Algorithm)

Việc sử dụng thuật toán tham lam giúp cho quá trình tìm kiếm phân tách diễn ra rất nhanh chóng. Tuy nhiên, khi tập dữ liệu trở nên quá lớn, dữ liệu không thể nằm gọn trong bộ nhớ và khi cài đặt xử lý phân tán thì thuật toán tham lam trở nên rất chậm, không hiệu quả. Để khắc phục được điều này, nhóm tác giả đã sử dụng thuật toán gần đúng trong “XGBoost”, tính toán sự phân chia trong các cây là gần đúng. Tổng thể thuật toán gần đúng được trình bày ở thuật toán 2.

Algorithm 2: Approximate Algorithm for Split Finding

```
for  $k = 1$  to  $m$  do
  | Propose  $S_k = s_{k1}, s_{k2}, \dots, s_{kl}$  by percentiles on feature  $k$ .
  | Proposal can be done per tree (global), or per split(local).
end
for  $k = 1$  to  $m$  do
  |  $G_{kv} \leftarrow \sum_{j \in \{j | s_{k,v} \geq x_{jk} > s_{k,v-1}\}} g_j$ 
  |  $H_{kv} \leftarrow \sum_{j \in \{j | s_{k,v} \geq x_{jk} > s_{k,v-1}\}} h_j$ 
end
Follow same step as in previous section to find max score only
among proposed splits.
```

c. Tìm kiếm phân tách nhận biết sự thưa thớt (Sparsity-aware Split Finding)

Thuật toán này giúp xử lý các thông tin bị thiếu trong dữ liệu và cung cấp cơ sở về cách xử lý dữ liệu bị thiếu mới. Tập dữ liệu sẽ được chia thành hai nhóm. Một nhóm chứa dữ liệu không bị thiếu, một nhóm chứa dữ liệu bị thiếu. Đầu tiên, xây dựng một cây cơ sở dựa theo nhóm dữ liệu không bị thiếu. Sau đó, quá trình tìm kiếm phân tách sẽ qua hai bước:

- Đầu tiên, sẽ tính toán gain bằng cách thêm các giá trị bị thiếu từ nhóm hai vào bên trái cây.
- Thứ hai, sẽ tính toán gain bằng cách thêm các giá trị bị thiếu từ nhóm hai vào bên phải cây.

Chương 3

Mô hình “TabNet”

Chương này trình bày về mô hình mà chúng em tập trung tìm hiểu trong khóa luận này - “TabNet” [1]. Đây là một mô hình học sâu cho bài toán học có giám sát với dữ liệu dạng bảng. “TabNet” sử dụng sự chú ý tuần tự (sequential attention) để lựa chọn các đặc trưng quan trọng cho từng bước quyết định. Ưu điểm của “TabNet” là: (1) mô hình cho phép con người có thể hiểu về dự đoán mà mô hình đưa ra (trên cả tổng thể và từng đầu vào cụ thể), (2) dễ dàng mở rộng kiến trúc mạng (ví dụ, để dự đoán nhiều output thì chỉ cần thêm các nơ-ron vào tầng output), (3) do dùng “SGD” nên có thể hoạt động được với ngữ cảnh học “online” (tại một thời điểm chỉ có một mẫu dữ liệu để cập nhật mô hình) và học với dữ liệu lớn (chỉ có thể dùng một tập con để cập nhật mô hình, do không đủ bộ nhớ để load toàn bộ dữ liệu). Đầu tiên chúng em sẽ đưa ra kiến trúc tổng thể của “TabNet”, sau đó sẽ đi sâu vào kiến trúc và nhiệm vụ của từng thành phần cụ thể.

Quy ước:

- n_{steps} : số bước quyết định.

- n_a : kích thước đầu vào của khối Attentive Transformer.
- n_d : kích thước đầu vào của khối quyết định (khối dự đoán).
- $n_{features}$: số lượng đặc trưng đầu vào.
- B : kích thước batch (batch size).

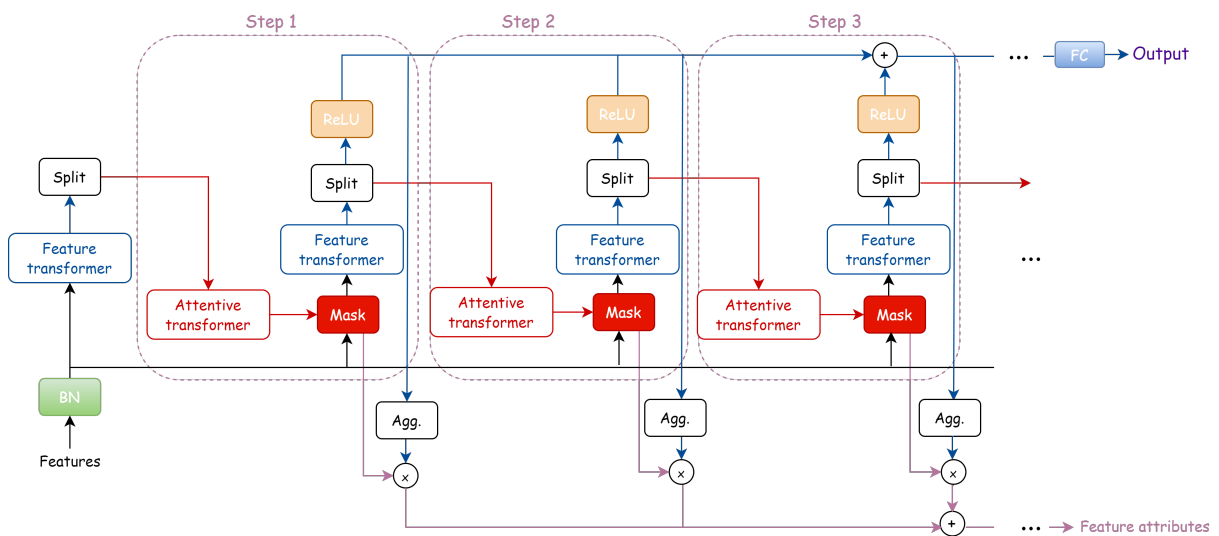
3.1 Kiến trúc chung

“TabNet” là một mạng nơ-ron sâu mới cho dữ liệu dạng bảng. Ý tưởng chung của “TabNet” là dựa trên “sự chú ý tuần tự” (sequential attention), để lựa chọn các đặc trưng quan trọng ở các bước quyết định khác nhau. Kiến trúc của nó bao gồm nhiều bước quyết định tuần tự nhau, liên tiếp nhau (n_{steps}) (hình 3.1). Bước thứ i nhận thông tin đã được xử lý từ bước $i-1$ để quyết định đặc trưng nào sẽ được sử dụng. Mỗi bước quyết định gồm các quá trình: **Attentive Transformer** \rightarrow **Mask** \rightarrow **Feature Transformer** \rightarrow **Split** \rightarrow **Output**. Khối Attentive Transformer nhận đầu ra của khối Feature Transformer của bước quyết định trước đó làm đầu vào, tại đây dữ liệu sẽ được xử lý, chuyển đổi để đưa ra một ma trận thưa thớt. Dựa vào ma trận này và các đặc trưng đầu vào ta sẽ có được các Mask, các Mask này giúp mô hình có thể chọn được các đặc trưng quan trọng ở từng bước quyết định cụ thể. Về cơ bản, các Mask che đi một số đặc trưng, điều này giúp cho mô hình tập trung vào các đặc trưng được cho là quan trọng, bởi khối Attentive Transformer ở mỗi bước quyết định. Các đặc trưng sau khi được chọn sẽ chuyển đến khối Feature Transformer để xử lý. Sau đó, tách ra thành hai phần, một phần dùng làm đầu ra cho bước quyết định (có kích thước n_d) (1), một phần dùng làm đầu vào cho bước quyết định tiếp theo, hay cụ thể hơn là khối Attentive Transformer (có kích thước n_a) (2). Phần (1) sẽ đi qua một hàm kích hoạt ReLU để đưa ra đầu ra cuối cùng cho bước quyết định, kết hợp đầu ra của tất cả các bước quyết định lại với nhau sau đó đi qua một tầng kết nối đầy đủ

(fully connected layer - đây là những tầng trong mạng nơ-ron mà mỗi node trong tầng đó được kết nối với tất cả các node trong tầng trước đó) ta sẽ có được đầu ra cuối cùng của mô hình.

Dựa vào hình 3.1, đầu vào của bước quyết định đầu tiên là dữ liệu đầu vào với tất cả các đặc trưng được chuẩn hóa thông qua “batch normalization” (BN - đây là một phương pháp chuẩn hóa đưa dữ liệu về dạng phân phối chuẩn, nghĩa là trung bình xấp xỉ 0 và độ lệch chuẩn xấp xỉ 1), sau đó chuyển đến khối Feature Transformer xử lý và tách một phần làm đầu vào cho bước quyết định đầu tiên.

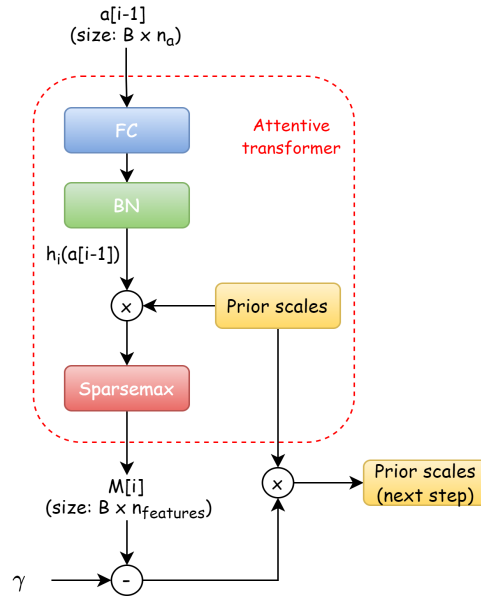
Thêm vào đó, “TabNet” có khả năng đưa ra diễn giải trên cả toàn cục và cục bộ, điều này phần nào giúp con người có thể hiểu được các dự đoán mà mô hình đưa ra trên tổng thể và từng input cụ thể. Và để thực hiện được chức năng này, mô hình sẽ dựa vào các Mask và đầu ra của từng bước quyết định.



Hình 3.1: Kiến trúc chung của “TabNet”

3.2 Attentive Transformer

Như đã trình bày ở phần trước, dữ liệu sau khi được xử lý bởi khối Feature Transformer sẽ chia thành hai phần: một phần làm đầu ra cho bước quyết định, một phần làm đầu vào cho bước quyết định tiếp theo hay cụ thể hơn là làm đầu vào cho khối Attentive Transformer. Tại đây dữ liệu sẽ được xử lý, chuyển đổi để đưa ra một ma trận thưa thớt (có một số phần tử bằng 0). Kết hợp ma trận này và các đặc trưng ban đầu sẽ có được các Mask - thành phần quan trọng để chọn được các đặc trưng ở từng bước quyết định và đưa ra khả năng diễn giải của mô hình. Hình 3.2 dưới đây biểu diễn các thành phần cơ bản của một khối Attentive Transformer:



Hình 3.2: Khối Attentive Transformer

Trong “TabNet” để lựa chọn các đặc trưng quan trọng trong từng bước quyết định cụ thể, nhóm tác giả sử dụng ma trận Mask có thể học được $M[i]$ (đây là một ma trận thưa thớt). Nhờ vào việc lựa chọn thưa thớt (ít) các đặc trưng nổi bật nhất nên quá trình xử lý, học tập ở mỗi bước quyết định không bị lãng phí vào những đặc trưng không liên quan, do đó mô hình trở nên hiệu quả hơn về mặt tham số. Và để có được $M[i]$, nhóm tác

giả đã sử dụng khối Attentive Transformer với đầu vào là các đặc trưng đã được xử lý từ bước trước, $a[i - 1]$. Cụ thể như sau:

- Đầu tiên, đầu ra của khối Feature Transformer ở bước quyết định trước là một ten-xơ, được đưa đến mô-đun Split. Tại đây, ten-xơ sẽ được tách ra hai phần, ta sẽ có được $a[i - 1]$.
- Tiếp đó, $a[i - 1]$ đi qua hàm h_i , đây là một hàm có thể huấn luyện, được biểu diễn bằng cách sử dụng một tầng kết nối đầy đủ (FC layer) theo sau bởi chuẩn hóa “batch normalization” (BN). Vai trò của h_i là đạt được sự kết hợp tuyến tính giữa các đặc trưng từ đó đưa ra được các đặc trưng có số chiều nhiều hơn (cụ thể là từ kích thước n_a đến kích thước $n_{features}$) và đã được chuẩn hóa.
- Lấy đầu ra từ hàm h_i nhân với một “tỉ lệ trước” (prior scale). Cuối cùng, $M[i]$ được xác định thông qua hàm kích hoạt “sparsemax” theo công thức:

$$M[i] = \text{sparsemax}(P[i - 1].h_i(a[i - 1])) \quad (3.1)$$

“Tỉ lệ trước” biểu thị cho mức độ sử dụng các đặc trưng trong các bước quyết định trước. Các đặc trưng đã được sử dụng nhiều trong các bước trước sẽ có trọng số nhỏ trong bước hiện tại. Vì vậy, sử dụng ma trận Mask để cập nhật cho “tỉ lệ trước”:

$$P[i] = \prod_{j=1}^i (\gamma - M[j]) \quad (3.2)$$

Trong đó, γ là tham số “relaxation”, khi $\gamma = 1$ có nghĩa là các đặc trưng chỉ được sử dụng ở một bước quyết định duy nhất; và khi γ tăng lên, lớn hơn 1 có nghĩa là các đặc trưng có thể được sử dụng ở các bước quyết định khác nhau. Ban đầu $P[0]$ được khởi tạo bằng 1, tức là tất cả các đặc trưng đều như nhau, đều chưa được sử dụng.

Như đã trình bày ở chương 2, hàm kích hoạt “sparsemax” có thể đưa ra một ma trận xác suất thưa thớt (đưa các xác suất nhỏ hơn một ngưỡng cụ thể về bằng 0), điều này giúp cho “TabNet” có thể lựa chọn một cách thưa thớt các đặc trưng quan trọng ở từng bước quyết định. “Sparsemax” sẽ làm cho $\sum_{j=1}^D M[i]_{b,j} = 1$, với D là kích thước của đặc trưng (hay số lượng đặc trưng). “Sparsemax” thực hiện đưa ra các trọng số cho từng đặc trưng j của mỗi mẫu b và đảm bảo rằng tổng trọng số của tất cả các đặc trưng của mỗi mẫu bằng 1.

- Vì việc lựa chọn thưa thớt các đặc trưng ở mỗi bước quyết định, nên cần phải kiểm soát thêm về mức độ thưa thớt này, nhóm tác giả sử dụng “chính quy hóa thưa thớt” (sparsity regularization) dưới dạng entropy [5]:

$$L_{sparse} = \sum_{i=1}^{n_{steps}} \sum_{b=1}^B \sum_{j=1}^D \frac{-M_{b,j}[i] \log(M_{b,j}[i] + \epsilon)}{n_{steps} \cdot B} \quad (3.3)$$

trong đó, ϵ là một số nhỏ để ổn định số, vì $M_{b,j}[i]$ có thể bằng 0, điều này sẽ làm hàm log bị lỗi không xác định được.

“Chính quy hóa thưa thớt” được thêm vào hàm mất mát L với một hệ số λ_{sparse} :

$$L = L - \lambda \cdot L_{sparse} \quad (3.4)$$

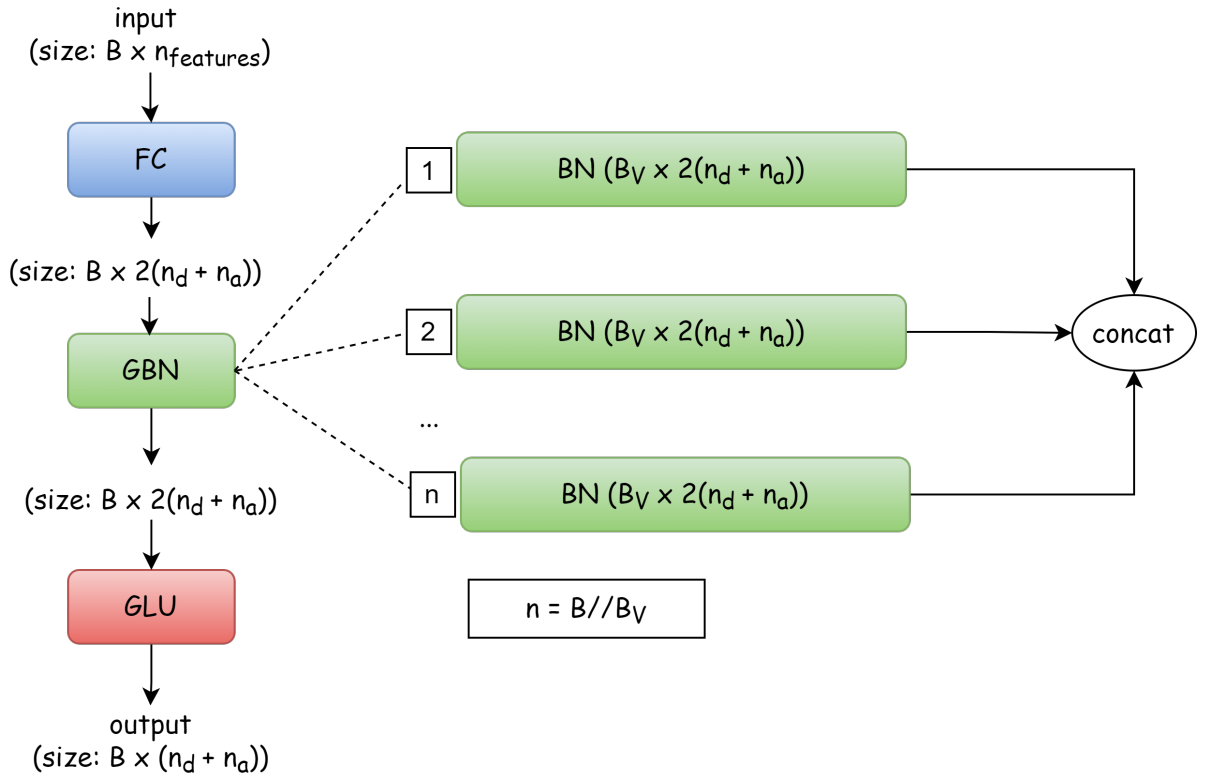
- Cuối cùng, sau khi có được $M[i]$, Mask sẽ được xác định bằng cách lấy $M[i]$ nhân với ten-xơ các đặc trưng đầu vào ($Mask = M[i].f$) để thực hiện việc lựa chọn đặc trưng ở bước quyết định hiện tại.

3.3 Feature Transformer

Đặc trưng sau khi được chọn bởi Mask sẽ đưa vào khối Feature Transformer để xử lý.

Cấu trúc Feature Transformer bao gồm các khối Gated Linear Units (GLU). Ở mỗi khối Gated Linear Units (GLU) gồm ba tầng nối liền nhau:

- Tầng kết nối đầy đủ (fully-connected): Đầu vào của tầng kết nối đầy đủ là một ten-xơ (hình 3.3). Kích thước của ten-xơ đầu vào là số đặc trưng được chọn ($n_{features}$) bởi Mask nếu thuộc khối GLU đầu tiên của khối Feature Transformer, hoặc là bằng kích thước đầu vào của khối Attentive Transformer cộng với kích thước đầu vào của khối quyết định ($n_d + n_a$).
- Chuẩn hóa hàng loạt (BN - Batch Normalization): Sau tầng kết nối đầy đủ là tầng chuẩn hóa hàng loạt (hình 3.3). BN được sử dụng với batch size lớn để giúp mô hình học sâu cho dữ liệu dạng bảng được huấn luyện nhanh hơn. BN sẽ được áp dụng vào chuẩn hóa dữ liệu thô ban đầu và tại mỗi khối GLU sẽ sử dụng “chuẩn hóa ghost” (ghost BN - GBN) thay vì BN. GBN chia batch size thành các mini-batch để BN hoạt động hiệu quả hơn.
- GLU (Gated Linear Units): GLU theo sau tầng chuẩn hóa hàng loạt (BN) (hình 3.3). GLU sẽ giúp mô hình huấn luyện dữ liệu ổn định.



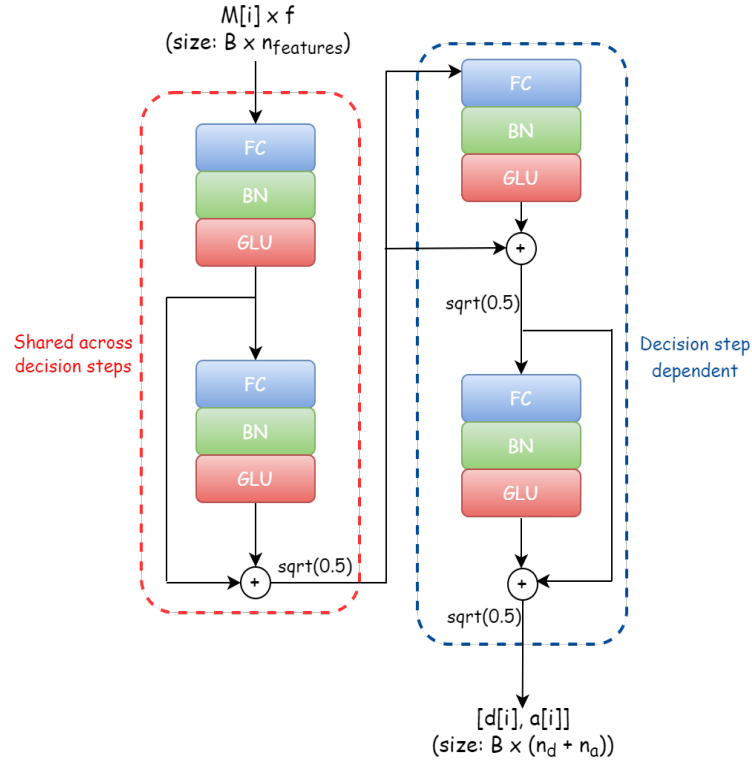
Hình 3.3: Cấu trúc khối Gated Linear Units (GLU).

Mỗi khối Feature Transformer (hình 3.4) gồm bốn khối Gated Linear Units (khối GLU):

- Hai khối đầu là hai khối dùng chung cho các bước quyết định: Ở hai khối dùng chung, trọng số của tầng kết nối đầy đủ sẽ được xác định một lần duy nhất ở Feature Transformer đầu tiên. Về sau, tất cả các tầng kết nối đầy đủ thuộc hai khối dùng chung đều sử dụng trọng số đã được xác định ở trên. Việc xác định duy nhất một lần giúp mô hình tiết kiệm tính toán tham số.
- Hai khối cuối là hai khối dùng riêng cho mỗi bước quyết định: Trọng số của tầng kết nối đầy đủ ở hai khối cuối sẽ khác nhau ở mỗi khối Feature Transformer. Điều này giúp mô hình linh hoạt hơn.

Ở khối Feature Transformer, sau khi thực hiện hai khối dùng chung và sau mỗi khối GLU dùng riêng sẽ thực hiện “chuẩn hóa phần dư” (normalized

residual). “Chuẩn hóa phần dư” với $\sqrt{0.5}$ giúp cho mô hình huấn luyện ổn định bằng cách đảm bảo variance (phương sai) không thay đổi khi qua các tầng.



Hình 3.4: Khối Feature Transformer

3.4 Decision Steps

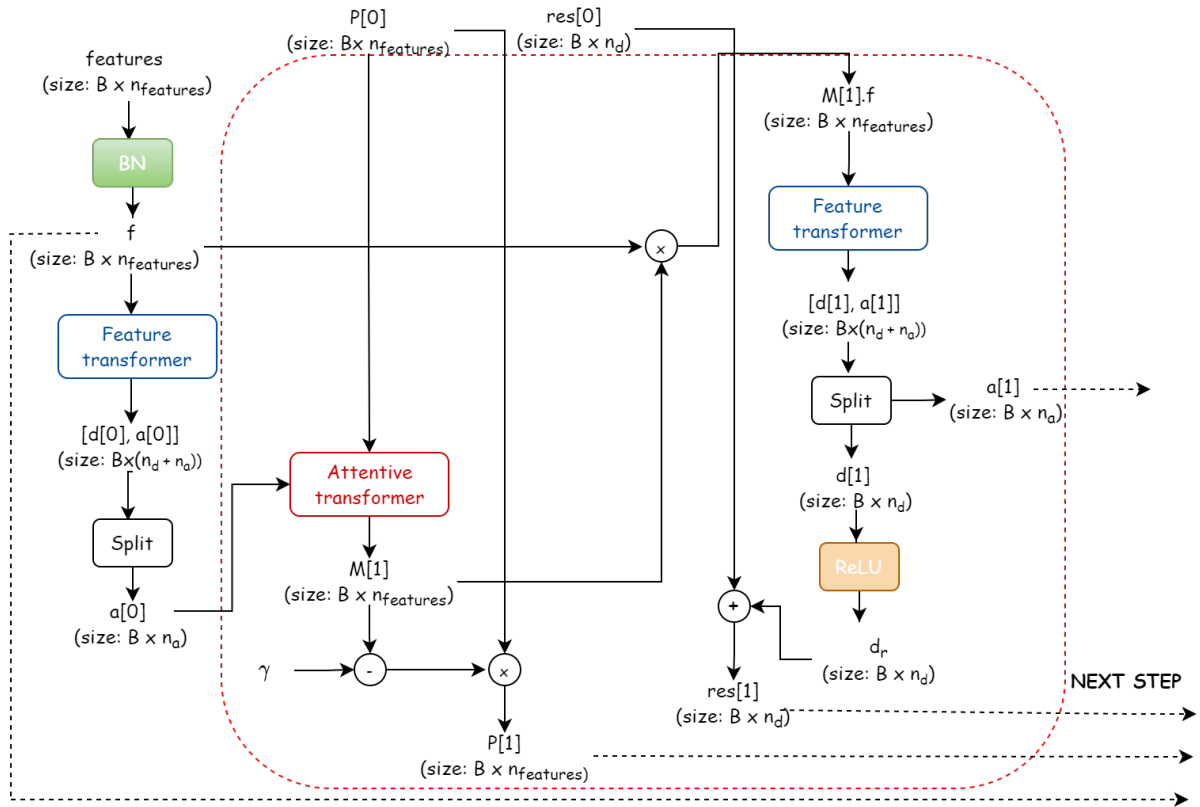
Mô hình “TabNet” bao gồm nhiều bước quyết định (Decision steps) nối tiếp nhau. Hai cấu trúc quan trọng của mô hình “TabNet” là Feature Transformer và Attentive Transformer. Tại mỗi bước quyết định sẽ bao gồm hai cấu trúc trên. Cấu trúc chung của các bước quyết định (hình 3.6):

- Khối Attentive Transformer nhận đầu ra của khối Feature Transformer ở bước quyết định trước đó làm đầu vào.
- Ten-xơ prior ở bước thứ i ($P[i - 1]$) được cập nhật ở bước trước, kết hợp với khối Attentive Transformer để tạo ma trận $M[i]$.

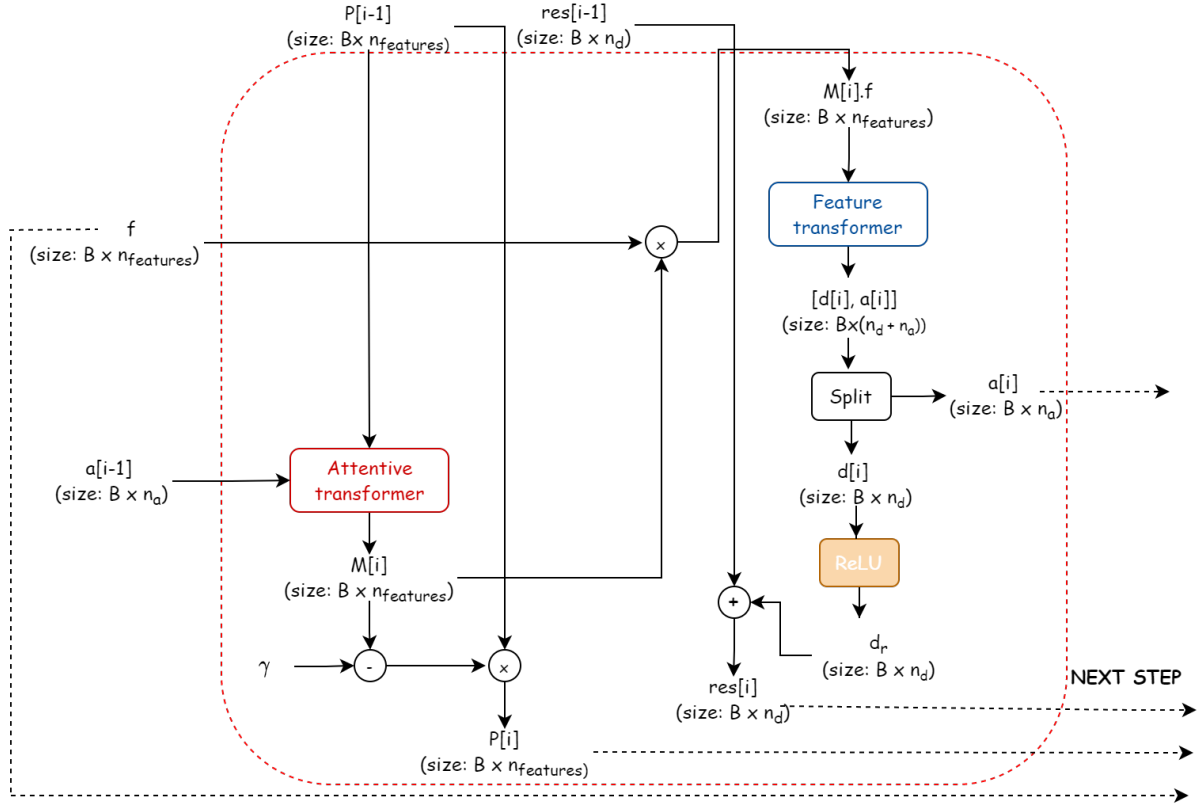
- Ma trận $M[i]$ sẽ cập nhật ten-xơ prior $P[i]$ để sử dụng cho những bước quyết định sau này. Ma trận $M[i]$ còn dùng để lựa chọn những đặc trưng sẽ được đưa vào khối Feature Transformer, bằng cách lấy ma trận $M[i]$ nhân với các đặc trưng đầu vào đã được chuẩn hóa BN (f) cho ra Mask ($\text{Mask} = M[i].f$).
- Mask làm đầu vào cho khối Feature Transformer. Feature Transformer sẽ thực hiện xử lý đặc trưng (feature processing) cho đầu ra có kích thước $n_a + n_d$.
- Đầu ra của khối Feature Transformer được chia làm hai phần. Một phần có kích thước n_a sẽ làm đầu vào cho khối Attentive Transformer của bước quyết định tiếp theo. Phần còn lại có kích thước n_d sẽ được đưa vào hàm ReLU để đưa ra đầu ra cho bước quyết định.
- Res là tổng sigma các giá trị đầu ra của hàm ReLU qua các bước quyết định.

Ở bước quyết định đầu tiên, đầu vào khác với các bước quyết định thứ i (hình 3.5):

- Dữ liệu dạng bảng thô (features) được đưa vào mô hình “TabNet”. Tiếp đó, các đặc trưng phân loại (categorical) được chuẩn hóa “embedding” và các đặc trưng dạng số (numerical) không cần xử lý. Sau khi thực hiện “embedding” sẽ đi qua tầng BN để chuẩn hóa ra được dữ liệu (f).
- Dữ liệu đã chuẩn hóa (f) tiếp tục đi qua khối Feature Transformer tạo ra dữ liệu có kích thước là $n_a + n_d$.
- Đầu ra Feature Transformer được chia làm hai phần, một phần n_a là đầu vào cho khối Attentive Transformer. Ở bước quyết định đầu tiên, khởi tạo một ten-xơ prior $P[0]$ với các giá trị bằng 1, để tạo ma trận $M[1]$.



Hình 3.5: Cấu trúc Decision Step (Step 1)



Hình 3.6: Cấu trúc Decision Step (Step i)

3.5 Khả năng diễn giải

Ma trận Mask giúp làm rõ việc lựa chọn đặc trưng ở từng bước quyết định. Nếu đặc trưng thứ j của mẫu b không được chọn ở bước quyết định thứ i thì giá trị Mask của nó bằng 0 ($M_{b,j}[i] = 0$). Giá trị $M_{b,j}[i]$ sẽ là tầm quan trọng của đặc trưng thứ j của mẫu b ở bước quyết định thứ i . Sau cùng, ma trận Mask riêng lẻ ở mỗi bước quyết định sẽ được kết hợp lại thành một ma trận Mask tổng hợp. Vì mỗi bước quyết định cho ra một ma trận Mask khác nhau, nên cần phải đánh trọng số mức độ quan trọng ở mỗi bước quyết định. Nhóm tác giả đã đề xuất:

$$\eta_b[i] = \sum_{c=1}^{N_d} ReLU(d_{b,c}[i]) \quad (3.5)$$

để thể hiện mức độ đóng góp của bước quyết định thứ i vào tổng thể cho mẫu b . Nếu có bất kì $d_{b,c}[i] < 0$ thì toàn bộ đặc trưng ở bước quyết định thứ i sẽ không tham gia đưa ra quyết định chung. Nếu giá trị tầm quan trọng càng cao thì đặc trưng đó càng đóng vai trò quan trọng trong việc đưa ra quyết định cuối cùng. Scaling ma trận Mask ở bước quyết định với $\eta_b[i]$. Nhóm tác giả thực hiện việc tổng hợp Mask bằng công thức:

$$M_{agg-b,j} = \frac{\sum_{i=1}^{N_{steps}} \eta_b[i] M_{b,j}[i]}{\sum_{j=1}^D \sum_{i=1}^{N_{steps}} \eta_b[i] M_{b,j}[i]} \quad (3.6)$$

“TabNet” đồng thời cho phép hai loại diễn giải:

- Diễn giải cục bộ (local interpretability) bằng cách trực quan mức độ quan trọng của đặc trưng. “TabNet” cho phép trực quan dễ dàng ma trận Mask ở mỗi bước quyết định và ma trận Mask tổng hợp.
- Diễn giải toàn cục (global interpretability) định lượng mức độ quan trọng của đặc trưng trên toàn bộ tập dữ liệu. “TabNet” đưa ra giá trị thể hiện mức độ quan trọng của đặc trưng vào kết quả quyết định cuối cùng.

3.6 Cách xử lý dữ liệu dạng categorical

Đối với dữ liệu dạng categorical, cách xử lý thông thường là sử dụng mã hóa “one-hot”. Tuy nhiên, khi các cột categorical có nhiều giá trị khác nhau thì việc mã hóa “one-hot” sẽ dẫn đến số lượng đặc trưng tăng lên rất nhiều. Để khắc phục điều này, “TabNet” sử dụng kỹ thuật “embedding” để xử lý dữ liệu categorical. Kỹ thuật “embedding” cũng có phần tương tự mã hóa “one-hot” đó là từ một cột sẽ tạo ra được nhiều cột khác nhau, tuy nhiên nó hiệu quả hơn “one-hot” bởi số cột nó tạo ra là ít hơn và cũng sẽ thể hiện được sự tương đồng giữa các giá trị khác nhau tốt hơn.

Đầu tiên, các cột dữ liệu có dạng categorical sẽ được chuẩn hóa sang

dạng số (ví dụ, bằng cách sử dụng “LabelEncoder”, ...). Các số này sẽ được làm đầu vào cho quá trình “embedding”. Quá trình “embedding” sẽ diễn ra như sau:

- Khởi tạo một ma trận trọng số (ma trận “embedding”).
- Ma trận trọng số sau đó sẽ được học dựa trên mối quan hệ sẵn có của tập dữ liệu.
- Sử dụng các số đầu vào để lấy ra dòng/cột tương ứng của ma trận trọng số (dòng/cột là do quy ước), ta sẽ được véc-tơ đầu ra sau khi “embedding”.

Chương 4

Thí nghiệm

Chương này trình bày về các kết quả thí nghiệm để đánh giá mô hình “TabNet” trên nhiều khía cạnh (thực hiện tác vụ hồi quy, thực hiện tác vụ phân loại, khả năng diễn giải các dự đoán, khả năng lựa chọn đặc trưng, ...). Các thí nghiệm được thực hiện trên các tập dữ liệu dạng bảng khác nhau: tập dữ liệu giả lập Synthetic và các bộ dữ liệu trong thế giới thực: Sarcos, Rossmann Store Sales. Kết quả thí nghiệm cho thấy mô hình do chúng em cài đặt lại đạt được kết quả xấp xỉ so với kết quả được công bố trong bài báo [1] trên tập Synthetic và tập Rossmann Store Sales. Thêm vào đó, chúng em còn thực hiện thêm các thí nghiệm để so sánh “TabNet” và “XGBoost” trên nhiều phương diện, và thí nghiệm trong nội bộ mô hình “TabNet” để rút ra được ảnh hưởng của một số siêu tham số.

4.1 Tập dữ liệu sử dụng

Chúng em tiến hành thí nghiệm trên các tập dữ liệu dạng bảng vừa và lớn ở các lĩnh vực khác nhau, để đánh giá các khía cạnh khác nhau của mô hình “TabNet”. Cụ thể, về các tập dữ liệu như sau:

Synthetic datasets [2]

Đây là 6 tập dữ liệu dạng bảng, mỗi tập bao gồm 10000 mẫu huấn luyện và 10000 mẫu kiểm tra, được tạo ra bằng cách một tập con đặc trưng xác định kết quả đầu ra:

- Từ Syn1 đến Syn3: các đặc trưng quan trọng không thay đổi, giống nhau trong mọi trường hợp (ví dụ, đầu ra của Syn2 phụ thuộc vào những đặc trưng X_3 đến X_6).
- Đối với Syn4 đến Syn6: các đặc trưng quan trọng phụ thuộc vào từng trường hợp (ví dụ, đầu ra của Syn4 phụ thuộc vào X_1 đến X_2 hoặc X_3 đến X_6 tùy thuộc vào giá trị của X_{11}), điều này làm cho việc lựa chọn đặc trưng toàn cục không còn tối ưu.

Có 11 đặc trưng đầu vào, được tạo theo phân phối Gaussian và không có mối tương quan giữa các đặc trưng ($X \sim \mathcal{N}(0, I)$). Nhãn Y được lấy mẫu dưới dạng biến ngẫu nhiên Bernoulli với $\mathbb{P}(Y = 1|X) = \frac{1}{1+\text{logit}(X)}$, trong đó $\text{logit}(X)$ có thể thay đổi để tạo ra 3 bộ dữ liệu khác nhau:

- Syn1: $\exp(X_1 X_2)$.
- Syn2: $\exp(\sum_{i=3}^6 X_i^2 - 4)$.
- Syn3: $-10 \times \sin 2X_7 + 2|X_8| + X_9 + \exp(-X_{10})$.

Với mục đích đánh giá thêm về khả năng lựa chọn đặc trưng trên từng mẫu, nhóm tác giả tạo ra thêm 3 bộ dữ liệu:

- Syn4: Nếu $X_{11} < 0$ thì logit theo Syn1, ngược lại logit theo Syn2.
- Syn5: Nếu $X_{11} < 0$ thì logit theo Syn1, ngược lại logit theo Syn3.
- Syn6: Nếu $X_{11} < 0$ thì logit theo Syn2, ngược lại logit theo Syn3.

Sarcos [8]

Tập dữ liệu bao gồm 44484 mẫu huấn luyện và 4449 mẫu kiểm tra với 21 cột đặc trưng dùng để dự đoán (7 vị trí khớp, 7 vận tốc khớp, 7 gia tốc khớp) và 7 cột dùng làm mục tiêu (tương đương với 7 mô men). Mục tiêu là ước lượng mô men xoắn của một động cơ robot để đáp ứng với véc-tơ đầu vào. Một vấn đề với tập dữ liệu này đó là hầu hết thông tin của tập test đã có trên tập train [7]. Vì vậy, dù đây là tập được sử dụng trong bài báo gốc [1] để đánh giá hiệu suất của “TabNet” so với các mô hình khác trên tập test, nhưng chúng em chỉ sử dụng để so sánh về độ lỗi với “XGBoost” trên tập validation được tách ra từ tập train.

Rossmann Store Sales [6]

Tập dữ liệu bao gồm hơn 1 triệu mẫu với 17 cột đặc trưng. Nhiệm vụ là dự báo doanh số bán hàng của cửa hàng từ các đặc trưng tĩnh và thay đổi theo thời gian. Đối với tập dữ liệu này chúng em sử dụng tiền xử lý như đề xuất trong bài báo gốc, lấy dữ liệu năm 2014 làm tập huấn luyện, dữ liệu năm 2015 làm tập kiểm tra. Và sau khi tiền xử lý, tập dữ liệu bao gồm 373855 mẫu huấn luyện và 236380 mẫu kiểm tra với 31 cột đặc trưng. Trong đó, có 21 cột dạng categorical và 10 cột dạng số.

4.2 Các thiết lập thí nghiệm

Để đánh giá hiệu quả của mô hình, chúng em chia tập train của các tập dữ liệu thành 2 tập: train và validation theo tỉ lệ 9:1, riêng với tập Rossmann Store Sales thì chia 100k mẫu cho tập validation. Và độ đo được sử dụng để đánh giá là “AUC” đối với 6 tập Synthetic, “MSE” đối với tập Sarcos và “RMSE” đối với tập Rossmann Store Sales (chi tiết về các độ đo được trình bày trong phần phụ lục A).

Đối với mô hình “TabNet”, chúng em thiết lập các tham số mô hình như sau:

- **Synthetic:** Với toàn bộ 6 tập đều sử dụng các tham số $N_d = N_a =$

16, $n_shared = n_independent = 2$, Adam với learning rate = 0.02, $B = 3000$, $B_V = 100$, $m_B = 0.3$, max_epoch = 500 và StepLR với gamma = 0.7, step_size = 80. Với Syn1 sử dụng $\lambda_{sparse} = 0.02$, $N_{steps} = 4$, $\gamma = 2.0$; Syn2 và Syn3 sử dụng $\lambda_{sparse} = 0.01$, $N_{steps} = 4$, $\gamma = 2.0$; và Syn4, Syn5, Syn6 sử dụng $\lambda_{sparse} = 0.005$, $N_{steps} = 5$, $\gamma = 1.5$.

- **Sarcos:** $N_d = N_a = 64$, $n_shared = n_independent = 2$, $\lambda_{sparse} = 0.0001$, $N_{steps} = 5$, $\gamma = 1.5$, Adam với learning rate = 0.02, $B = 4096$, $B_V = 128$, $m_B = 0.2$, max_epoch = 300 và StepLR với gamma = 0.9, step_size = 400.
- **Rossmann Store Sales:** $N_d = N_a = 28$, $n_shared = n_independent = 2$, $\lambda_{sparse} = 0.00002$, $N_{steps} = 6$, $\gamma = 1$, Adam với learning rate = 0.02, $B = 4096$, $B_V = 512$, $m_B = 0.8$, max_epoch = 300 và ReduceLR với mode = min, patience = 4, min_lr = 1e-5, factor = 0.5.

Trong đó, N_d là kích thước đầu vào của khối quyết định, N_a là kích thước đầu vào của khối Attentive Transformer, B là kích thước batch (batch size), B_V là kích thước mini-batch, m_B là momentum, n_shared là số khối GLU dùng chung cho các bước quyết định, $n_independent$ là số khối GLU dùng riêng cho mỗi bước quyết định, max_epoch là số epoch huấn luyện, λ_{sparse} là hệ số để thêm “chính quy hóa thưa thớt” L_{sparse} vào hàm mất mát tổng thể, N_{steps} là số bước quyết định, γ là tham số “relaxation” để kiểm soát việc chọn lựa các đặc trưng.

Đối với mô hình “XGBoost”, chúng em sử dụng kỹ thuật “Grid Search” để điều chỉnh tham số. Các tham số mà chúng em điều chỉnh là: learning_rate: [1e-3, 1e-2, 0.1, 0.3], max_depth: [3, 5, 10], n_estimators: [50, 100, 500, 1000].

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 em trình bày kết quả của mô hình mà chúng em cài đặt. Cụ thể, sẽ so sánh kết quả mô hình “TabNet” mà chúng em cài đặt với kết quả trong bài báo. Mô hình được đánh giá trên tập dữ liệu Synthetic và tập Rossmann Store Sales với các thiết lập như đã nêu ở phần 4.2.

Dựa vào kết quả ở bảng 4.1, có thể thấy ở tập dữ liệu Synthetic, mô hình mà chúng em cài đặt đạt được kết quả nhỉnh hơn một chút so với kết quả trong bài báo.

Ngoài khả năng xử lý các bài toán phân loại, “TabNet” còn có thể xử lý các bài toán hồi quy và để kiểm tra cho điều này nhóm em tiến hành thí nghiệm trên tập dữ liệu Rossmann Store Sales. Kết quả độ lỗi RMSE trong bảng 4.2 cho thấy mô hình mà chúng em cài đặt cho kết quả kém hơn một chút so với bài báo gốc, nhưng không đáng kể.

Bảng 4.1: Kết quả cài đặt trên tập Synthetic so với kết quả trong bài báo [1]

Test AUC						
	Syn1	Syn2	Syn3	Syn4	Syn5	Syn6
TabNet	0.691	0.896	0.902	0.800	0.810	0.885
TabNet (cài đặt của tác giả)	0.682	0.892	0.897	0.776	0.789	0.878

Bảng 4.2: Kết quả cài đặt trên tập dữ liệu Rossmann Store Sales so với kết quả trong bài báo [1]

	Test RMSE
TabNet	488
TabNet (cài đặt của tác giả)	485.12

4.3.2 “TabNet” so với “XGBoost”

Như đã trình bày rõ trong phần 2.2, “XGBoost” là một thuật toán học máy rất mạnh mẽ đối với bài toán học có giám sát với dữ liệu dạng bảng, và nó đã dành được chiến thắng trong rất nhiều cuộc thi trên Kaggle. Vì vậy, trong phần này chúng em trình bày các thí nghiệm thêm để xem xét thử so với “XGBoost” thì “TabNet” sẽ hơn được những mặt nào và thua những mặt nào.

a. Khả năng lựa chọn đặc trưng

Bảng 4.3 cho thấy kết quả AUC trên tập Synthetic của mô hình “TabNet” và “XGBoost” ngang ngang nhau, “TabNet” có nhỉnh hơn một chút nhưng không đáng kể. Tuy nhiên, thời gian huấn luyện của “XGBoost” là nhanh hơn nhiều so với “TabNet”.

Bảng 4.3: Kết quả trên tập Synthetic của “TabNet” và “XGBoost”

Test AUC							Runtime(s)
	Syn1	Syn2	Syn3	Syn4	Syn5	Syn6	
TabNet	0.691	0.896	0.902	0.800	0.810	0.885	300
XGBoost	0.675	0.895	0.900	0.785	0.784	0.880	20

Với các đặc điểm của tập dữ liệu Synthetic như đã trình bày trong phần 4.1, có thể thấy đây là một tập dữ liệu rất tốt để kiểm tra khả năng

lựa chọn đặc trưng của các mô hình (vì đầu ra chỉ phụ thuộc vào một số cột đầu vào). Và cả hai mô hình “TabNet”, “XGBoost” đều có khả năng lựa chọn các đặc trưng mạnh mẽ. Quan sát hình 4.1 và hình 4.2, đây là tầm quan trọng của các đặc trưng trên tổng thể được đưa ra bởi “TabNet” và “XGBoost”. Có thể thấy hai mô hình đều đưa ra đúng các đặc trưng quan trọng theo như cách tạo ra tập dữ liệu. Cụ thể:

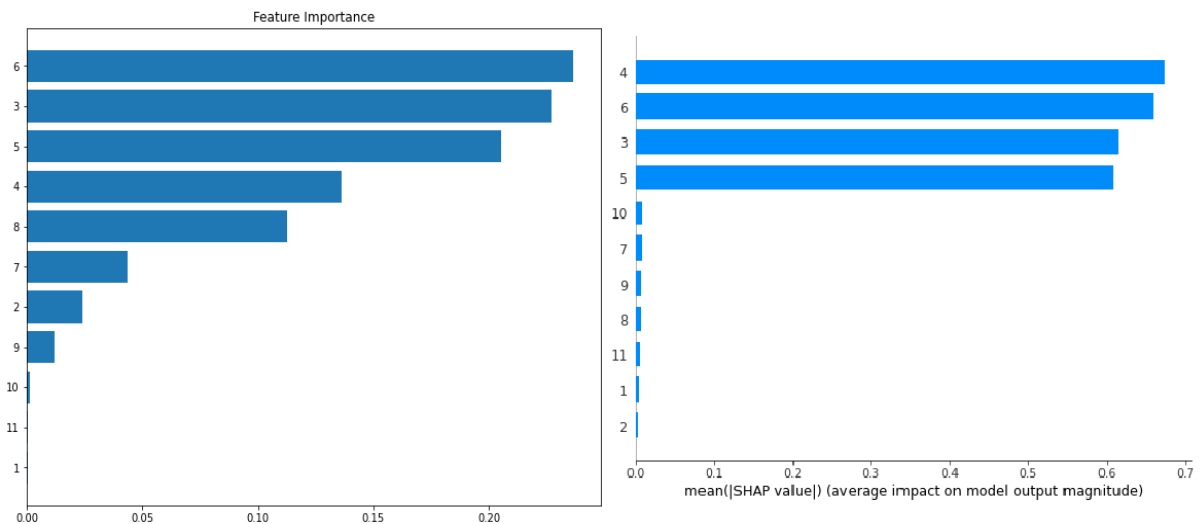
- Đối với tập synthetic 2: Theo cách tạo ra dữ liệu, syn2 sẽ phụ thuộc vào các đặc trưng X_3 đến X_6 , hai mô hình đều xếp hạng 4 đặc trưng này là quan trọng nhất.
- Đối với tập synthetic 4: Theo cách tạo ra dữ liệu, syn4 sẽ phụ thuộc vào X_1 đến X_2 hoặc X_3 đến X_6 tùy thuộc vào giá trị của X_{11} . Hai mô hình đều xếp hạng các đặc trưng này có tầm quan trọng cao.

Thêm vào đó, cả “TabNet” và “XGBoost” đều có thể lựa chọn đặc trưng khác nhau cho từng đầu vào cụ thể (instance-wise feature selection). Hình 4.4 và hình 4.5 thể hiện cho điều này. Trên các mẫu đều phụ thuộc chủ yếu vào các đặc trưng X_1 đến X_2 , X_3 đến X_6 và X_{11} .

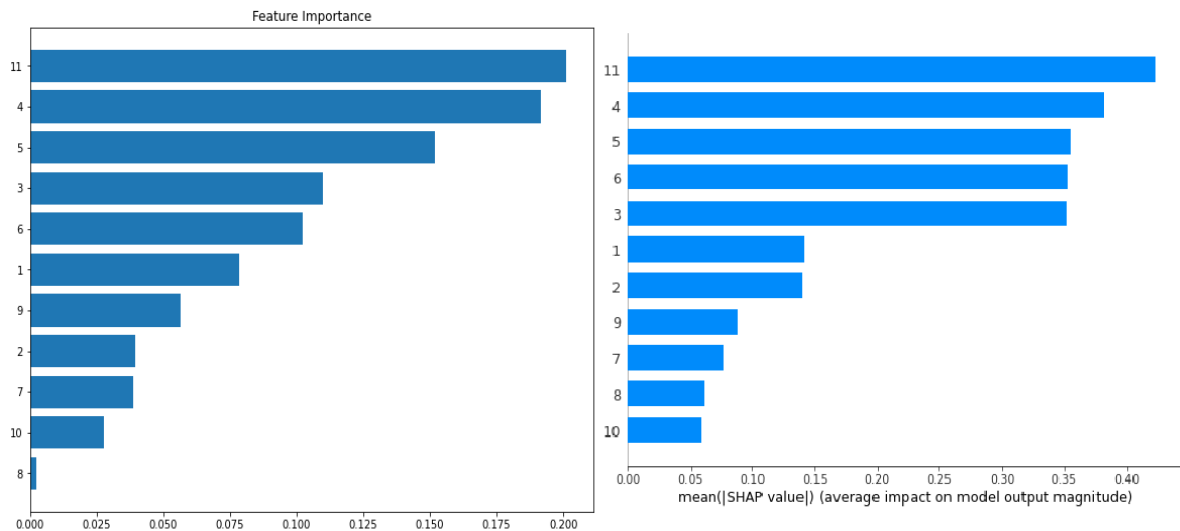
Tập Synthetic là tập dữ liệu giả lập, vì vậy để kiểm chứng rõ hơn về khả năng lựa chọn đặc trưng của hai mô hình trong thực tế thì nhóm em thực hiện thêm một thí nghiệm trên tập Rossmann Store Sales. Hình 4.3 cho thấy hai mô hình đưa ra lựa chọn các đặc trưng quan trọng khác nhau. Tuy nhiên, các đặc trưng như: Store, DayOfWeek, Customers, Promo đều được cả hai mô hình xếp hạng tầm quan trọng cao hoặc khá cao. Và theo nhìn nhận cảm tính của chúng em thì điều này là khá hợp lý. Bởi vì, doanh thu của cửa hàng sẽ phụ thuộc vào cửa hàng đó là cửa hàng nào, ngày đó là ngày nào trong tuần vì thường vào cuối tuần mọi người có xu hướng đi mua sắm nhiều hơn, lượng khách hàng đến trong ngày cũng ảnh hưởng lớn đến doanh thu, và cuối cùng khi cửa hàng có chương trình khuyến mãi có thể giúp đẩy doanh thu cao hơn.

Như đã trình bày ở các phần trước, “TabNet” ngoài khả năng diễn giải trên toàn cục, nó cũng có thể diễn giải trên cục bộ, đưa ra tầm quan trọng của các đặc trưng trên từng mẫu. Còn đối với “XGBoost” thì chỉ có thể đưa ra tầm quan trọng trên toàn cục. Để đưa ra được tầm quan trọng trên cục bộ như hình 4.5 thì cần phải dùng thêm các thư viện hỗ trợ, cụ thể chúng em sử dụng thêm “SHAP” (SHapley Additive exPlanations).

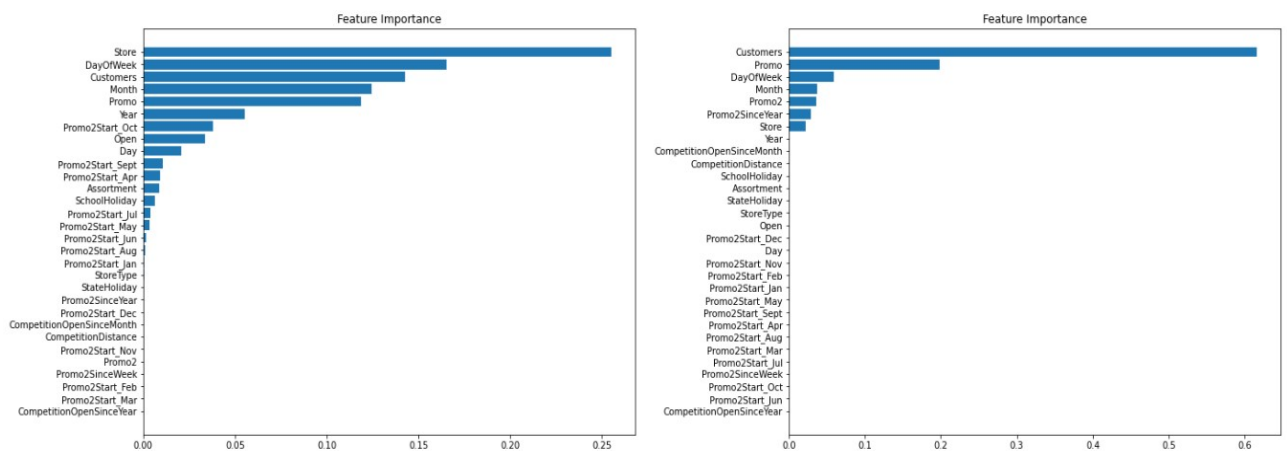
Như vậy, qua kết quả thí nghiệm trên giúp ta thấy được rằng khả năng lựa chọn đặc trưng của “TabNet” và “XGBoost” là tương tự nhau và rất mạnh mẽ. Tuy nhiên, “TabNet” tiện hơn trong việc đưa ra diễn giải trên cục bộ, giúp chúng ta hiểu hơn về các quyết định, dự đoán của mô hình.



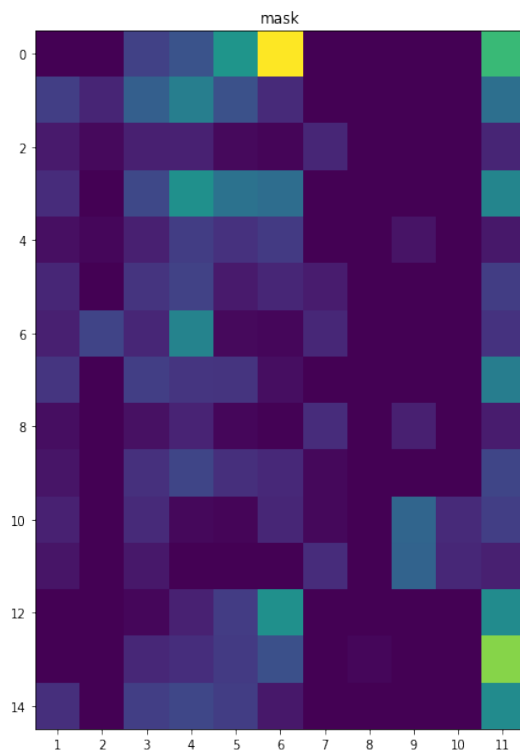
Hình 4.1: Tầm quan trọng của các đặc trưng trên tập Syn2 (trái: “TabNet”, phải: “XGBoost”). Ở hai hình, trục tung thể hiện cho các đặc trưng từ X_1 đến X_{11} , trục hoành thể hiện cho mức độ quan trọng của đặc trưng.



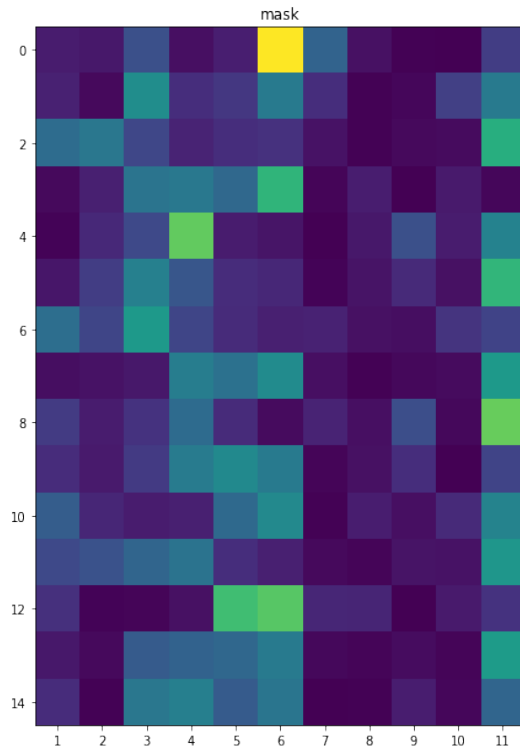
Hình 4.2: Tầm quan trọng của các đặc trưng trên tập Syn4 (trái: “TabNet”, phải: “XGBoost”). Ở hai hình, trục tung thể hiện cho các đặc trưng từ X_1 đến X_{11} , trục hoành thể hiện cho mức độ quan trọng của đặc trưng.



Hình 4.3: Tầm quan trọng của các đặc trưng trên tập Rossmann Store Sales (trái: “TabNet”, phải: “XGBoost”). Ở hai hình, trục tung thể hiện cho các đặc trưng từ X_1 đến X_{11} , trục hoành thể hiện cho mức độ quan trọng của đặc trưng.



Hình 4.4: Tầm quan trọng trên cục bộ của các đặc trưng đối với tập Syn4 biểu diễn bằng “TabNet”. Hình thể hiện cho tầm quan trọng của 11 đặc trưng trên 15 mẫu đầu tiên của tập test. Trục tung biểu diễn cho 15 mẫu, trục hoành biểu diễn cho 11 đặc trưng.



Hình 4.5: Tầm quan trọng trên cục bộ của các đặc trưng đối với tập Syn4 biểu diễn bằng “XGBoost”. Hình thể hiện cho tầm quan trọng của 11 đặc trưng trên 15 mẫu đầu tiên của tập test. Trục tung biểu diễn cho 15 mẫu, trục hoành biểu diễn cho 11 đặc trưng.

b. Xử lý các trường hợp multi-output

Như đã trình bày trong phần 4.1, Sarcos là một tập dữ liệu có nhiều output, cụ thể ở đây là 7 cột output. Kết quả trong bảng 4.4 cho thấy giá trị MSE trên tập validation được tách ra từ tập train và thời gian huấn luyện của hai mô hình là ngang nhau.

Về mặt lý thuyết, với những trường hợp multi-output, ở mỗi bước “XGBoost” sẽ phải xây dựng nhiều cây, mỗi cây tương ứng với một output. Trong khi đó, đối với “TabNet”, vì bản chất là một mô hình học sâu nên nó sẽ xử lý dễ dàng hơn và linh hoạt hơn trong trường hợp này. Không những vậy, tại vì phải xây dựng nhiều cây ở mỗi bước nên khi tập dữ liệu có nhiều output, thời gian huấn luyện của “XGBoost” sẽ tăng lên rất nhiều, còn thời gian huấn luyện của “TabNet” có thể cũng sẽ tăng

nhưng tăng nhẹ hơn so với “XGBoost”. Để kiểm chứng cho điều này, chúng em thực hiện một thí nghiệm trên tập dữ liệu Sarcos. Cụ thể như sau: giữ nguyên các siêu tham số của “XGBoost” và “TabNet” như thí nghiệm đánh giá MSE trên tập validation, lần lượt thay đổi số lượng cột output là 1, 3, 5, 7. Sau đó, thực hiện việc huấn luyện với từng trường hợp để theo dõi sự thay đổi thời gian huấn luyện của “XGBoost” và “TabNet”. Quan sát bảng 4.5, kết quả thí nghiệm cho thấy, khi tăng số lượng cột output lên thì thời gian huấn luyện của “XGBoost” càng ngày càng tăng, còn thời gian huấn luyện của “TabNet” thì hầu như không thay đổi.

Như vậy, kết quả thí nghiệm là đúng với lý thuyết, khi số lượng output tăng lên thì thời gian huấn luyện của “TabNet” không tăng lên nhiều bằng “XGBoost”. Do đó, với một số lượng output lớn thì rất có thể “TabNet” sẽ huấn luyện nhanh hơn “XGBoost”.

Bảng 4.4: Kết quả trên tập Sarcos của “TabNet” và “XGBoost”

	Valid MSE	Runtime (s)
TabNet	11.2	225
XGBoost	11.3	204

Bảng 4.5: Thời gian chạy trên tập Sarcos của “TabNet” và “XGBoost” khi thay đổi số lượng output

	Runtime (s)			
Số lượng output	1	3	5	7
TabNet	224	220	227	225
XGBoost	34s	94	147	204

c. Xử lý trường hợp dữ liệu dạng categorical có nhiều giá trị khác nhau

Như đã trình bày ở phần 4.1, Rossmann Store Sales sau khi được tiền xử lý bao gồm 31 thuộc tính trong đó có 21 thuộc tính thuộc dạng categorical. Và cột categorical đáng chú ý nhất đó là cột “Store”, nó bao gồm 1115 giá trị riêng biệt. Đây cũng chính là một thách thức lớn đối với dữ liệu dạng bảng như đã trình bày trong chương 1. Để xử lý dữ liệu dạng categorical, “TabNet” sử dụng “embedding” - một cách làm được đánh giá là tốt hơn nhiều so với cách xử lý thông thường là “one-hot”. Còn đối với “XGBoost”, có hỗ trợ hai cách xử lý đó là: (1) mã hóa “one-hot”, (2) phân tách, gom các giá trị khác nhau thành các nhóm để dự đoán. Có thể thấy, đối với trường hợp cột categorical có nhiều giá trị khác nhau thì cách xử lý thứ 2 của “XGBoost” có lẽ sẽ tốt hơn, tuy nhiên tính năng này vẫn đang được thử nghiệm và chưa ổn định. Vì vậy, ở thí nghiệm này chúng em sẽ so sánh kết quả của “TabNet” (xử lý bằng “embedding”), và “XGBoost” (xử lý theo cách thông dụng là sử dụng “one-hot”). Kết quả ở bảng 4.6 cho thấy độ lỗi RMSE trên tập kiểm tra của “TabNet” nhỏ hơn nhiều so với “XGBoost”. Điều này, chứng tỏ trong trường hợp cột categorical có quá nhiều giá trị riêng biệt thì cách xử lý “embedding” tốt hơn nhiều so với “one-hot”. Như vậy, “TabNet” hoạt động hiệu quả hơn “XGBoost” trong trường hợp này (vì cách làm tốt hơn của “XGBoost” vẫn đang trong quá trình thử nghiệm).

Bảng 4.6: Kết quả trên tập Rossmann Store Sales của “TabNet” và “XGBoost”

	Test RMSE	Runtime (s)
TabNet	488	482
XGBoost	1501	215

⇒ **Kết luận:** Như vậy qua những thí nghiệm trên, có thể đưa ra được các kết luận sau về “TabNet” và “XGBoost”:

- Trong đa số trường hợp “TabNet” và “XGBoost” có độ chính xác dự đoán ngang ngửa nhau, tuy nhiên thời gian huấn luyện của “XGBoost” sẽ nhanh hơn.
- Cả hai mô hình đều có khả năng lựa chọn các đặc trưng quan trọng mạnh mẽ, có khả năng lựa chọn các đặc trưng khác nhau cho từng đầu vào cụ thể (instance-wise feature selection).
- Cả hai mô hình đều có khả năng đưa ra diễn giải trên toàn cục, và “TabNet” thuận tiện hơn trong việc đưa ra diễn giải trên cục bộ giúp chúng ta hiểu rõ về các dự đoán của mô hình, trong khi “XGBoost” phải sử dụng thêm các thư viện hỗ trợ mới có thể đưa ra diễn giải trên cục bộ.
- Vì là một kiến trúc mô hình học sâu nên đối với các trường hợp multi-output, “TabNet” sẽ xử lý tiện hơn vì dễ dàng điều chỉnh, mở rộng kiến trúc mạng (ví dụ, để dự đoán nhiều output thì chỉ cần thêm các nơ-ron vào tầng output). Và khi số lượng output đủ lớn thì rất có thể “TabNet” sẽ huấn luyện nhanh hơn “XGBoost”.
- Đối với trường hợp các cột categorical có nhiều giá trị riêng biệt thì “TabNet” hoạt động hiệu quả hơn nhờ vào việc sử dụng “embedding” thay vì “one-hot” như “XGBoost” (vì cách làm tốt hơn của “XGBoost” còn đang thử nghiệm).

4.3.3 Ảnh hưởng của các siêu tham số trong “TabNet”

Trong phần này, chúng em trình bày các thí nghiệm để xác định sự ảnh hưởng của các siêu tham số trong “TabNet”. Cụ thể là các siêu tham số: n_d , n_a và λ_{sparse} . Các siêu tham số này ảnh hưởng lớn đến kết quả của mô hình. Để làm rõ điều này, chúng em đã thực hiện một số thí nghiệm trên tập Syn1 và kết quả của nó như sau:

a. Ảnh hưởng của siêu tham số λ_{sparse}

Như đã trình bày ở chương 3, λ_{sparse} là hệ số để kiểm soát thêm về mức độ thừa thớt của các đặc trưng đã chọn, cụ thể nó sẽ kiểm soát việc ở mỗi bước quyết định có bao nhiêu đặc trưng được chọn. Vì vậy, khi λ_{sparse} thay đổi thì số lượng đặc trưng được chọn ở mỗi bước cũng thay đổi, cụ thể khi λ_{sparse} tăng lên thì số lượng đặc trưng được chọn ở mỗi bước giảm xuống, điều này sẽ làm cho độ chính xác trên tập huấn luyện giảm xuống \rightarrow khi λ_{sparse} lớn thì rất có thể mô hình của ta sẽ bị “chưa khớp dữ liệu” (underfitting). Để kiểm chứng điều này, chúng em thực hiện thí nghiệm trên tập Syn1, giữ nguyên các siêu tham số, thay đổi giá trị $\lambda_{sparse} = [0, 0.1, 100]$, và kết quả thu được như bảng 4.7 và bảng 4.8. Dựa vào bảng 4.8 có thể thấy khi λ_{sparse} tăng lên thì trung bình số lượng đặc trưng được chọn ở mỗi bước sẽ giảm xuống, điều này kéo theo loss ở tập train tăng lên và AUC ở tập train bị giảm xuống (bảng 4.7). Và khi $\lambda_{sparse} = 100$ mô hình của chúng em đã bị rơi vào trường hợp “chưa khớp dữ liệu”.

Bảng 4.7: Kết quả AUC và loss ở epoch 199 trên tập train và validation của Syn1 đối với các giá trị λ_{sparse} khác nhau

λ_{sparse}	loss	train_auc	val_auc
0	0.61776	0.70777	0.66158
0.1	0.63957	0.68822	0.66648
100	1.66168	0.60463	0.56495

Bảng 4.8: Trung bình số lượng đặc trưng được chọn ở mỗi bước trên tập train của Syn1 đối với các giá trị λ_{sparse} khác nhau

λ_{sparse}	step1	step2	step3	step4
0	2.38	1.18	1.06	1.22
0.1	1.64	1.16	1.02	1.04
100	1.05	1.01	1.00	1.01

b. Ảnh hưởng của siêu tham số n_d và n_a

Khi các siêu tham số này quá nhỏ nghĩa là mô hình của ta quá đơn giản (“capacity” của mô hình thấp), điều này có thể làm cho mô hình bị “chưa khớp dữ liệu” (underfitting); khi các siêu tham số này quá lớn nghĩa là mô hình của ta quá phức tạp (“capacity” của mô hình cao), điều này có thể làm mô hình bị “quá khớp dữ liệu” (overfitting). Để kiểm chứng điều này, chúng em thực hiện thí nghiệm trên tập Syn1, giữ nguyên các siêu tham số, thay đổi giá trị $n_d = n_a = [2, 4, 6, 8, 10, 16]$. Qua kết quả trong bảng 4.9, có thể thấy ở trường hợp $n_d = n_a = 2$ mô hình của chúng em đã bị “chưa khớp dữ liệu”. Hơn nữa, khi tăng n_d và n_a từ 2 đến 8 thì loss giảm xuống và train_auc tăng lên (đồng thời val_auc cũng tăng lên), khi tiếp tục tăng n_d và n_a lên 10 và 16 thì loss không tiếp tục giảm và train_auc không tiếp tục tăng; điều này có thể là do việc cực tiểu hóa loss chưa đủ tốt (vì mô hình càng phức tạp thì việc cực tiểu hóa loss càng khó khăn). Vì vậy mà ở thí nghiệm này chúng em vẫn chưa thấy được trường hợp bị “quá khớp dữ liệu”.

Bảng 4.9: Kết quả AUC và loss ở epoch 199 trên tập train và validation của Syn1 đối với các giá trị n_d và n_a khác nhau

$n_d = n_a$	loss	train_auc	val_auc
2	0.69481	0.51266	0.50343
4	0.63613	0.68974	0.66011
6	0.6238	0.70604	0.67101
8	0.61544	0.71829	0.67119
10	0.62354	0.70509	0.67768
16	0.62658	0.70372	0.66259

\Rightarrow **Kết luận:** Như vậy qua các thí nghiệm trên, có thể thấy được ảnh hưởng của các siêu tham số n_d , n_a và λ_{sparse} đến hiệu quả của mô hình. Vì vậy, khi lựa chọn tham số cần phải cân nhắc lựa chọn các giá trị phù hợp. Bài báo gốc [1] đề xuất các giá trị của n_d và n_a nên là $\{8, 16, 24, 32, 64, 128\}$, và lúc chọn nên chọn $n_d = n_a$. Còn λ_{sparse} nên chọn các giá trị $\{0, 0.000001, 0.0001, 0.001, 0.01, 0.1\}$.

Chương 5

Tổng kết và hướng phát triển

5.1 Tổng kết

Trong khóa luận này, chúng em nghiên cứu về bài toán học có giám sát với dữ liệu dạng bảng bằng mô hình học sâu. Cụ thể, chúng em tập trung tìm hiểu mô hình “TabNet” được đề cập trong bài báo [1]. Phương pháp chọn đặc trưng của “TabNet” là chọn các đặc trưng khác nhau đối với mỗi đầu vào cụ thể (instance-wise). Một mô hình “TabNet” có thể bao gồm nhiều bước quyết định. Trong mỗi bước quyết định, hai khối cấu trúc quan trọng nhất là Attentive Transformer và Feature Transformer. Khối Attentive Transformer sẽ hỗ trợ quá trình chọn đặc trưng. Khối Feature Transformer thực hiện xử lý những đặc trưng được chọn nhằm rút trích thông tin, đưa ra output. Kết quả cuối cùng của “TabNet” sẽ được tổng hợp bởi kết quả đầu ra của các bước quyết định. “TabNet” có khả năng diễn giải cục bộ dễ dàng bằng cách trực quan các Mask ở mỗi bước quyết định để giải thích những đặc trưng được chọn ở mỗi mẫu dữ liệu và diễn giải toàn cục bằng cách định lượng mức độ quan trọng của đặc trưng trên toàn bộ tập dữ liệu.

Chúng em tìm hiểu mô hình và thực hiện cài đặt lại. Các tập dữ liệu chúng em sử dụng để kiểm tra mô hình là bộ dữ liệu Synthetic, Sarcos và Rossmann Store Sales. Để đánh giá hiệu quả của mô hình, chúng em sử

dụng các độ đo: “AUC”, “MSE”, “RMSE”. Sau khi thực hiện lại những thí nghiệm, kết quả mang lại khá tương tự với kết quả được công bố trong bài báo đối với các tập dữ liệu Synthetic, Rossmann Store Sales. Tập dữ liệu Sarcos vì hầu hết thông tin trên tập test đã có ở tập train nên chúng em không sử dụng để so sánh với kết quả trong bài báo.

Chúng em thực hiện một số thí nghiệm nhằm so sánh mô hình mà chúng em cài đặt với mô hình trong bài báo [1]. Kết quả cho thấy mô hình mà chúng em cài đặt có kết quả tương đồng với bài báo. Ngoài ra, nhằm phân tích khả năng lựa chọn đặc trưng của mô hình, chúng em thực hiện so sánh mô hình “TabNet” với một mô hình cũng có khả năng lựa chọn đặc trưng là “XGBoost”. Cả hai mô hình đều cho thấy khả năng lựa chọn đặc trưng mạnh mẽ, kết quả đạt được cũng khá tương đồng nhau. Thí nghiệm trên còn cho thấy “TabNet” và “XGBoost” đều có khả năng lựa chọn đặc trưng toàn cục và cục bộ. Tiếp theo, chúng em thực hiện một thí nghiệm với dữ liệu có multi-output. Qua thí nghiệm cũng cho thấy “TabNet” xử lý mạnh mẽ và linh hoạt hơn “XGBoost”. Ta có thể dễ dàng thấy được để giải quyết những bài toán multi-output, đối với “TabNet” chỉ cần thêm nơ-ron ở tầng output, trong khi “XGBoost” phải xây dựng nhiều cây, mỗi cây sẽ đảm nhiệm một output. Và do đó, thời gian để “XGBoost” xử lý dạng bài toán này sẽ tăng đáng kể khi số lượng output tăng lên, còn “TabNet” thì tăng rất ít. Chúng em còn thực hiện một thí nghiệm để kiểm tra khả năng xử lý dữ liệu dạng categorical có nhiều giá trị khác nhau của “TabNet” và “XGBoost”. Trong thí nghiệm này, “TabNet” mang lại kết quả tốt hơn khi sử dụng “embedding” thay vì “one-hot” như XGBoost. Cuối cùng, để giúp hiểu rõ ảnh hưởng của các siêu tham số trong “TabNet” vào quá trình huấn luyện, chúng em đã thực hiện một số thí nghiệm để tìm hiểu thêm vấn đề này. Sau thí nghiệm, cho ta thấy n_d , n_a , λ_{sparse} có ảnh hưởng đến hiệu quả mô hình. Vì vậy, ta nên chọn tham số phù hợp cho mô hình.

5.2 Hướng phát triển

“TabNet” là một mô hình học sâu mạnh mẽ áp dụng cho bài toán học có giám sát, dữ liệu được đưa vào mô hình không cần phải tiền xử lý phức tạp. Nhưng với một mô hình học sâu, thì mô hình dễ bị “học quá khớp” (overfitting). Ngoài ra, quá trình rút trích đặc trưng cũng sẽ được yêu cầu đối với những bài toán phức tạp đòi hỏi độ chính xác cao hoặc độ lỗi thấp. Nhưng, quá trình “thiết kế đặc trưng thủ công” (feature engineering) rất tốn thời gian và công sức, cùng yêu cầu chuyên môn cao đối với người lập trình. Thêm vào đó, vì mô hình “TabNet” chỉ sử dụng dữ liệu có gán nhãn để giải những bài toán học có giám sát. Thực tế, dữ liệu có nhãn thường có ít nhưng dữ liệu không được gán nhãn thì có rất nhiều. Nhằm giúp cho mô hình giảm khả năng học quá khớp và huấn luyện tốt hơn, ta có thể tận dụng nguồn dữ liệu không được gán nhãn. Để có thể khai thác được nguồn dữ liệu không gán nhãn, một hướng tiếp cận là “học đặc trưng không giám sát” (unsupervised feature learning) học trên tập dữ liệu này. Đây là phương pháp sử dụng dữ liệu không gán nhãn để rút trích đặc trưng tự động. Thực tế, trong bài báo gốc nhóm tác giả cũng đã nghiên cứu về vấn đề này, tuy nhiên trong phạm vi khóa luận chúng em chưa đủ thời gian để tìm hiểu sâu về nó. Vậy nên, việc phát triển mô hình kết hợp với “học đặc trưng không giám sát” để cải thiện hiệu quả cho bài toán học có giám sát là một hướng nghiên cứu đầy tiềm năng trong tương lai.

Tài liệu tham khảo

- [1] Sercan O Arik and Tomas Pfister. “Tabnet: Attentive interpretable tabular learning”. In: *AAAI*. Vol. 35. 2021, pp. 6679–6687.
- [2] Jianbo Chen et al. “Learning to explain: An information-theoretic perspective on model interpretation”. In: *International Conference on Machine Learning*. PMLR. 2018, pp. 883–892.
- [3] Tianqi Chen and Carlos Guestrin. “Xgboost: A scalable tree boosting system”. In: *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. 2016, pp. 785–794.
- [4] Andrew Frank. “UCI machine learning repository”. In: (2010). URL: <http://archive.ics.uci.edu/ml>.
- [5] Yves Grandvalet and Yoshua Bengio. “Semi-supervised learning by entropy minimization”. In: *Advances in neural information processing systems* 17 (2004).
- [6] “Kaggle. 2019b. Rossmann Store Sales.” In: *Accessed: 2019-11-10*. (). URL: <https://www.kaggle.com/c/rossmann-store-sales>.
- [7] Online. *Running Code and Failing Models*. URL: <https://www.datarobot.com/blog/running-code-and-failing-models/> (visited on 02/10/2021).

- [8] Sethu Vijayakumar and Stefan Schaal. “Locally weighted projection regression: An $O(n)$ algorithm for incremental real time learning in high dimensional space”. In: *Proceedings of the seventeenth international conference on machine learning (ICML 2000)*. Vol. 1. Morgan Kaufmann. 2000, pp. 288–293.

Phụ lục A

Các độ đo đánh giá hiệu suất mô hình

A.1 Độ đo “Area Under the Receiving Operating Characteristic” (AUC)

Trước khi tìm hiểu về độ đo AUC, chúng em xin được trình bày độ đo receiver operating characteristic curve (ROC curve). ROC curve là một đường cong thể hiện hiệu suất phân lớp tại tất cả các ngưỡng phân lớp. Trong phân lớp nhị phân sẽ có hai lớp tương ứng là 'positive' và 'negative'. Một vài quy ước:

- “positive” : P.
- “negative” : N.
- true positive (TP): số mẫu “positive” được dự đoán đúng.
- true negative (TN): số mẫu “negative” được dự đoán đúng.
- false positive (FP): số mẫu “positive” dự đoán sai.
- false negative (FN): số mẫu “negative” dự đoán sai.

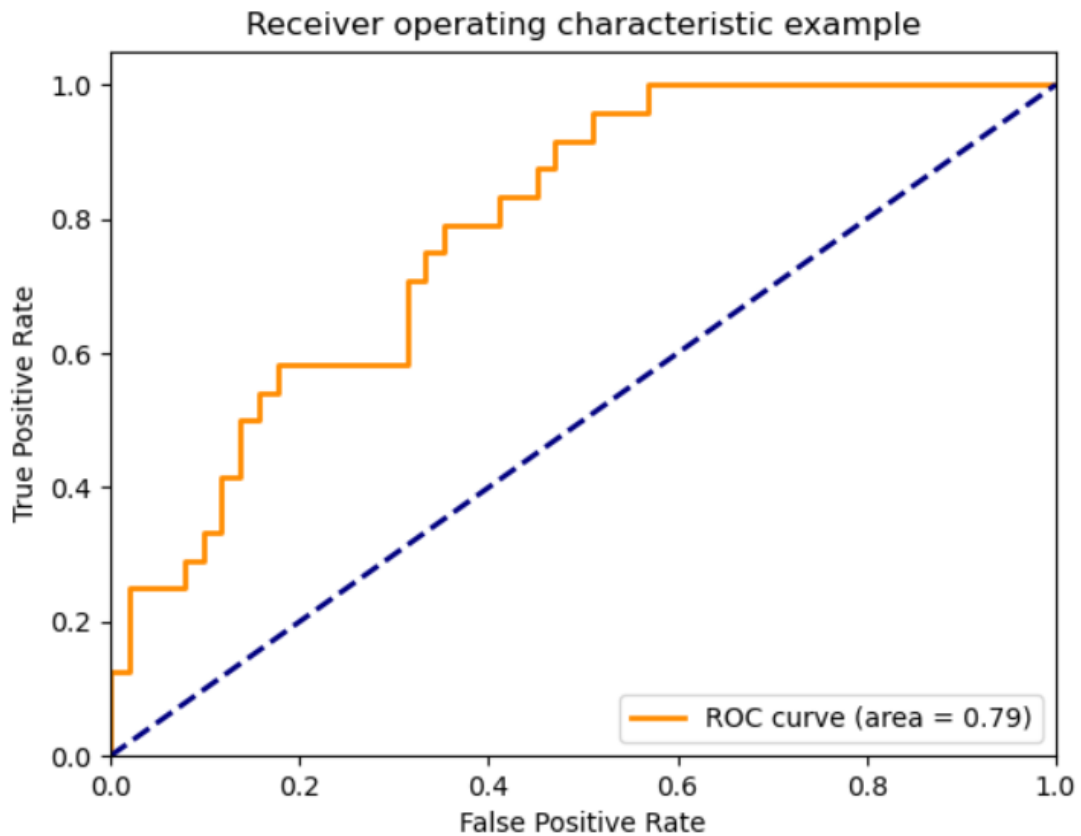
ROC curve được xác định trong một đồ thị hai chiều (hình A.1) như sau:

- True Positive Rate (TPR):

$$TPR = \frac{TP}{TP + FN} \quad (A.1)$$

- False Positive Rate (FPR):

$$FPR = \frac{TN}{TN + FP} \quad (A.2)$$



Hình A.1: Ví dụ đường cong ROC curve (Nguồn: scikit-learn.org)

AUC được tính bằng cách tính diện tích được bao quanh bởi đường cong ROC curve và trục hoành cận từ $[0, 0]$ đến $[0, 1]$. AUC thể hiện khả năng phân lớp của mô hình. Diện tích được tính sẽ có giá trị nằm trong

khoảng từ $[0, 1]$, khi diện tích càng lớn, giá trị AUC càng lớn càng gần về 1, mô hình phân lớp càng tốt.

A.2 Độ đo “Mean Squared Error” (MSE)

Sai số toàn phương trung bình (MSE - mean squared error) là một độ đo ước lượng trung bình của bình phương sai số - tức là tính trung bình bình phương chênh lệch giữa giá trị thực và giá trị tính toán. MSE là một hàm rủi ro, ước lượng giá trị dự đoán các giá trị kỳ vọng bao xa. Công thức tính MSE:

$$MSE^4 = \frac{1}{n} \sum_{i=1}^n (Y_i - \hat{Y}_i)^2 \quad (\text{A.3})$$

Trong đó:

- n : là số mẫu dữ liệu.
- Y_i : giá trị thực của mẫu i .
- \hat{Y}_i : giá trị dự đoán, giá trị được tính toán của mẫu i .

MSE là độ đo đo độ lỗi, chính vì vậy, một mô hình có giá trị MSE càng lớn thì mô hình càng không tốt, chứng tỏ giá trị dự đoán chênh lệch lớn với giá trị kỳ vọng. MSE có giá trị luôn luôn dương và lớn hơn 0.

A.3 Độ đo “Root Mean Square” (RMSE)

Độ đo “root mean square” (RMSE) dùng để tính sự khác biệt giữa giá trị dự đoán được bởi mô hình so với giá trị thực. RMSE ước lượng căn bậc

⁴Sai số toàn phương trung bình

hai của trung bình bình phương sai số. Công thức tính RMSE:

$$RMSE^5 = \sqrt{\frac{\sum_{i=1}^N (x_i - \hat{x}_i)^2}{N}} \quad (A.4)$$

Trong đó:

- i : là mẫu thứ i .
- N : là số mẫu dữ liệu.
- x_i : là giá trị thực.
- \hat{x}_i là giá trị dự đoán.

Cũng tương tự MSE, RMSE là độ đo đo độ lỗi. Khi giá trị được dự đoán có độ lỗi RMSE nhỏ, thì sự khác biệt của giá trị dự đoán và giá trị thực càng bé, mô hình càng có độ tin cậy cao.

⁵Đánh giá model trong Machine Learning

Phụ lục B

Thí nghiệm thêm

Ngoài các thí nghiệm trên các tập dữ liệu đã trình bày trong chương 4, chúng em còn thực hiện thêm các thí nghiệm trên tập dữ liệu Poker Hand [4]. Tập dữ liệu bao gồm 25k mẫu huấn luyện và 1 triệu mẫu kiểm tra, với 10 cột đặc trưng. Các đặc trưng ở dạng categorical (đã được mã hóa về dạng số). Mỗi mẫu là một ví dụ về một ván bài bao gồm 5 lá bài được lấy ra từ bộ bài 52 lá. Mỗi lá bài được mô tả bằng hai thuộc tính (suit và rank). Nhiệm vụ chính là phân loại ván bài poker từ hai thuộc tính của các lá bài.

Các thiết lập thí nghiệm: Chúng em chia tập train của tập dữ liệu thành 2 tập: train và validation theo tỉ lệ 9:1, và thiết lập các tham số mô hình như sau: $N_d = N_a = 128$, $n_shared = n_independent = 2$, $\lambda_{sparse} = 0.000001$, $N_{steps} = 4$, $\gamma = 1.5$, Adam với learning rate = 0.01, $B = 4096$, $B_V = 1024$, $m_B = 0.05$, max_epoch = 800 và StepLR với gamma = 0.95, step_size = 80.

B.1 Kết quả so với bài báo gốc [1]

Với các kết quả ở tập dữ liệu Poker Hand được thể hiện trong bảng B.1, mô hình mà chúng em cài đặt cho kết quả có phần sai lệch, kém hơn so với trong bài báo một chút. Vấn đề này có thể nằm ở chỗ kích thước

“embedding” cho các cột dữ liệu dạng categorical mà chúng em chọn sai khác so với cách chọn của nhóm tác giả. Tuy nhiên, vì số cột categorical lên tới 10 cột và mỗi cột có số phần tử riêng biệt lần lượt là [4, 13, 4, 13, 4, 13, 4, 13, 4, 13] nên việc tìm ra được bộ số tốt nhất là rất khó khăn. Dù vậy thì kết quả độ chính xác 96.7% trên dữ liệu thô cũng đã rất tốt.

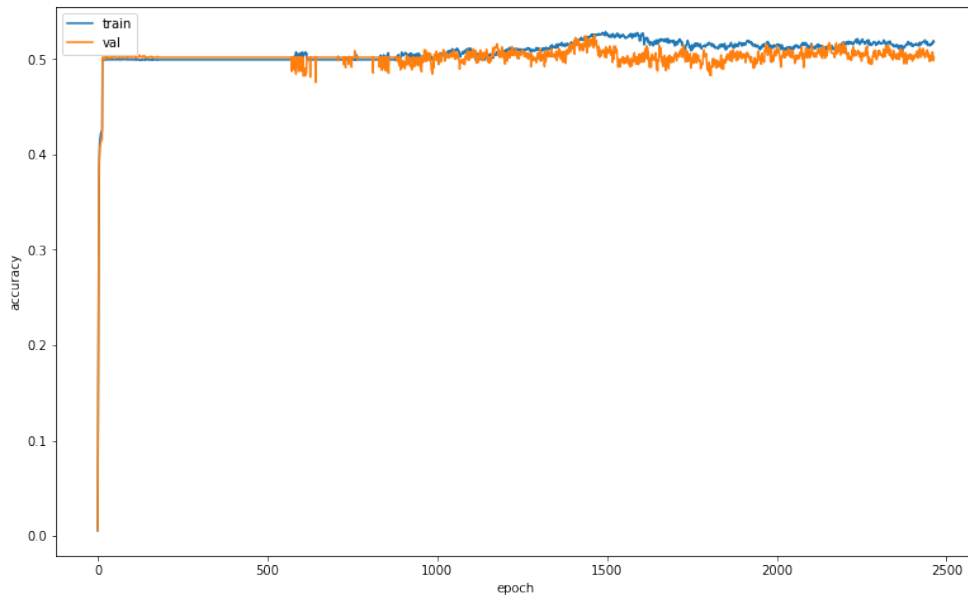
Bảng B.1: Kết quả cài đặt trên tập dữ liệu Poker Hand so với kết quả trong bài báo [1]

	Test Accuracy (%)
TabNet	96.7
TabNet (cài đặt của tác giả)	99.2

B.2 Ảnh hưởng của các siêu tham số

a. Ảnh hưởng của siêu tham số n_d

Hình B.1 và hình B.2 cho thấy kết quả độ chính xác trên tập train và tập validation qua các “epoch” khi n_d rất nhỏ và n_d rất lớn. Qua đó, có thể thấy được trường hợp n_d rất nhỏ mô hình của chúng em đã bị “chưa khớp dữ liệu” (underfitting), còn trường hợp n_d rất lớn mô hình của chúng em đã bị “quá khớp dữ liệu” (overfitting).



Hình B.1: Độ chính xác trên tập train và tập validation khi n_d rất nhỏ (màu xanh: train, màu cam: val)

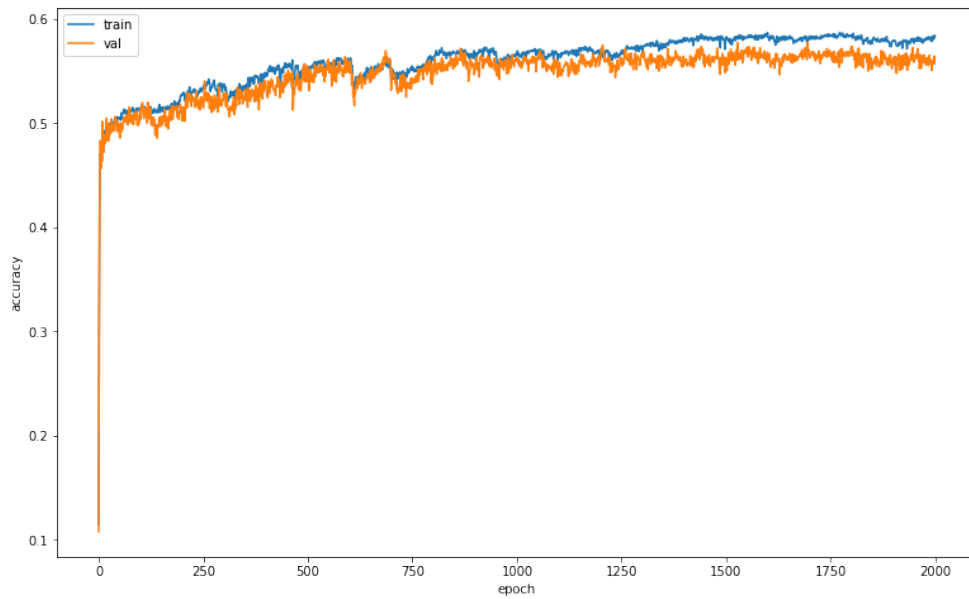


Hình B.2: Độ chính xác trên tập train và tập validation khi n_d rất lớn (màu xanh: train, màu cam: val)

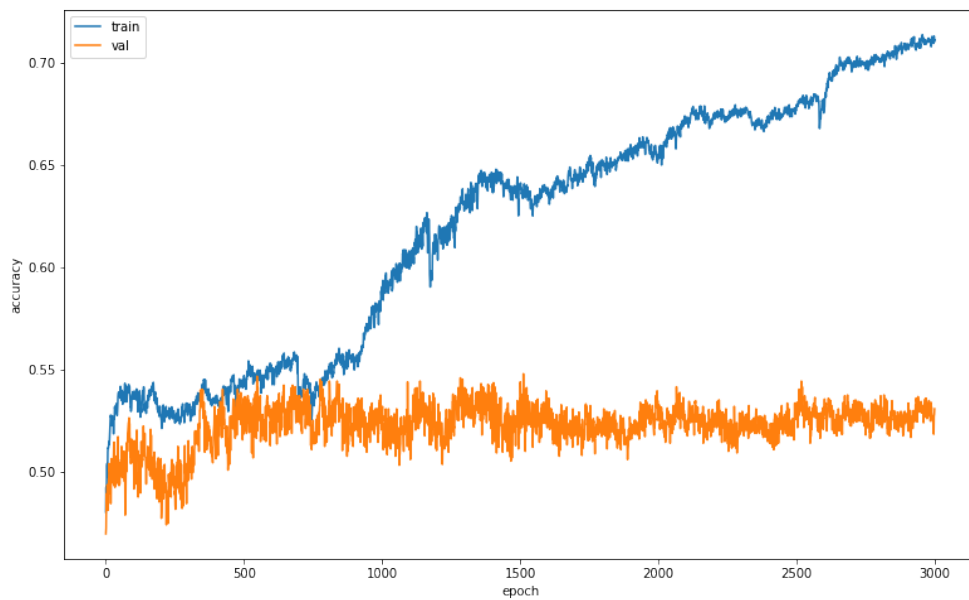
b. Ảnh hưởng của siêu tham số n_a

Qua hình B.3 và hình B.4, cho thấy ảnh hưởng của n_a tương tự như

n_d . Khi n_a rất nhỏ mô hình bị “chưa khớp dữ liệu” và khi n_a rất lớn mô hình bị “quá khớp dữ liệu”.



Hình B.3: Độ chính xác trên tập train và tập validation khi n_a rất nhỏ (màu xanh: train, màu cam: val)



Hình B.4: Độ chính xác trên tập train và tập validation khi n_a rất lớn (màu xanh: train, màu cam: val)