TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**ĐỒ ÁN CUỐI KÌ MÔN**

**MACHINE LEARNING**

**DỰ ĐOÁN GIÁ CỔ PHIẾU**

*Người hướng dẫn*: **TS LÊ ANH CƯỜNG**

*Người thực hiện*: **NGUYỄN NGỌC THIỆN - 51900711**

**TĂNG KIẾN TRUNG - 51900718**

**TRẦN BẢO KHA - 51900751**

Lớp **: 19050201**

Khoá  **: 23**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2022**

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**

Logo

Description automatically generated

**ĐỒ ÁN CUỐI KÌ MÔN**

**MACHINE LEARNING**

**DỰ ĐOÁN GIÁ CỔ PHIẾU**

*Người hướng dẫn*: **TS LÊ ANH CƯỜNG**

*Người thực hiện*: **NGUYỄN NGỌC THIỆN - 51900711**

**TĂNG KIẾN TRUNG - 51900718**

**TRẦN BẢO KHA - 51900751**

Lớp **: 19050201**

Khoá  **: 23**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2022**

LỜI CẢM ƠN

Lời nói đầu tiên chúng em xin gửi lời cảm ơn chân thành đến Thầy Lê Anh Cường, Giảng viên phụ trách môn học “Học máy” trong học kỳ này. Cảm ơn thầy đã đồng hành cùng chúng em trong suốt quá trình học, nhờ sự giảng dạy và dẫn dắt của thầy mà chúng em mới có đủ kiến thức và khả năng để hoàn thành bài báo cáo cuối kỳ này.

Tiếp đến chúng em muốn gửi lời cảm ơn đến Trường ĐH Tôn Đức Thắng nói chung và gửi lời cảm ơn trực tiếp đến khoa Công Nghệ Thông Tin nói riêng vì đã tạo điều kiện và cơ hội cho chúng em được học tập và phát triển bản thân. Không chỉ về mặt kiến thức mà còn về thể chất, tinh thần. Xin cảm ơn trường và khoa đã luôn làm việc tận tụy hết lòng giúp đỡ cho không chỉ chúng em mà còn toàn bộ sinh viên của trường có được điều kiện và môi trường học tập tốt nhất để có đủ khả năng và kiến thức ứng dụng thực tiễn.

Sau cùng thì chúng em hy vọng sẽ nhận được những sự góp ý và đóng góp chân thành từ thầy, do bản thân còn nhiều hạn chế và giới hạn về kiến thức, trình độ cho nên không thể tránh khỏi những sai sót. Chúng em sẽ tiếp thu những đóng góp, nhận xét từ thầy và tiếp tục hoàn thiện bản thân, vận dụng những nhận xét đó vào trong những môn học tiếp theo. Chúng em xin chúc thầy người luôn tận tụy dành thời gian cho sinh viên của mình có những ngày làm việc thật là hiệu quả và tràn đầy sức khỏe.

Chúng em xin gửi lời cảm ơn sâu sắc đến thầy Bùi Thanh Hùng!

**ĐỒ ÁN ĐƯỢC HOÀN THÀNH**

**TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

Chúng tôi xin cam đoan đây là sản phẩm đồ án của riêng chúng tôi và được sự hướng dẫn của TS. Lê Anh Cường. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong đồ án còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

**Nếu phát hiện có bất kỳ sự gian lận nào tôi xin hoàn toàn chịu trách nhiệm về nội dung đồ án của mình.** Trường đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do tôi gây ra trong quá trình thực hiện (nếu có).

*TP. Hồ Chí Minh, ngày tháng năm 2022*

*Tác giả*

*(ký tên và ghi rõ họ tên)*

*Nguyễn Ngọc Thiện*

*Tăng Kiến Trung*

*Trần Bảo Kha*

TÓM TẮT

MỤC LỤC

[LỜI CẢM ƠN iii](#_Toc120476283)

[TÓM TẮT v](#_Toc120476284)

[MỤC LỤC 1](#_Toc120476285)

[DANH MỤC KÍ HIỆU VÀ CHỮ VIẾT TẮT 2](#_Toc120476286)

[DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ 3](#_Toc120476287)

[CHƯƠNG 1 – Dataset 6](#_Toc120476288)

[Chương 2: Tiền xử lý dữ liệu 8](#_Toc120476289)

[Chương 3: Mô hình học máy 10](#_Toc120476290)

[3.1 LTSM 10](#_Toc120476291)

[TÀI LIỆU THAM KHẢO 15](#_Toc120476292)

DANH MỤC KÍ HIỆU VÀ CHỮ VIẾT TẮT

**CÁC KÝ HIỆU**

**CÁC CHỮ VIẾT TẮT**

DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ

**DANH MỤC HÌNH**

**No table of contents entries found.**

CHƯƠNG 1 – CƠ SỞ LÝ THUYẾT

1.1 Dataset

Giá cả chứng khoán luôn liên tục biến động, không ngừng thay đổi do chịu ảnh hưởng của nhiều yếu tố cả vi mô lần vĩ mô, như chính trị, chiến tranh, kinh tế, tình hình tài chính công ty,,, Đồng nghĩa với việc có rất nhiều dữ liệu để dự đoán được giá chứng khoán. Trong đề tài này chúng em sẽ sử dụng kiến thức học máy đã học để dự đoán giá cổ phiếu.

Bộ dữ liệu bao gồm 30 mã cổ phiếu chứa thông tin về giá cả (open, high, low, close) và khối lượng giao dịch chứng khoán từ ngày 04/01/2021 đến ngày 29/10/2021 được lấy từ package [vnquant](https://github.com/phamdinhkhanh/vnquant.git).

Bài toán gồm các dataset:

* price\_train.csv: Giá cả và khối lượng giao dịch chứng khoán, dùng để training mô hình
* price\_test.csv: Tập dữ liệu test dùng để đánh giá kết quả mô hình trên Kaggle.
* business\_train.csv: Báo cáo kinh doanh.
* finance\_train.csv: Báo cáo tài chính.

Trong đó mỗi dataset có các thuộc tính sau:

**price\_train**:

* date: thời gian giao dịch
* open: giá mở cửa (giá thực hiện tại lần khớp lệnh đầu tiên trong ngày giao dịch chứng khoán)
* high: giá cao nhất trong ngày
* low: giá thấp nhất trong ngày
* close: giá đóng cửa (giá thị trường của các cổ phiếu vào thời điểm đóng cửa một phiên giao dịch) (target\_variable)
* volume: khối lượng cổ phiếu giao dịch trong ngày (target variable)
* symbol: mã cổ phiếu

**business\_train**:

* index: chỉ số kết quả hoạt động kinh doanh, các cột còn lại dạng YYYY-MM: tháng báo cáo tài chính, thường là 3, 6, 9, 12 hàng năm.

**finance\_train**:

* index: chỉ số tài chính
* các cột còn lại dạng YYYY-MM: tháng báo cáo tài chính, thường là 3, 6, 9, 12 hàng năm.

**price\_test**: Đây là dữ liệu chứa các ngày và mã chứng khoán tương ứng mà chúng ta cần phải dự báo giá close.

1.2 Accuracy:

1.2.1 Khái niệm

Accuracy nghĩa tiếng việt là ‘Sự chính xác’. Ví dụ chúng ta đưa vào model 50 cái ảnh gồm cả chó và mèo vào một model phân loại và nhận đây là ảnh chó hay ảnh mèo. Sau khi chạy xong model thì thấy rằng có 30 ảnh được nhận diện đúng còn lại 20 ảnh nhận sai.

1.2.2 Cách tính:

Vậy ta có Accuaracy:

A picture containing diagram

Description automatically generated

Hình 11. Cách thính Accuracy

Tổng quát công thức trên ta có:

Text, letter

Description automatically generated

Hình 12. Công thức tổng quát cho Accuracy

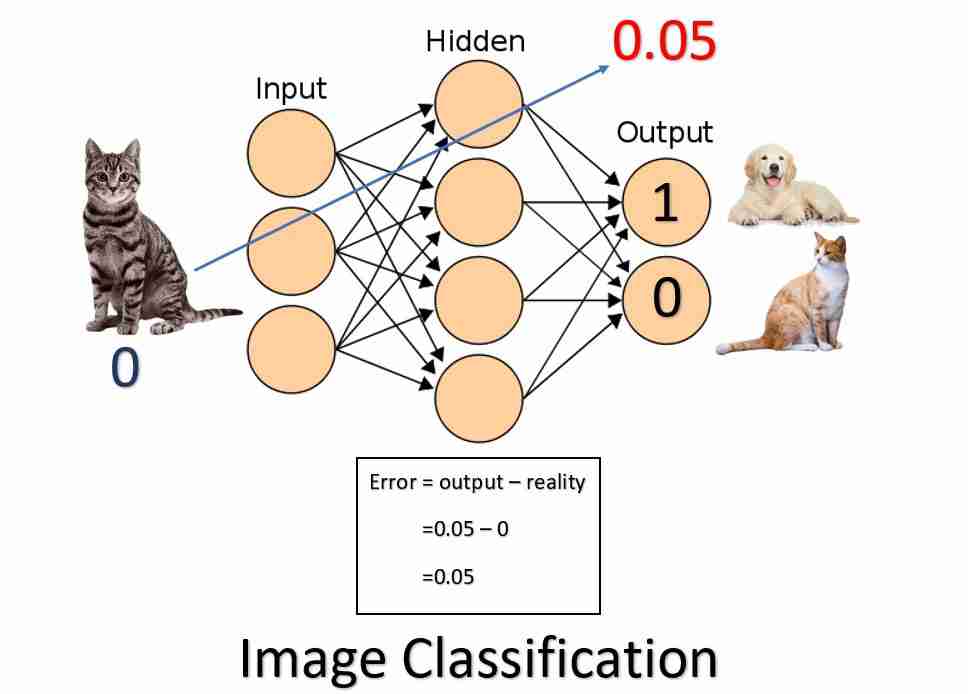
Trong đó:

* n: Số sample dự đoán đúng
* N: Tổng số sample đưa vào dự đoán

1.3 Loss:

1.3.1 Khái niệm:

Loss nghĩa Tiếng Việt là sự mất mát. Loss thường là một số thực không âm (trừ một số trường hợp loss là cosin proximity ) thể hiện sự chênh lệch giữa hai đại lượng: nhãn thật của dữ liệu và nhãn của dữ liệu do model predic ra.



Hình 13. Ví dụ về Loss

Hay nói một cách đơn giản thì Loss là khoảng cách giữa vector nhãn thực và vector nhãn model predict ra, model dự đoán càng lệch so với giá trị thực thì Loss (độ sai) càng to và ngược lại, nếu dự đoán càng sát với giá trị thực thì Loss (độ sai) sẽ nhỏ dần về 0.

1.3.2 Loss vs Accuracy:

Hai phương án có thể có cùng Accuracy nhưng Loss sẽ có thể khác nhau. Chúng ta rất hay nhầm lẫn và đánh đồng hai khái niệm này.

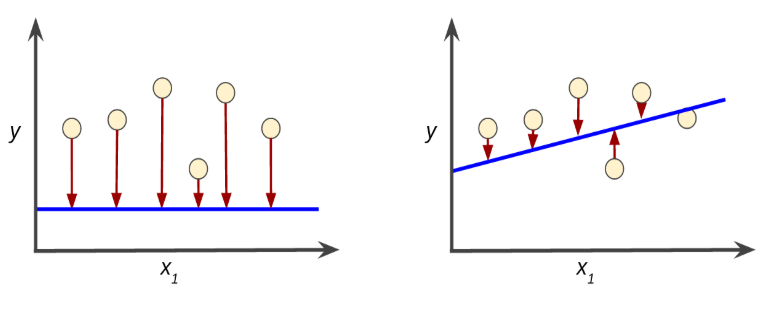
Ví dụ như với bài toán chó mèo, giả sử dữ liệu predict chỉ có duy nhất 1 ảnh CHÓ (vector nhãn thực là [1 , 0], probability là chó là 1 và mèo là 0, tạm gọi là y) và:

Phương án 01: Dự đoán ra probability của một ảnh đầu vào như sau 0.8 là chó và 0.2 là mèo (vector predict y = [0.8, 0.2]) => Ta quyết định ảnh này là CHÓ và dự đoán đúng => Accuracy = 1.

Phương án 02: Vector predict y = [0.6, 0.4] => Ta cũng vẫn quyết định đây là CHÓ vì probability là chó vẫn cao hơn và => Accuracy = 1

Tuy nhiên nhìn bằng mắt thường ta cũng thấy rõ ràng Phương án 1 tốt hơn vì dự đoán sát hơn với vector nhãn thực vì xác suất của dự đoán ảnh chó là 0.8 tốt hơn so với 0.6 là của Phương án 2. Trong thực tế chúng ta sẽ sử dụng một số hàm loss khác nhau tùy vào bài toán cụ thể như: Hinge Loss, Cross Entropy Loss,…nhưng tóm lại nôm na là Loss là hàm số trả về một số thực không âm để chỉ ra model của ta đang dự đoán sát (loss nhỏ) hay xa (loss to) so với nhãn thực của dữ liệu.

Chúng ta có thể nhìn 02 graph dưới đây để thấy rõ. Hình bên trái loss khá lớn vì khoảng cách giữa đường dự đoán (xanh blue) và các điểm dữ liệu là khá xa. Hình bên phải thì các điểm dữ liệu nằm gần hơn so với đường dự đoán nhiều => Loss sẽ nhỏ.



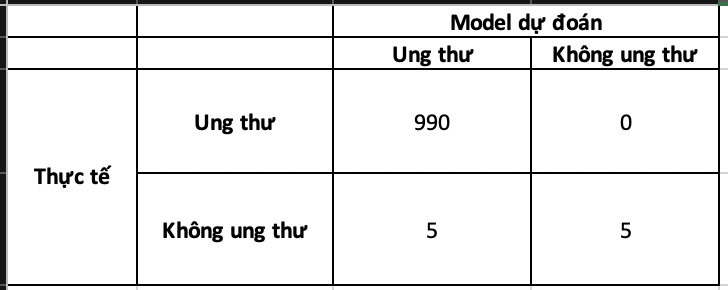
Hình 13. So sánh loss giữa 2 model

2.3 Confusion matrix:

2.3.1 Khái niệm:

Confusion Matrix là tổng kết giữa nhãn thực tế của dữ liệu với nhãn do model dự đoán ra. Ví dụ với bài toán chuẩn đoán ung thư (output trả về có/không ung thư).

Sau khi xây dựng xong model và chạy predict với toàn bộ dữ liệu thì ta dựa trên kết quả để xây dựng bảng như sau:



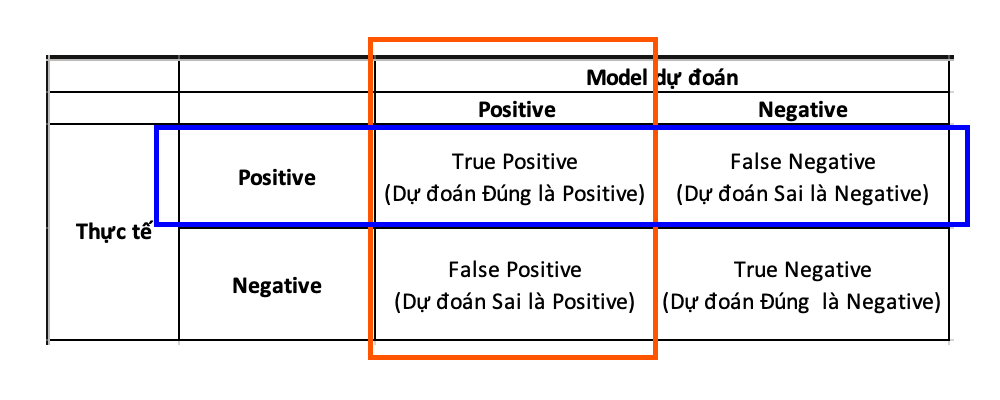
Hình 15. Ví dụ về dự đoán ung thư

Với bảng trên, ta có thể đưa ra những phân tích như sau khi quan sát:

Đầu tiên nhìn dòng số 1: Như vậy thực tế trong dữ liệu có 990 người ung thư và đều được model nhận đúng là Ung thư, không có ai bị dự đoán nhầm thành Không ung thư quá. Khá tốt.

Bây giờ nhìn dòng số 2, tổng dữ liệu của chúng ta có 5+5 = 10 người Không bị ung thư. Trong đó 5 người đã bị nhận nhầm thành Ung thư và 5 người đã được nhận đúng là Không ung thư.

Trong bài toán trên, output của bài toán có 2 lớp: Ung thư và Không ung thư. Bây giờ ta chọn 1 lớp nguy hiểm hơn (đó là Ung thư) làm lớp Positive và lớp còn lại là Negative và tiến hành thay nhãn:



Hình 16. Bảng phân bổ dự đoán

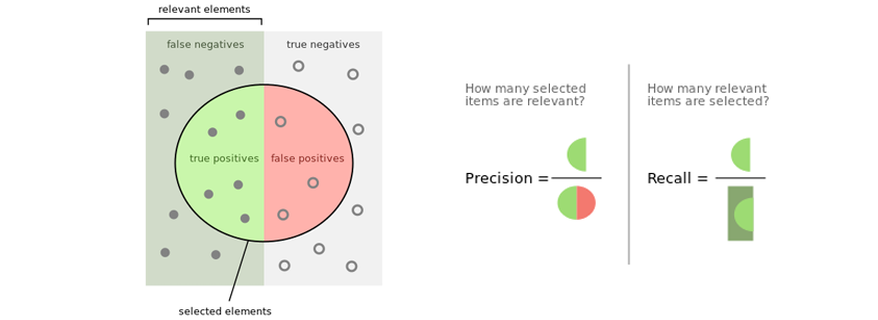
Từ đó ta có các khái niệm: True Positive, False Negative, False Positive và True Negative. Với bài toán trên thì:

* TP = 990
* FN = 0
* FP =TN = 5

Đột quỵ xảy ra khi một mạch máu vận chuyển oxy và chất dinh dưỡng đến não bị tắc nghẽn bởi cục máu đông hoặc khi mạch máu bị vỡ. Khi điều đó xảy ra, một phần của não không thể nhận được máu và oxy cần thiết nên các tế bào não sẽ chết.

Thuật toán học máy dưới đây sẽ dự đoán khả năng bị đột quỵ của một vài cá nhân trong tập dữ liệu bên dưới thông qua một số dữ liệu đã được thu thập.

2.4 Precision vs Recall:



Hình 17. Mối tương quan giữa Precision vs Recall

2.4.1 Precision:

Công thức tính Precision:

Text

Description automatically generated

Hình 18. Công thức tính Precision

Như vậy, có thể thấy Precision là tỷ lệ giữa số sample được tính là True Positive (TP) với tổng số sample được phân loại là Positive (bằng chính TP + FP). Chú ý hình chủ nhật màu đỏ bên trên hình trên.

Và khi đó 0 < Precision <= 1, Precision càng lớn có nghĩa là độ chính xác của các điểm tìm được càng cao. Ví dụ như bài trên trong 995 điểm được nhận là Positive thì có 990 điểm là TP và 5 điểm là FP nên ta có Precision khá cao:

A picture containing diagram

Description automatically generated

Hình 19. Ví dụ tính Precision

Precision sẽ cần được coi trọng hơn khi lựa chọn xây dựng model để giải quyết các bài toán cụ thể khi mà việc dự đoán nhầm False Positive mang lại hậu quả tồi tệ. Ví dụ với bài toán chặn Spam Mail chẳng hạn, khi đó việc nhận nhầm FP (nhầm 1 mail thường thành mail spam) sẽ làm ảnh hưởng đến công việc của người dùng vì miss một cái mail quan trọng (hợp đồng hàng nghìn tỷ đồng chẳng hạn).

2.4.2 Recall:

Công thức của Recall:

Text

Description automatically generated with low confidence

Hình 20. Công thức tính Recall

Công thức trên có nghĩa là tỷ lệ giữa các điểm positive thực được nhận đúng trên tổng điểm positive thực. Như vậy, Recall cao có nghĩa tỉ lệ bỏ sót các sample positive thực thấp. Chú ý hình chữ nhật màu xanh bên trên hình trên.

Do đó, Recall nên được gán trọng số cao hơn khi cân nhắc lựa chọn model tốt nhất khi mà việc nhận nhầm các nhãn Positive thực thành False Negative mang lại hậu quả khôn lường. Ví dụ như bài toán ung thư bên trên kìa, việc nhận nhầm người Ung thư thành người bình thường và trả về nhà không điều trị sớm thì sẽ rất nguy hiểm với người bệnh.

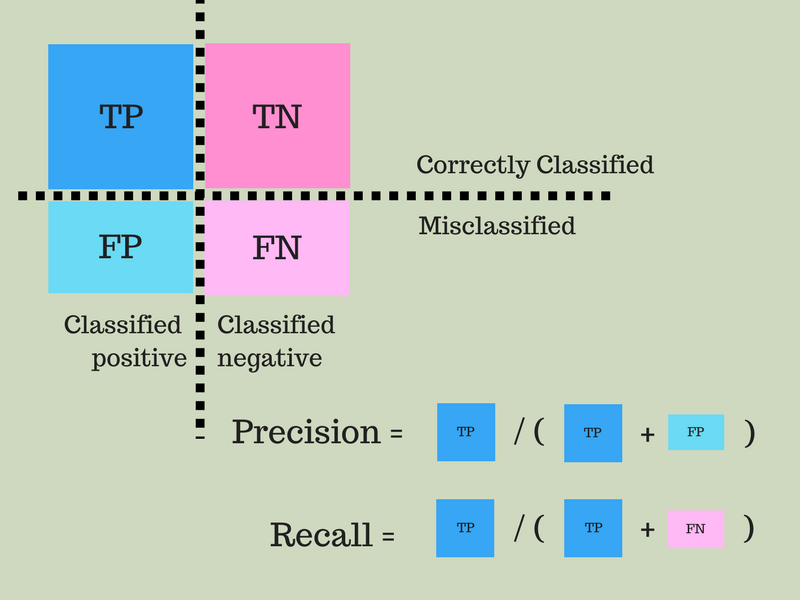
Cụ thể khi áp dụng công thức thì với bài ung thư bên trên ta có

A picture containing shape

Description automatically generated

Hình 21. Ví dụ tính Recall

Model của chúng ta không bỏ sót một sample positive thực nào!



Hình 22. Tổng quát giữa Precision vs Recall

2.5 F-1 Score

2.5.1 Khái niệm:

Như vậy chúng ta đã có 2 khái niệm Precision và Recall và mong muốn 2 giá trị này càng cao càng tốt. Tuy nhiên trong thực tế nếu ta điều chỉnh model để tăng Recall quá mức có thể dẫn đến Precision giảm và ngược lại, cố điều chỉnh model để tăng Precision có thể làm giảm Recall. Nhiệm vụ của chúng ta là phải cân bằng 2 đại lượng này.

Vậy thì bài toán mới được đặt ra là: giả sử chúng ta đang xây dựng một tập các model (để sau đó chọn ra model tốt nhất). Và chúng ta như “đứng giữa 2 con đường” khi lựa chọn các model sao cho cân bằng giữa Precision và Recall. Nhưng thật may có thêm một tham số nữa dung hòa giữa 2 cái và ta có thể căn vào đó để lựa chọn, đó là F-1 Score

2.5.2 Công thức:

Text

Description automatically generated

Hình 22. Công thức tính F1-score

Từ giờ chúng ta cứ dựa vào F1 để chọn model, F1 càng cao thì càng tốt và trường hợp lý tưởng nhất thì F1 = 1 (khi Recall = Precision=1).

Quay lại ví dụ bài toán ung thư, ta lại có:

A picture containing text, clock

Description automatically generated

Hình 23. Ví dụ tính F1-score

2.6 ROC CURVE

Đầu tiên ta sẽ tìm hiểu về True Positive Rate (TPR) chính là Recall và False Positive Rate (FPR) là tỷ lệ cảnh báo sai.

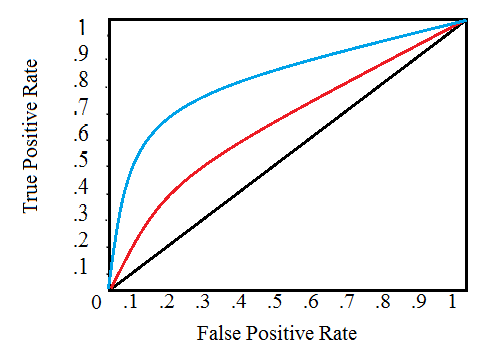
Text, letter

Description automatically generated

Hình 24. Công thức tính TPR vs FPR

Thực chất cái đại lượng ROC được dùng để chúng ta hiện lên mối quan hệ giữa TPR và FPR khi chúng ta thay đổi ngưỡng threshold của model.

Ví dụ ta có thể đặt một ngưỡng threshold=0.5 khi phân loại mail rác chẳng hạn, các mail nào có probability trên threshold sẽ được coi là mail rác. Bây giờ chúng ta sẽ thay đổi threshold để xem sự thay đổi của TPR và FPR trên đồ thị.



Hình 25. Sơ đồ biểu thị sự thay đổi TPR và FPR

Sau khi tính toán các đại lượng, chúng ta sẽ vẽ độ thị biểu diễn chúng ứng với sự thay đổi threshold như hình trên.

Khi threshold = 1.0, chúng ta sẽ ở góc dưới trái bởi vì model của chúng ta không predict bất kì điểm nào là positive và dẫn tới true positives = false positives = 0 (hay TPR = FPR = 0). Khi ta giảm dần threshold, số sample được predict là positivte tăng lên và dẫn tới true positive và false positive cũng tăng. Khi threshold về 0 thì TPR = FPR = 1 và chúng ta lên đỉnh trên cùng bên phải.

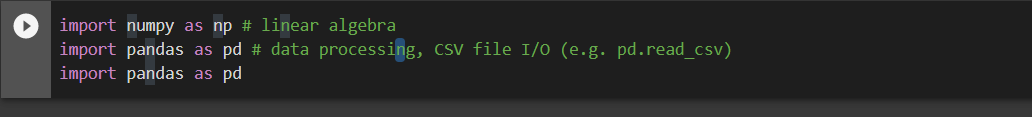
Việc vẽ đồ thị ROC sẽ dẫn tới một khái niệm tính toán vùng nằm dưới đường ROC (Area Under the Curve – AUC ). Đại lượng này có giá trị từ 0-1, giá trị càng cao thì model càng tốt và có nghĩa là đường ROC càng cong sát phía trên. Ví dụ như hình trên, đường màu xanh có AUC lớn hơn thể hiện model tốt hơn.

2.7 Tổng kết:

* True positives: Các điểm Positive thực được nhận Đúng là Positive
* False positives: Các điểm Negative thực được nhận Sai là Positive
* True negatives: Các điểm Negative thực được nhận Đúng là Negative
* False negatives: Các điểm Positive thực được nhận Sai là Negative
* Recall: Thể hiện khả năng phát hiện tất cả các postivie, tỷ lệ này càng cao thì cho thấy khả năng bỏ sót các điểm Positive là thấp
* Precision: Thể hiện sự chuẩn xác của việc phát hiện các điểm Positive. Số này càng cao thì model nhận các điểm Positive càng chuẩn.
* F1 score: Là số dung hòa Recall và Precision giúp ta có căn cứ để lựa chọn model. F1 càng cao càng tốt.
* Đường ROC : Thể hiện sự tương quan giữa Precision và Recall khi thay đổi threshold.
* Area Under the ROC: Là vùng nằm dưới ROC, vùng này càng lớn thì model càng tốt.
* Như vậy có chúng ta sẽ đặt câu hỏi nếu có nhiều hơn 2 class thì tính các món trên như thế nào? Khi đó bạn có thể xây dựng bảng Confusion Matrix cho mỗi lớp bằng cách ta coi lớp đó là lớp Positive, còn tất cả các lớp còn lại ta gộp chung vào thành một lớp Negative là hoàn tất.

CHƯƠNG 2: TIỀN XỬ LÝ DỮ LIỆU

Import thư viện



Hình 2.1

Đọc dữ liệu

Text

Description automatically generated with medium confidence

Hình 2.2

Để thuận tiện xử lý dữ liệu, chúng ta cần sắp xếp những mã chứng khoán này về dạng bảng có các cột là các mã chứng khoán và dòng là chuỗi thời gian.

Text

Description automatically generated

Hình 2.3

Kiểm tra nan values

Graphical user interface, website

Description automatically generated

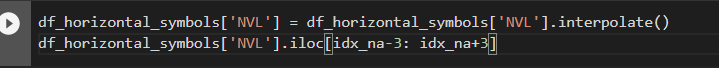
Hình 2.4

Do có cột NVL có một phần tử là NaN nên sử lý dữ liệu bằng phương pháp feed forward và sau đó sẽ fill dữ liệu bằng nội suy tuyến tính

Graphical user interface, text

Description automatically generated

Hình 2.5



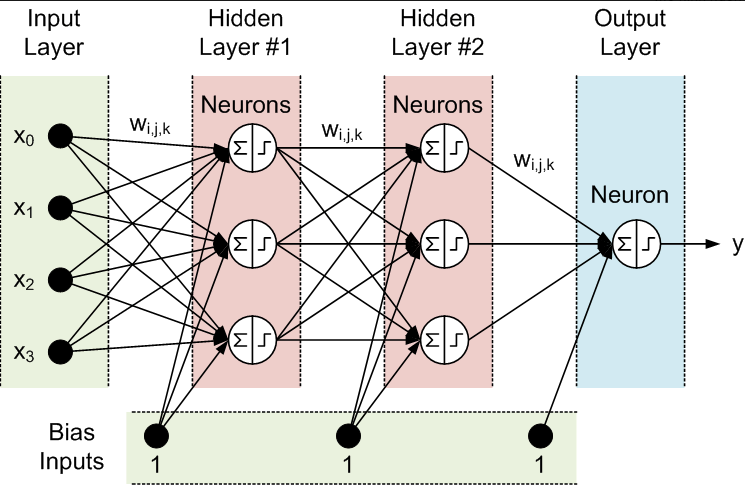
Hình 2.6

CHƯƠNG 3: TỔNG QUAN GIẢI THUẬT

3.1 Reccurent Neural Network (RNN)

3.1.1 Khái niệm về Reccurent Neural Network.

Reccurent Neural Network (RNN): còn được gọi là Mạng Hồi Quy. Dưới đây là mô hình của Neural Network.

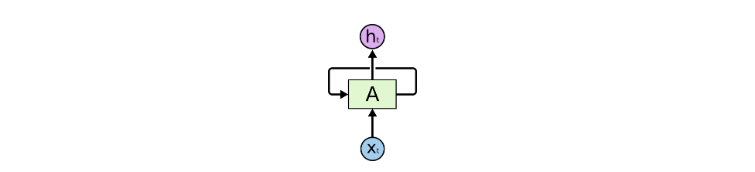


Hình : Mô hình mạng hồi quy

Neural Network bao gồm 3 phần chính là Input layer, Hidden layer và Output layer, ta có thể thấy là đầu vào và đầu ra của mạng neuron này là độc lập với nhau. Như vậy mô hình này không phù hợp với những bài toán dạng chuỗi như mô tả, hoàn thành câu, ... vì những dự đoán tiếp theo như từ tiếp theo phụ thuộc vào vị trí của nó trong câu và những từ đằng trước nó.

Một ví dụ đó là,bạn hãy tưởng tượng bạn muốn phân loại loại sự kiện nào đang xảy ra tại mọi thời điểm trong một bộ phim. Không rõ làm thế nào một mạng lưới thần kinh truyền thống có thể sử dụng suy luận của nó về các sự kiện trước đó trong phim để thông báo cho những sự kiện sau đó.

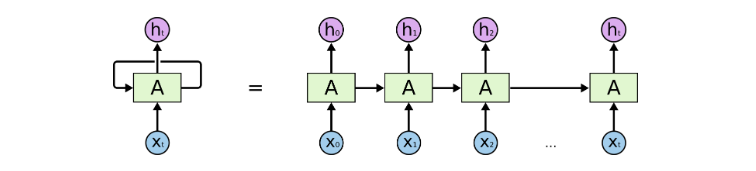
Recurrent neural networks giải quyết vấn đề này. Chúng là những mạng có vòng lặp bên trong, cho phép thông tin tồn tại lâu dài.



Hình : **Recurrent Neural Networks have loops.**

Trong sơ đồ trên, một đoạn mạng thần kinh, A, nhìn vào số đầu vào và xuất ra một giá trị . Một vòng lặp cho phép thông tin được truyền từ bước này sang bước tiếp theo của mạng.

Những vòng lặp này làm cho các mạng thần kinh tái phát có vẻ bí ẩn. Tuy nhiên, nếu bạn nghĩ kỹ hơn một chút, hóa ra chúng không khác mấy so với một mạng thần kinh bình thường. Một mạng thần kinh tái phát có thể được coi là nhiều bản sao của cùng một mạng, mỗi bản truyền một thông điệp cho người kế nhiệm. Xem xét điều gì sẽ xảy ra nếu chúng ta hủy kiểm soát vòng lặp:

****

Hình : một mạng lưới **Recurrent Neural Network không được kiểm soát**

Bản chất giống như chuỗi này tiết lộ rằng các mạng thần kinh tái phát có liên quan mật thiết đến các chuỗi và danh sách. Chúng là kiến ​​trúc tự nhiên của mạng lưới thần kinh để sử dụng cho những dữ liệu đó.

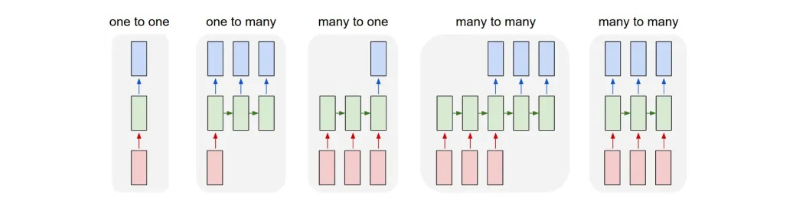
3.1.2 Dữ liệu dạng Sequence

Dữ liệu có thứ tự như các ảnh tách từ video ở trên được gọi là sequence, time-series data.

Trong bài toán dự đoán đột quỵ tim cho bệnh nhân bằng các dữ liệu tim mạch khám trước đó. Input là dữ liệu của những lần khám trước đó, ví dụ i1 là lần khám tháng 1, i2 là lần khám tháng 2,… i8 là lần khám tháng 8. (i1,i2,..i8) được gọi là sequence data. RNN sẽ học từ input và dự đoán xem bệnh nhân có bị đột quy tim hay không.

Ví dụ khác là trong bài toán dịch tự động với input là 1 câu, ví dụ “tôi yêu Việt Nam” thì vị trí các từ và sự xắp xếp cực kì quan trọng đến nghĩa của câu và dữ liệu input các từ [‘tôi’, ‘yêu’, ‘việt’, ‘nam’] được gọi là sequence data. **Trong bài toán xử lý ngôn ngữ (NLP) thì không thể xử lý cả câu được và người ta tách ra từng từ làm input, giống như trong video người ta tách ra các ảnh (frame) làm input.**

3.1.3 Phân loại bài toán RNN



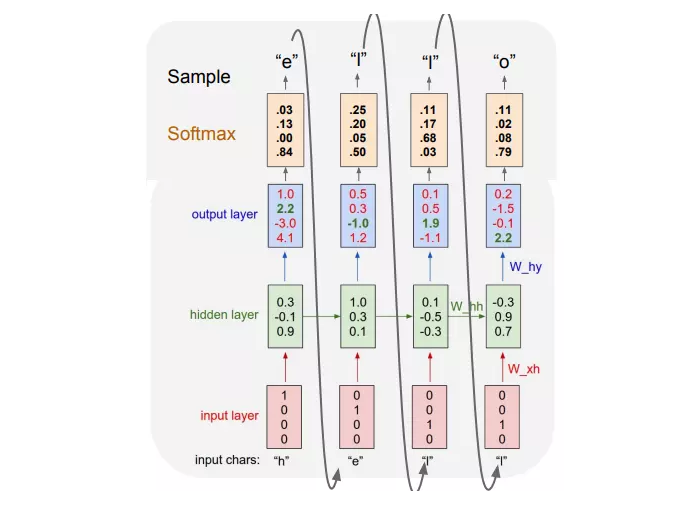
Hình : Các dạng bài toàn RNN

* **One to one**: mẫu bài toán cho Neural Network (NN) và Convolutional Neural Network (CNN), 1 input và 1 output, ví dụ với CNN input là ảnh và output là ảnh được segment.
* **One to many**: bài toán có 1 input nhưng nhiều output, ví dụ: bài toán caption cho ảnh, input là 1 ảnh nhưng output là nhiều chữ mô tả cho ảnh đấy, dưới dạng một câu.
* **Many to one**: bài toán có nhiều input nhưng chỉ có 1 output, ví dụ bài toán phân loại hành động trong video, input là nhiều ảnh (frame) tách ra từ video, ouptut là hành động trong video
* **Many to many**: bài toán có nhiều input và nhiều output, ví dụ bài toán dịch từ tiếng anh sang tiếng việt, input là 1 câu gồm nhiều chữ: “I love Vietnam” và output cũng là 1 câu gồm nhiều chữ “Tôi yêu Việt Nam”.

3.1.4 Ứng dụng bài toán RNN

Để hiểu hơn về mô hình RNN ta lấy một ví dụ sau: Cho tập input x = [h,e,l,o], sử dụng mô hình RNN để tạo ra một từ có nghĩa. Ta sẽ encode các chữ cái dưới dạng [one hot encoding](https://hackernoon.com/what-is-one-hot-encoding-why-and-when-do-you-have-to-use-it-e3c6186d008f).

Và kết quả như sau:



Hình : Ứng dụng mô hình RNN để tạo một từ có nghĩa

Ta thấy kí tự bắt đầu là "h" từ đó ta tìm ra chữ cái tiếp theo có xác suất lớn nhất là "e" và "e" tiếp tục trở thành input vào của cụm tiếp theo,... cứ như vậy cho đến khi tạo thành một từ có nghĩa, trong trường hợp này là từ "hello".

RNN được ứng dụng và thành công ở rất nhiều bài toán, đặc biệt là ở lĩnh vực NLP(xử lý ngôn ngữ tự nhiên). Trên lý thuyết thì đúng là RNN có khả năng nhớ được những tính toán (thông tin) ở trước nó, nhưng mô hình RNN truyền thống không thể nhớ được những bước ở xa do bị mất mát đạo hàm (sẽ được đề cập ở bài sau) nên những thành công của mô hình này chủ yếu đến từ một mô hình cải tiến khác là LSTM (Long Short-Term Memory, sẽ được đề cập ở những bài sau). LSTM về cơ bản cũng giống với RNN truyền thống ngoài việc thêm các cổng tính toán ở hidden layer để quyết định giữ lại các thông tin nào.

3.1.5 RNN và Backpropagation Through Time

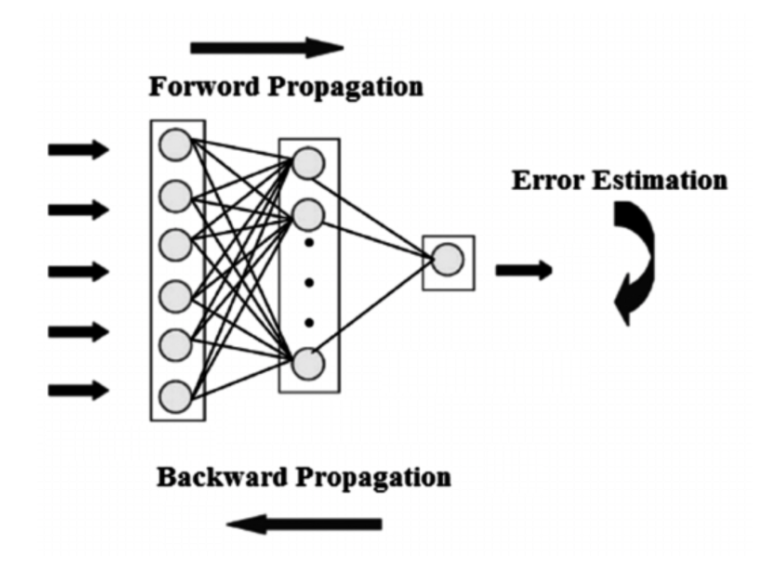
Backpropagation (viết tắt là BP hoặc backprop) được biết đến như một thuật toán đặc biệt trong học máy. Lan truyền ngược được sử dụng để tính toán độ dốc của hàm lỗi đối với trọng số của mạng thần kinh. Thuật toán hoạt động ngược lại thông qua các lớp độ dốc khác nhau để tìm ra đạo hàm riêng của các lỗi đối với các trọng số. Backprop sau đó sử dụng các trọng số này để giảm tỷ lệ lỗi khi đào tạo.

Trong các mạng thần kinh, về cơ bản, bạn thực hiện lan truyền chuyển tiếp để lấy đầu ra của mô hình của mình và kiểm tra xem đầu ra này đúng hay sai để nhận lỗi. Lan truyền ngược không là gì ngoài việc quay ngược qua mạng thần kinh của bạn để tìm đạo hàm riêng của sai số đối với các trọng số, điều này cho phép bạn trừ giá trị này khỏi các trọng số.

Các đạo hàm đó sau đó được sử dụng bằng phương pháp giảm dần độ dốc, một thuật toán có thể lặp đi lặp lại cực tiểu hóa một hàm đã cho. Sau đó, nó điều chỉnh trọng số lên hoặc xuống, tùy thuộc vào việc giảm sai số. Đó chính xác là cách một mạng lưới thần kinh học được trong quá trình đào tạo.

Vì vậy, với backpropagation, về cơ bản, bạn cố gắng điều chỉnh trọng số của mô hình trong khi đào tạo.

Hình ảnh bên dưới minh họa khái niệm lan truyền xuôi và lan truyền ngược trong forward neural network:

Hình : khái niệm lan truyền xuôi và lan truyền ngược trong mạng nơ ron

BPTT về cơ bản chỉ là một từ buzz ưa thích để thực hiện lan truyền ngược trên mạng thần kinh hồi quy không được kiểm soát. Unrolling là một công cụ trực quan và khái niệm, giúp bạn hiểu những gì đang diễn ra trong mạng. Hầu hết thời gian khi triển khai mạng thần kinh hồi quy trong các khung lập trình phổ biến, quá trình lan truyền ngược sẽ tự động được xử lý, nhưng bạn cần hiểu cách thức hoạt động của nó để khắc phục sự cố có thể phát sinh trong quá trình phát triển.

Bạn có thể xem RNN dưới dạng một chuỗi các mạng nơ-ron mà bạn đào tạo lần lượt bằng kỹ thuật lan truyền ngược.

3.1.6 Vấn đề của RNN

Có hai trở ngại lớn mà RNN phải giải quyết, nhưng để hiểu chúng, trước tiên bạn cần biết gradient là gì.

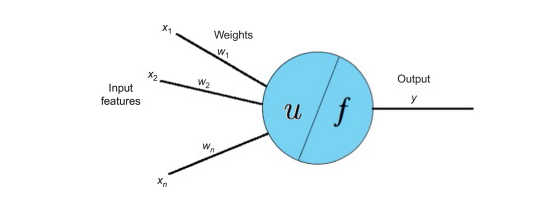
Một gradient là một đạo hàm riêng đối với các đầu vào của nó. Nếu bạn không biết điều đó có nghĩa là gì, chỉ cần nghĩ về nó như sau: độ dốc đo lường mức độ thay đổi đầu ra của hàm nếu bạn thay đổi đầu vào một chút.

Bạn cũng có thể coi gradient là hệ số góc của một hàm. Độ dốc càng cao, độ dốc càng lớn và mô hình có thể học càng nhanh. Nhưng nếu độ dốc bằng 0, mô hình sẽ ngừng học. Độ dốc chỉ đơn giản là đo lường sự thay đổi trong tất cả các trọng số liên quan đến thay đổi sai số.

3.2 Multi Layer Percepton ( MLP )

3.2.1 Khái niệm về Multi Layer Percepton.

Multi Layer Percepton (MLP): Còn được gọi là mạng tế bào thần kinh được tạo thành từ các tế bào thần kinh riêng lẻ được gọi là **perceptron.**



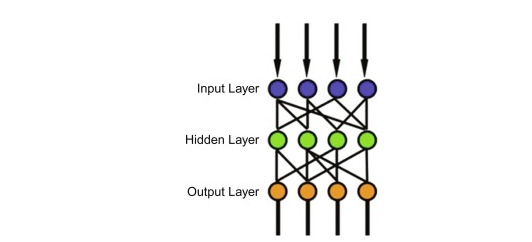
Hình : Sơ đồ của Perceptron với n input features

Nhìn vào hình trên, chúng ta có thể thấy rằng một tế bào thần kinh có thể nhận nhiều đầu vào và tạo ra một đầu ra duy nhất. mẫu perceptron cũng tương tự.

Một perceptron sẽ nhận một hoặc nhiều đầu **x**trong hệ nhị phân và tạo ra một kết quả nhị phân duy nhất **y**. đầu vào được tính theo thông số trọng lượng tương ứng của chúng **w**và đầu ra được quyết định dựa trên ngưỡng quyết định **u.**

Các tính năng đầu vào được chuyển đến một hàm đầu vào u , ta có thể tính tổng trọng số của các input đầu vào:

3.2.2 Kiến trúc mạng nơ-ron nhân tạo



Hình : Kiến trúc mạng nơ-ron nhân tạo

Mạng nơ-ron là sự kết hợp của các lớp perceptron, còn được gọi là perceptron nhiều lớp, như được hiển thị bên dưới

Một mạng nơ-ron sẽ có 3 loại lớp:

* **lớp đầu vào**: là lớp ngoài cùng bên trái của mạng thể hiện các đầu vào của mạng.
* **Lớp đầu ra**: là lớp ngoài cùng bên phải của mạng thể hiện các kết quả đầu ra của mạng.
* **lớp ẩn**: là lớp nằm giữa lớp đầu vào và lớp đầu ra thể hiện suy luận logic của mạng.

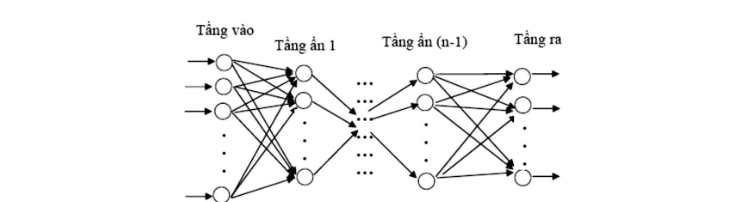
Có một lưu ý ở đây là nơ-ron chỉ có 1 lớp đầu vào và 1 lớp đầu ra, nhưng nó có thể có nhiều lớp ẩn.

Trong mạng nơ-ron, mỗi nút là một nơ-ron thần kinh, nhưng chức năng kích hoạt của nó có thể khác nhau. tuy nhiên trong thực tế người ta thường để nguyên dạng để tính toán cho tiện.

Trong mỗi lớp, số lượng nút thần kinh có thể khác nhau tùy thuộc vào vấn đề và giải pháp. nhưng thường khi làm việc, người ta để các lớp ẩn có số nơron bằng nhau. hơn nữa, các tế bào thần kinh phân lớp thường kết hợp với nhau để tạo thành một **mạng được kết nối đầy đủ**(mạng được kết nối đầy đủ).

3.2.3 Kiến trúc mạng Neural MLP ( Multi-layer Perceptron )

Mô hình mạng nơron được sử dụng rộng rãi nhất là mô hình mạng nhiều tầng truyền thẳng (MLP: Multi Layer Perceptron). Một mạng MLP tổng quát là mạng có n (n≥2) tầng (thông thường tầng đầu vào không được tính đến): trong đó gồm một tầng đầu ra (tầng thứ n) và (n-1) tầng ẩn.



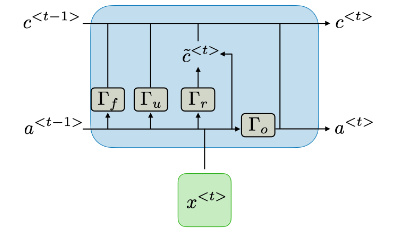
Hình : Kiến trúc của một mạng MPL tổng quát

* Đầu vào là các vector (x1, x2, …, xp) trong không gian p chiều, đầu ra là các vector (y1, y2, …, yq) trong không gian q chiều. Đối với các bài toán phân loại, p chính là kích thước của mẫu đầu vào, q chính là số lớp cần phân loại.
* Mỗi neural thuộc tầng sau liên kết với tất cả các nơron thuộc tầng liền trước nó.
* Đầu ra của neural tầng trước là đầu vào của nơron thuộc tầng liền sau nó.

Hoạt động của mạng MLP như sau: tại tầng đầu vào các neural nhận tín hiệu vào xử lý (tính tổng trọng số, gửi tới hàm truyền) rồi cho ra kết quả (là kết quả của hàm truyền); kết quả này sẽ được truyền tới các neural thuộc tầng ẩn thứ nhất; các nơron tại đây tiếp nhận như là tín hiệu đầu vào, xử lý và gửi kết quả đến tầng ẩn thứ 2. Quá trình tiếp tục cho đến khi các neural thuộc tầng ra cho kết quả.

Chương 4: Mô hình học máy

4.1 LTSM



Mạng trí nhớ ngắn hạn định hướng dài hạn còn được viết tắt là LSTM làm một kiến trúc đặc biệt của RNN. Bước đầu tiên trong LSTM sẽ quyết định xem thông tin nào chúng ta sẽ cho phép đi qua ô trạng thái (cell state). Nó được kiểm soát bởi hàm sigmoid trong một tầng gọi là tầng quên (forget gate layer).

Đầu tiên nó nhận đầu vào là 2 giá trị ht−1 và xt và trả về một giá trị nằm trong khoảng 0 và 1 cho mỗi giá trị của ô trạng thái Ct−1. Nếu giá trị bằng 1 thể hiện ‘giữ toàn bộ thông tin’ và bằng 0 thể hiện ‘bỏ qua toàn bộ chúng’.



Bước tiếp theo chúng ta sẽ quyết định loại thông tin nào sẽ được lưu trữ trong ô trạng thái. Bước này bao gồm 2 phần. Phần đầu tiên là một tầng ẩn của hàm sigmoid được gọi là tầng cổng vào (input gate layer) quyết định giá trị bao nhiêu sẽ được cập nhật. Tiếp theo, tầng ẩn hàm tanh sẽ tạo ra một véc tơ của một giá trị trạng thái mới C~t mà có thể được thêm vào trạng thái. Tiếp theo kết hợp kết quả của 2 tầng này để tạo thành một cập nhật cho trạng thái.



Đây là thời điểm để cập nhật một ô trạng thái cũ, Ct−1 sang một trạng thái mới Ct. Những bước trước đó đã quyết định làm cái gì, và tại bước này chỉ cần thực hiện nó.

Chúng ta nhân trạng thái cũ với ft tương ứng với việc quên những thứ quyết định được phép quên sớm. Phần tử đề cử it∗C~t là một giá trị mới được tính toán tương ứng với bao nhiêu được cập nhật vào mỗi giá trị trạng thái.



Cuối cùng cần quyết định xem đầu ra sẽ trả về bao nhiêu. Kết quả ở đầu ra sẽ dựa trên ô trạng thái, nhưng sẽ là một phiên bản được lọc. Đầu tiên, chúng ta chạy qua một tầng sigmoid nơi quyết định phần nào của ô trạng thái sẽ ở đầu ra. Sau đó, ô trạng thái được đưa qua hàm tanh (để chuyển giá trị về khoảng -1 và 1) và nhân nó với đầu ra của một cổng sigmoid, do đó chỉ trả ra phần mà chúng ta quyết định.

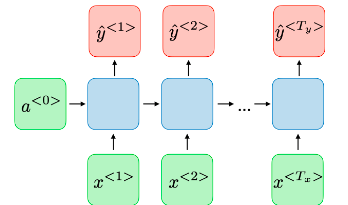
Xây dựng model

Text

Description automatically generated

Many-to-Many

Dựa trên mô hình LSTM chúng ta có thể xem xét bài toán là một tác vụ dự báo `many-to-many`. Trong đó mỗi một giá trị đầu vào là một véc tơ window của một mã chứng khoán thứ i. Các đầu ra là giá trị dự báo của phiên tiếp theo của chính mã chứng khoán đó.



Xây dựng model

Text

Description automatically generated

TÀI LIỆU THAM KHẢO

1. [Khoa học dữ liệu (phamdinhkhanh.github.io)](https://phamdinhkhanh.github.io/2019/04/22/Ly_thuyet_ve_mang_LSTM.html#3-m%E1%BA%A1ng-tr%C3%AD-nh%E1%BB%9B-ng%E1%BA%AFn-h%E1%BA%A1n-%C4%91%E1%BB%8Bnh-h%C6%B0%E1%BB%9Bng-d%C3%A0i-h%E1%BA%A1n-lstm---long-short-term-memory)

2. <https://github.com/phamdinhkhanh/vnquant?fbclid=IwAR08gL691NgRftf0Qi1Jxjt5lesCVN6xdlAgR4dyZOvtdWYMG090Uk3Q_9>s

3. <https://www.kaggle.com/competitions/stock-market-prediction/data?fbclid=IwAR08gL691NgRftf0Qi1Jxjt5lesCVN6xdlAgR4dyZOvtdWYMG090Uk3Q_9s>

4. <https://builtin.com/data-science/recurrent-neural-networks-and-lstm>

5. <https://www.geeksforgeeks.org/introduction-to-recurrent-neural-network/>

6. <https://viblo.asia/p/recurrent-neural-networkphan-1-tong-quan-va-ung-dung-jvElaB4m5kw>

7. <https://nttuan8.com/bai-13-recurrent-neural-network/#Recurrent_Neural_Network_la_gi>

8. <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>

9. <https://httl.com.vn/wiki/mlp-la-gi/>

10. <https://gocnhinso.com/mang-mlp-multi-layer-perceptron-la-gi-nen-tang-cua-deep-learning/>

11. <https://www.sciencedirect.com/topics/computer-science/multilayer-perceptron>

12. <https://tek4.vn/phat-trien-cac-mo-hinh-multilayer-perceptron-keras-co-ban>

13. <https://www.kaggle.com/code/phamdinhkhanh/multi-layer-perceptron-in-timeseries-forecast/notebook>

**PHỤ LỤC**