

**TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO CUỐI KỲ
MÔN NHẬP MÔN HỌC MÁY**

DỰ ÁN CUỐI KỲ

Người hướng dẫn: **TS LÊ ANH CƯỜNG**

Người thực hiện: **ĐÀO NGỌC TIỀN – 51603329**

HOÀNG PHÚC THIÊN AN – 51900644

Lớp : 190050401 – 16050304

Khoá : 20 & 23

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2023

**TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO CUỐI KỲ
MÔN NHẬP MÔN HỌC MÁY**

DỰ ÁN CUỐI KỲ

Người hướng dẫn: **TS LÊ ANH CƯỜNG**

Người thực hiện: **ĐÀO NGỌC TIỀN – 51603329**

HOÀNG PHÚC THIÊN AN – 51900644

Lớp : 190050401 – 16050304

Khoá : 20 & 23

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2023

LỜI CẢM ƠN

“Nhóm em xin chân thành cảm ơn giảng viên bộ môn – Thầy Lê Anh Cường đã giảng dạy chi tiết, nhiệt tình và có những ví dụ thực nghiệm bổ ích trong thời gian học cũng như vận dụng vào bài báo cáo này.

Do chưa có nhiều kinh nghiệm nên trong quá trình làm bài và viết báo cáo sẽ không tránh khỏi những thiếu sót. Nhóm em mong nhận được những nhận xét, đóng góp ý kiến từ thầy để bài báo cáo được hoàn thiện nhất.

Lời cuối cùng, nhóm em xin chúc thầy nhiều sức khỏe, thành công và hạnh phúc.”

BÁO CÁO ĐƯỢC HOÀN THÀNH TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG

Nhóm xin cam đoan đây là sản phẩm đồ án của riêng nhóm và được sự hướng dẫn của Thầy 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 30 tháng 04 năm 2023

Tác giả

(ký tên và ghi rõ họ tên)

Đào Ngọc Tiên

Hoàng Phúc Thiên An

PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN

Phần xác nhận của GV hướng dẫn

Tp. Hồ Chí Minh, ngày tháng năm
(kí và ghi họ tên)

Phần đánh giá của GV chấm bài

Tp. Hồ Chí Minh, ngày tháng năm
(kí và ghi họ tên)

TÓM TẮT

Nội dung bài báo cáo tìm hiểu qua một số thuật toán về hiện tượng Overfitting xảy ra khi huấn luyện các mô hình học máy và các phương pháp để tránh hiện tượng đó. Giải bài toán Recommendation. Tìm hiểu về mô hình học sâu Long Short Term Memory, áp dụng mô hình vào bài toán thực tế và so sánh với một số mô hình học máy truyền thống.

MỤC LỤC

LỜI CẢM ƠN	i
PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN	iii
TÓM TẮT	iv
MỤC LỤC	1
DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ	4
PHẦN 1 – HIỆN TƯỢNG OVERFITTING.....	6
1.1 Overfitting là gì?	6
1.2 Overfitting xảy ra khi nào?	6
1.3 Các phương pháp phòng tránh	8
1.3.1 Validation.....	8
1.3.1.1 Validation thông thường.....	8
1.3.1.2 Cross-validation	9
1.3.2 Regularization	10
1.3.2.1 Early Stopping	10
1.3.2.2 Network-reduction	12
1.3.2.3 Expansion of the training data (Mở rộng tập dữ liệu training).....	13
1.3.2.4 L1 regularization.....	14
1.3.2.5 L2 regularization (Weight Decay)	14
1.3.2.6 Dropout	15
1.4 Thực nghiệm và so sánh giữa các phương pháp	16
PHẦN 2 – XÂY DỰNG RECOMMENDATION SYSTEM	25
2.1 Bộ dữ liệu Amazon Fine Food Reviews	25
2.2 Mô hình Recommendation Systems	25
2.2.1 Phương pháp Collaborative Filtering.....	26
2.3 Thư viện <i>Surprise</i> trong Recommendation Systems	26

2.4 Recommendation Systems với Amazon Fine Food Reviews	27
2.4.1 Tiền xử lý dữ liệu	28
2.4.2 Xây dựng và đánh giá mô hình Singular Value Decomposition	29
2.4.3 Tạo recommendations cho một người dùng	30
PHẦN 3 – MÔ HÌNH HỌC SÂU LONG SHORT TERM MEMORY	33
3.1 Cơ sở lý thuyết	33
3.1.1 Ý tưởng cốt lõi	34
3.1.2 Phân tích mô hình LSTM.....	36
3.2 Một số biến thể.....	38
3.2.1 Biến thể nối cổng loại trừ và cổng đầu vào	38
3.2.2 Biến thể Gated Recurrent Unit.....	38
3.3 Thực nghiệm mô hình LSTM	39
3.3.1 Tiến hành tiền xử lý dữ liệu	41
3.3.2 Xây dựng mô hình LSTM.....	43
3.3.3 So sánh với LinearRegression - MLPRegressor - DecisionTreeRegressor	46

DANH MỤC KÍ HIỆU VÀ CHỮ VIẾT TẮT

CÁC KÝ HIỆU

CÁC CHỮ VIẾT TẮT

LSTM	Long Short Term Memory
MLP	Multilayer Perceptron
SVD	Singular Value Decomposition
RNN	Recurrent Neural Network
NaN	Not a Number
IQR	Interquartile Range
RMSE	Root Mean Squared Error
MAE	Mean Absolute Error

DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ

DANH MỤC HÌNH

Hình 1.1 Hình minh họa Overfitting	6
Hình 1.2 Ví dụ về underfitting, good fit	7
Hình 1.3 Ví dụ về overfitting	7
Hình 1.4 Cách lựa chọn mô hình dựa trên phương pháp validation	9
Hình 1.5 Minh họa Early stopping.....	11
Hình 1.6 Thực hiện chạy Early stopping	20
Hình 1.7 So sánh giá trị Accuaracy.....	23
Hình 1.8 So sánh giá trị Loss	24
Hình 2.1 Thông tin và mô tả tổng quan về bộ dữ liệu	28
Hình 2.2 Kết quả trước và sau khi xử lý missing values	28
Hình 2.3 Kết quả các chỉ số RMSE và MAE của mô hình	30
Hình 2.4 Kết quả 10 đề xuất cho người mới.....	31
Hình 2.5 Kết quả 10 đề xuất cho người đã có trong cơ sở dữ liệu	32
Hình 3.1 Minh họa mạng RNN	33
Hình 3.2 Cấu trúc mạng LSTM.....	34
Hình 3.3 Các kí hiệu sử dụng trong mạng LSTM.....	34
Hình 3.4 Minh họa trạng thái tế bào	35
Hình 3.5 Cổng trạng thái LSTM	35
Hình 3.6 Minh họa về input gate layer.....	36
Hình 3.7 Minh họa về cập nhật	37
Hình 3.8 Minh họa về quyết định đầu ra.....	37
Hình 3.9 Minh họa đầu ra	38
Hình 3.10 Biến thể nối 2 cổng loại trừ và đầu vào với nhau	38
Hình 3.11 Biến thể Gated Recurrent Unit.....	39
Hình 3.12 Năm hàng đầu của bộ dữ liệu.....	40

Hình 3.13 Năm hàng cuối của bộ dữ liệu.....	40
Hình 3.14 Thông tin và mô tả tổng quan về các giá trị thống kê của các cột dữ liệu....	41
Hình 3.15 Trước và sau khi xử lý cột bị thiếu dữ liệu	42
Hình 3.16 Kết quả điểm đánh giá mô hình trên tập test.....	46
Hình 3.17 Biểu đồ so sánh dữ liệu thực tế và dữ liệu dự đoán của mô hình	46
Hình 3.18 Kết quả đánh giá hiệu suất thông qua chỉ số RMSE và MAE	47
Hình 3.19 Biểu đồ đánh giá hiệu suất các mô hình.....	47

DANH MỤC BẢNG

PHẦN 1 – HIỆN TƯỢNG OVERFITTING

1.1 Overfitting là gì?

Overfitting là hiện tượng khi mô hình xây dựng thể hiện quá chi tiết bộ dữ liệu đào tạo. Điều này có nghĩa là cả dữ liệu nhiễu, hoặc dữ liệu bất thường trong tập đào tạo đều được chọn và học để đưa ra quy luật mô hình. Những quy luật này sẽ không có ý nghĩa nhiều khi áp dụng với bộ dữ liệu mới có thể có dạng dữ liệu nhiễu khác. Theo một cách dễ hiểu có nghĩa rằng chúng đạt kết quả tốt trong tập dữ liệu đào tạo nhưng lại kém trong tập dữ liệu kiểm tra (thực tế). Khi đó, nó ảnh hưởng tiêu cực tới độ chính xác của mô hình nói chung.

Hiện tượng Overfitting thường xảy ra trong các mô hình phi tham số hoặc phi tuyến do phương sai quá cao và khi mà phương sai quá thấp mà bias quá cao.



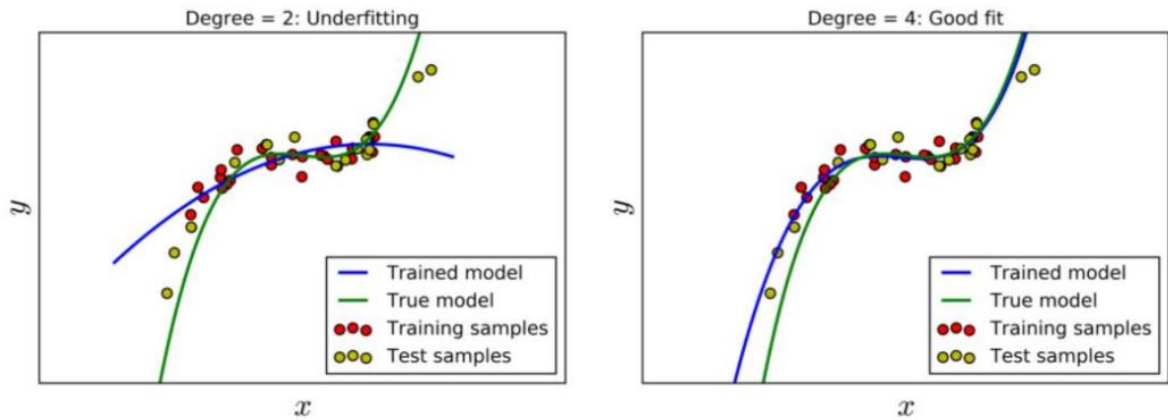
Hình 1.1 Hình minh họa Overfitting

1.2 Overfitting xảy ra khi nào?

Về cơ bản, overfitting xảy ra khi chưa có đủ dữ liệu để đánh giá hoặc do mô hình quá phức tạp để mô phỏng training data. Điều này đặc biệt xảy ra khi lượng dữ liệu

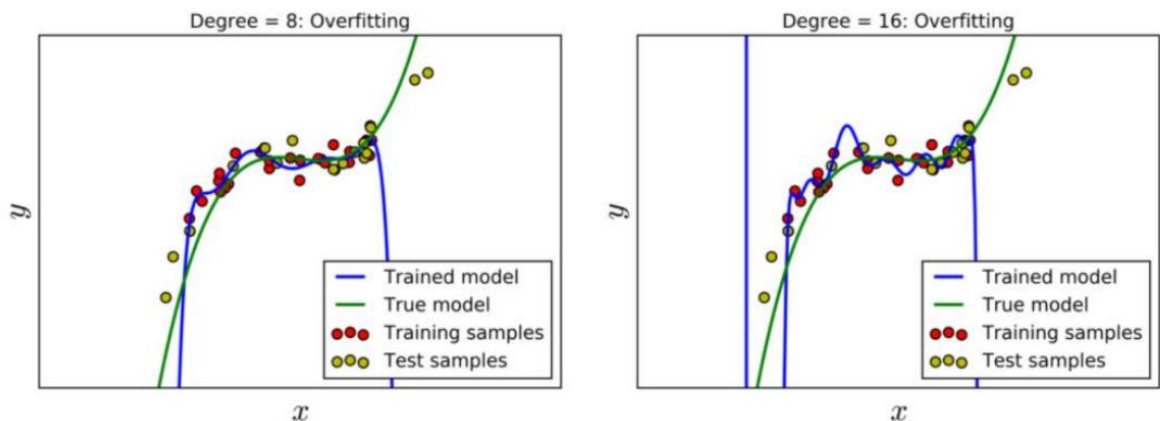
training quá nhỏ trong khi độ phức tạp của mô hình quá cao dẫn tới mất tính tổng quát của mô hình.

Để hiểu rõ hơn về overfitting, ta có các hình ảnh ví dụ như sau:



Hình 1.2 Ví dụ về underfitting, good fit

Trong hình 2, ta thấy trong trường hợp underfitting, mô hình huấn luyện (train model) quá đơn giản, không khớp so với dữ liệu huấn luyện. Còn bên good fit, mô hình có độ phức tạp vừa đủ, mang tính tổng quát, nó khớp với dữ liệu huấn luyện, dữ liệu test và gần giống với mô hình thực tế (true model).



Hình 1.3 Ví dụ về overfitting

Trong hình 3, ta thấy mô hình huấn luyện (train model) khá phức tạp, nó quá khớp với dữ liệu huấn luyện, nhưng lại không mang tính tổng quát và khác xa so với mô hình

thực tế (true model). Điều này dẫn đến việc mô hình sẽ dự đoán sai những dữ liệu mới không có trong dữ liệu huấn luyện. Ví dụ như những điểm test samples nằm trên cùng trong hình 3, mô hình huấn luyện hoàn toàn không thể chạm tới nó được.

1.3 Các phương pháp phòng tránh

1.3.1 *Validation*

1.3.1.1 Validation thông thường

Chúng ta thường quen với việc chia tập dữ liệu ra thành hai tập nhỏ: training data và test data. Tuy nhiên, khi xây dựng mô hình ta không được sử dụng test data mà chỉ được phép sử dụng training data để xây dựng nên một mô hình tổng quát để dự đoán một điểm y nào đó. Vậy làm cách nào để biết được chất lượng của mô hình với unseen data (tức dữ liệu chưa nhìn thấy bao giờ)?

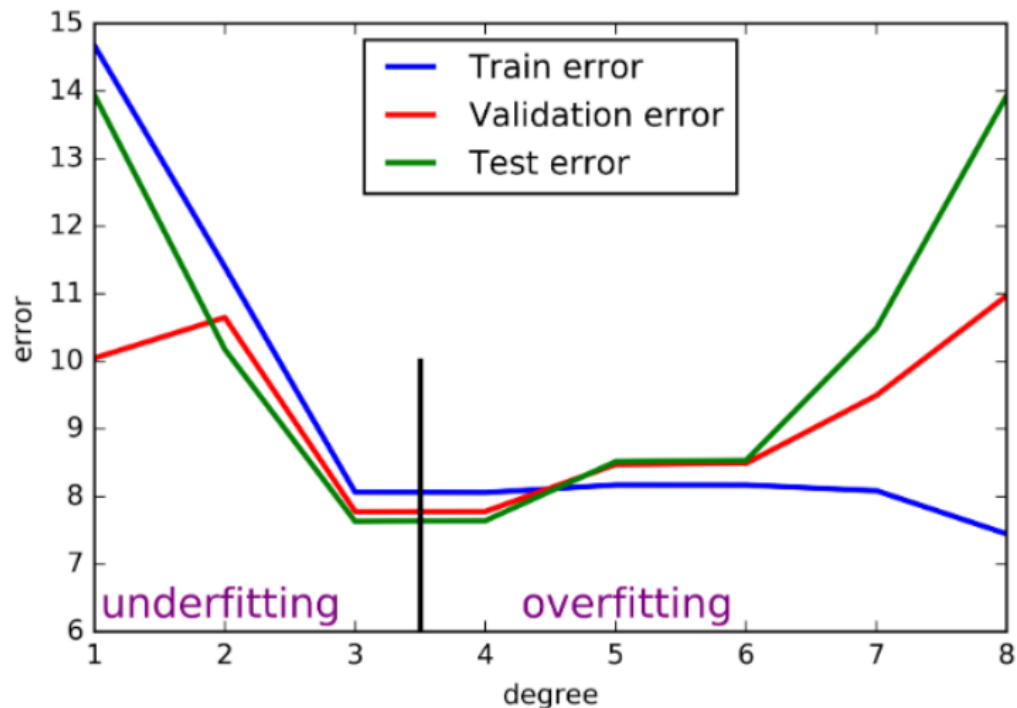
Để giải quyết vấn đề nêu trên, phương pháp đơn giản nhất là trích từ tập training data ra một tập con nhỏ (tập con này có vai trò như test data) và thực hiện việc đánh giá mô hình trên tập con nhỏ này. Tập con nhỏ được trích ra từ training set này được gọi là validation set. Lúc này, training set là phần còn lại của training set ban đầu. Train error được tính trên training set mới này và có một khái niệm nữa được định nghĩa tương tự như trên validation error, tức là error được tính trên tập validation.

Với khái niệm mới này, ta sẽ tìm mô hình sao cho cả train error và validation error đều nhỏ, qua đó có thể dự đoán được rằng test error cũng nhỏ. Phương pháp thường được sử dụng là sử dụng nhiều mô hình khác nhau. Mô hình nào cho validation error nhỏ nhất sẽ là mô hình tốt.

Thông thường, ta bắt đầu từ mô hình đơn giản, sau đó tăng dần độ phức tạp của mô hình. Tới khi nào validation error có chiều hướng tăng lên thì chọn mô hình ngay trước đó. Chú ý rằng mô hình càng phức tạp, train error có xu hướng càng nhỏ đi.

Lấy một ví dụ của một mô hình với bậc của đa thức tăng dần từ 1 đến 8 và tập validation gồm 10 điểm được lấy ra từ tập training data. Chúng ta hãy tạm chỉ xét hai

đường màu lam và đỏ, tương ứng với train error và validation error. Khi bậc của đa thức tăng lên, train error có xu hướng giảm. Điều này dễ hiểu vì đa thức bậc càng cao, dữ liệu càng được fit. Quan sát đường màu đỏ, khi bậc của đa thức là 3 hoặc 4 thì validation error thấp, sau đó tăng dần lên. Dựa vào validation error, ta có thể xác định được bậc cần chọn là 3 hoặc 4. Quan sát tiếp đường màu lục, tương ứng với test error, thật là trùng hợp, với bậc bằng 3 hoặc 4, test error cũng đạt giá trị nhỏ nhất, sau đó tăng dần lên.



Hình 1.4 Cách lựa chọn mô hình dựa trên phương pháp validation

Việc không sử dụng test data khi lựa chọn mô hình ở trên nhưng vẫn có được kết quả khả quan vì ta giả sử rằng validation data và test data có chung một đặc điểm nào đó và khi cả hai đều là unseen data, error trên hai tập này sẽ tương đối giống nhau.

1.3.1.2 Cross-validation

Cross validation là một cải tiến của validation với lượng dữ liệu trong tập validation là nhỏ nhưng chất lượng mô hình được đánh giá trên nhiều tập validation khác

nhau. Một cách thường dùng sử dụng là chia tập training ra k tập con không có phần tử chung, có kích thước gần bằng nhau. Tại mỗi lần kiểm thử, được gọi là run, một trong số k tập con được lấy ra làm validate set. Mô hình sẽ được xây dựng dựa vào hợp của $k-1$ tập con còn lại. Mô hình cuối được xác định dựa trên trung bình của các train error và validation error. Cách làm này còn có tên gọi là k -fold cross validation. Khi k bằng với số lượng phần tử trong tập training ban đầu, tức mỗi tập con có đúng 1 phần tử, ta gọi kỹ thuật này là leave-one-out.

1.3.2 Regularization

Một nhược điểm lớn của cross-validation là số lượng training runs tỉ lệ thuận với k . Điều đáng nói là mô hình polynomial như trên chỉ có một tham số cần xác định là bậc của đa thức. Trong các bài toán Machine Learning, lượng tham số cần xác định thường lớn hơn nhiều và khoảng giá trị của mỗi tham số cũng rộng hơn nhiều, chưa kể đến việc có những tham số có thể là số thực. Như vậy, việc chỉ xây dựng một mô hình thôi cũng là đã rất phức tạp rồi. Có một cách giúp số mô hình cần huấn luyện giảm đi nhiều, thậm chí chỉ một mô hình. Cách này có tên gọi chung là regularization.

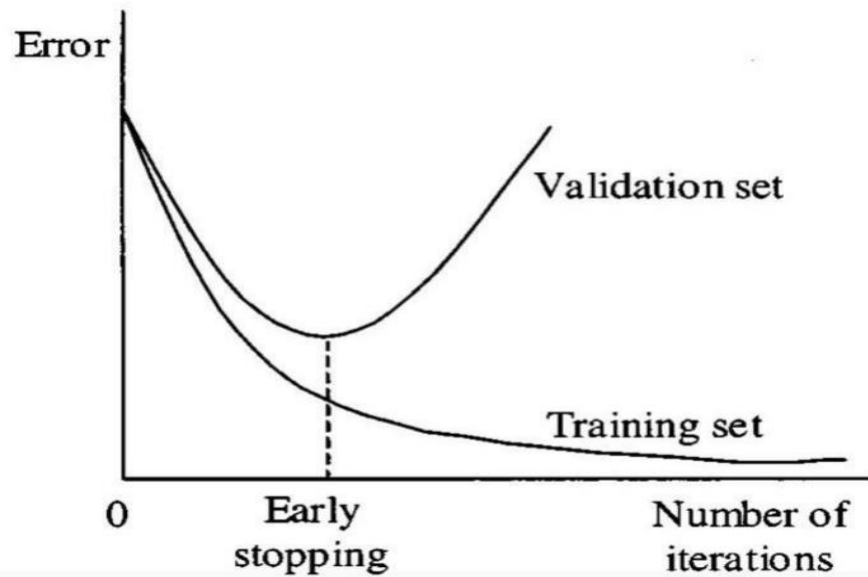
Regularization, một cách cơ bản, là thay đổi mô hình một chút để tránh overfitting trong khi vẫn giữ được tính tổng quát của nó (tính tổng quát là tính mô tả được nhiều dữ liệu, trong cả tập training và test). Một cách cụ thể hơn, ta sẽ tìm cách di chuyển nghiệm của bài toán tối ưu hàm mất mát tới một điểm gần nó. Hướng di chuyển sẽ là hướng làm cho mô hình ít phức tạp hơn mặc dù giá trị của hàm mất mát có tăng lên một chút.

1.3.2.1 Early Stopping

Early Stopping là một hình thức chính quy hóa trong khi đào tạo một mô hình với một phương pháp lặp đi lặp lại, tức là dừng thuật toán trước khi hàm mất mát đạt giá trị quá nhỏ. Vì tất cả các mạng nơ-ron độc quyền bằng cách sử dụng gradient descent, early stopping là một kỹ thuật áp dụng cho tất cả các vấn đề. Phương pháp này cập nhật mô

hình để làm cho nó phù hợp hơn với dữ liệu đào tạo với mỗi lần lặp lại. Đến một điểm, điều này cải thiện hiệu suất của mô hình trên dữ liệu của bộ thử nghiệm.

Tuy nhiên, qua thời điểm đó, việc cải thiện sự phù hợp của mô hình với dữ liệu đào tạo dẫn đến lỗi khái quát hóa tăng lên. Các quy tắc early stopping cung cấp hướng dẫn về số lần lặp lại có thể được chạy trước khi mô hình bắt đầu phù hợp.



Hình 1.5 Minh họa Early stopping

Đối với mạng lưới nơ-ron nhân tạo, quá trình học tập là tìm một tập hợp hoàn hảo của các weights và bias. Các nơ-ron với tốc độ được xác định bởi các dẫn xuất một phần của hàm chi phí: $\frac{\partial C}{\partial W}$ và $\frac{\partial C}{\partial b}$. Vì vậy, tốc độ học tập phụ thuộc vào giá trị của hai dẫn xuất một phần đó, có thể được mô tả tương ứng với các công thức sau, trong đó W_j là j^{th} trọng lượng, b là bias, C là chi phí, X_j là j^{th} đầu vào, y là đầu ra và a là đầu ra khi đầu vào là 1.

$$\frac{\partial C}{\partial W_j} = \frac{1}{n} \sum_x X_j (\sigma(z) - y)$$

$$\frac{\partial C}{\partial b} = \frac{1}{n} \sum_x X_j (\sigma(z) - y)$$

Như chúng ta có thể thấy, sau một số lần lặp lại, lỗi kiểm tra đã bắt đầu tăng lên trong khi lỗi đào tạo vẫn đang giảm. Do đó, mô hình đang quá mức. Vì vậy, để chống lại điều này, chúng ta dừng mô hình tại thời điểm khi điều này bắt đầu xảy ra.

Trên thực tế, để tìm ra được điểm ngừng, cách tốt nhất là theo dõi độ chính xác trên bộ dữ liệu thử nghiệm. Nói cách khác, chúng ta tính toán độ chính xác vào cuối mỗi epoch và ngừng training khi độ chính xác trên dữ liệu thử nghiệm ngừng cải thiện. Nói chung, chúng ta có thể theo dõi độ chính xác trên bộ xác thực thay vì tập test để xác định khi nào nên ngừng training. Nói cách khác, chúng ta sử dụng bộ xác thực để tìm ra một tập hợp các giá trị hoàn hảo cho các siêu tham số và sau đó sử dụng bộ test để thực hiện đánh giá cuối cùng về độ chính xác. Bằng cách này, mức độ khái quát hóa cao hơn có thể được đảm bảo, so với trực tiếp sử dụng dữ liệu bộ test để tìm ra giá trị của siêu tham số.

Chiến lược này đảm bảo rằng, ở mỗi bước của thuật toán lặp đi lặp lại, nó sẽ làm giảm bias nhưng làm tăng khoảng cách. Nhưng, khoảng cách của sự ước tính sẽ không quá cao. Bên cạnh đó, độ phức tạp tính toán sẽ thấp hơn.

1.3.2.2 Network-reduction

Như chúng ta đã biết, noise learning là một trong những nguyên nhân quan trọng gây ra overfitting. Vì vậy, về mặt logic, giảm noise trở thành một hướng nghiên cứu để phù hợp với sự ức chế overfitting. Dựa trên suy nghĩ này, pruning được đề xuất để giảm kích thước của các phân loại final, đặc biệt là trong tree learning. Pruning là một lý thuyết quan trọng được sử dụng để giảm độ phức tạp phân loại bằng cách loại bỏ dữ liệu ít có ý nghĩa hoặc không liên quan, và cuối cùng để ngăn chặn overfitting và cải thiện độ chính xác phân loại. Pre-pruning and post-pruning là hai cách tiếp cận tiêu chuẩn được sử dụng để đối phó với noise: chức năng của các thuật toán Pre-pruning trong quá trình học tập.

Thông thường, ta sử dụng các tiêu chí dừng để xác định khi nào nên ngừng thêm các điều kiện vào quy tắc hoặc thêm quy tắc vào mô hình, chẳng hạn như mã hóa giới hạn chiều dài dựa trên việc đánh giá chi phí mã hóa; kiểm tra ý nghĩa dựa trên sự khác biệt đáng kể giữa phân bố các ví dụ tích cực và tiêu cực; tiêu chí dừng cắt giảm dựa trên ngưỡng được xác định trước. Post-pruning chia tập training thành hai tập con: growing set and pruning set. So với trước khi học tập, các thuật toán post-pruning bỏ qua các vấn đề overfitting trong quá trình học tập trên bộ phát triển. Thay vào đó, chúng ngăn chặn overfitting thông qua việc xóa các điều kiện và quy tắc khỏi mô hình được tạo ra trong quá trình học. Cách tiếp cận này chính xác hơn nhiều, và tuy nhiên, kém hiệu quả hơn.

1.3.2.3 Expansion of the training data (Mở rộng tập dữ liệu training)

Trong học máy, thuật toán không phải là chìa khóa duy nhất ảnh hưởng đến độ chính xác phân loại cuối cùng. Hiệu suất của nó có thể bị ảnh hưởng đáng kể bởi số lượng và chất lượng của bộ dữ liệu training trong nhiều trường hợp, đặc biệt là trong lĩnh vực học tập có giám sát. Mô hình training thực sự là một quá trình điều chỉnh siêu tham số của nó. Các thông số được điều chỉnh tốt tạo ra sự cân bằng tốt giữa độ chính xác của training và thường xuyên, và sau đó ức chế tác dụng của overfitting, cũng như underfitting.

Để điều chỉnh các thông số này, mô hình cần đủ mẫu để học. Số lượng mẫu tỷ lệ thuận với số lượng thông số. Và càng có mô hình phức tạp, càng cần phải điều chỉnh nhiều thông số hơn. Nói cách khác, một bộ dữ liệu mở rộng có thể cải thiện độ chính xác dự đoán ở một mức độ lớn, đặc biệt là trong các mô hình phức tạp. Đó là lý do tại sao data augmentation (tăng cường dữ liệu) được sử dụng rộng rãi và được chứng minh là có hiệu quả như một chiến lược chung để cải thiện hiệu suất khái quát hóa của các mô hình trong nhiều lĩnh vực ứng dụng, chẳng hạn như nhận dạng mẫu và xử lý hình ảnh.

Tuy nhiên, kích thước khổng lồ của dữ liệu chắc chắn sẽ làm tăng thời gian training. Hơn nữa, dữ liệu training có thể tốn kém hoặc khó có được vì nó cần rất nhiều

sự can thiệp của con người, như labeling (dán nhãn). Để giải quyết vấn đề này, chúng ta có thể thao tác dữ liệu hiện có để tạo ra một số dữ liệu mới. Nói tóm lại, chủ yếu có bốn cách tiếp cận để mở rộng bộ training:

- Thu thập thêm dữ liệu training
- Thêm một số noise ngẫu nhiên vào bộ dữ liệu hiện có
- Thu thập lại một số dữ liệu từ bộ dữ liệu hiện có thông qua một số xử lý
- Tạo ra một số dữ liệu mới dựa trên việc phân phối bộ dữ liệu hiện có

1.3.2.4 L1 regularization

Với L1 regularization thì đại lượng được thêm vào là tổng trị tuyệt đối của tất cả các phần tử. Để tìm ra hàm chi phí tối thiểu, L1 regularization sử dụng Lasso Regression (Hồi quy Lasso), một lý thuyết hồi quy tuyến tính. Trong cách tiếp cận này, ta sử dụng cái gọi là khoảng cách taxi-taxi, tổng giá trị tuyệt đối của tất cả các trọng lượng làm penalty term (thời hạn phạt).

$$\Omega(\omega) = \|w\|_1 = \sum_i |w_i|$$

Để giảm thiểu chức năng chi phí, chúng ta cần đặt trọng lượng của một số tính năng là bằng 0. Nói cách khác, ta loại bỏ một số tính năng khỏi mô hình và chỉ giữ những tính năng có giá trị hơn. Bằng cách này, ta có thể có được một mô hình đơn giản hơn, dễ giải thích hơn. Tuy nhiên, đồng thời, ta sẽ đánh mất một số tính năng hữu ích có ảnh hưởng thấp hơn đến đầu ra cuối cùng.

1.3.2.5 L2 regularization (Weight Decay)

Làm giảm hàm mất mát (loss) từ đó giảm weight. Nên norm2 regularization còn được gọi là 'weight decay' (trọng số tiêu biến).

L2 regularization sử dụng lý thuyết “Ridge Regression”.

Trong cách tiếp cận này, ta sử dụng khoảng cách Euclidean làm penalty term.

$$\Omega(\omega) = \|w\|_2 = \sqrt{\sum_i w_i^2}$$

$$\Rightarrow \text{Loss: } J(w) = \frac{1}{2} \|y - Xw\|_2^2 + \lambda \|w\|_2^2$$

So với L1 regularization, cách tiếp cận này làm cho các mạng theo xu hướng học các tính năng có trọng lượng nhỏ. Thay vì từ chối những tính năng ít có giá trị hơn, ta cung cấp cho chúng trọng lượng thấp hơn. Vì vậy, ta nhận được càng nhiều thông tin càng tốt. Trọng lượng lớn chỉ có thể được trao cho các tính năng cải thiện đáng kể chức năng chi phí ban đầu.

1.3.2.6 Dropout

Dropout là một kỹ thuật phổ biến và hiệu quả chống lại sự overfitting trong mạng nơ-ron. Ý tưởng ban đầu của việc dropout là bỏ ngẫu nhiên các đơn vị và các kết nối có liên quan từ các mạng nơ-ron trong quá trình training. Điều này ngăn cản các đơn vị đồng thích nghi quá nhiều. Quy trình chung như sau:

- 1) Thả ngẫu nhiên một nửa số nơ-ron ẩn để xây dựng một mạng lưới đơn giản hơn .
- 2) Đào tạo mạng mỏng bằng cách sử dụng gradient descent ngẫu nhiên
Gradient cho mỗi tham số được tính trung bình trên các trường hợp training trong mỗi mini-batch. Bất kỳ trường hợp training nào không sử dụng tham số đều đóng góp gradient bằng 0 cho tham số đó.
- 3) Khôi phục các nơ-ron bị loại bỏ.
- 4) Loại bỏ ngẫu nhiên một nửa nơ-ron ẩn khỏi mạng được khôi phục để tạo thành một mạng mỏng mới.
- 5) Lặp lại quá trình trên cho đến khi có được một bộ thông số lý tưởng.

Bằng cách tạm thời loại bỏ một số đơn vị khỏi mạng nơ-ron, dropout xấp xỉ hiệu ứng của việc trung bình dự đoán của tất cả các mạng mỏng này. Bằng cách này,

overfitting bị ức chế ở một mức độ nào đó trong khi kết hợp các dự đoán của nhiều mạng nơ-ron lớn khác nhau tại thời điểm thử nghiệm. Bên cạnh đó, dropout làm giảm đáng kể số lượng tính toán. Điều này làm cho nó trở thành một lựa chọn hiệu quả cho các mạng lớn hoặc phức tạp cần nhiều dữ liệu training.

1.4 Thực nghiệm và so sánh giữa các phương pháp

Để thực nghiệm, nhóm em lựa chọn và sử dụng bộ dữ liệu MNIST (Modified National Institute of Standards and Technology database) là một bộ dữ liệu ảnh kích thước 28x28 pixel của các chữ số viết tay từ 0 đến 9. Bộ dữ liệu này là một trong những bộ dữ liệu phổ biến nhất trong lĩnh vực học máy và thị giác máy tính.

Trong thư viện Keras đã có sẵn dataset nên chỉ cần sử dụng phương thức *keras.datasets.mnist.load_data()* là sử dụng được bộ dữ liệu này. Bộ dữ liệu này có 60.000 ảnh cho việc huấn luyện và 10.000 ảnh cho việc kiểm tra.

```
import matplotlib.pyplot as plt
import numpy as np
from tensorflow import keras

(x_train, y_train), (x_test, y_test) =
keras.datasets.mnist.load_data()
#chuẩn hóa dữ liệu và chuyển đổi nhãn thành dạng one-hot vector
x_train = x_train / 255.0
x_test = x_test / 255.0

y_train = keras.utils.to_categorical(y_train, 10)
y_test = keras.utils.to_categorical(y_test, 10)

# Các thông số huấn luyện
epochs = 10 #số lần huấn luyện trên toàn bộ tập dữ liệu.
batch_size = 32 #số lượng mẫu được sử dụng để cập nhật trọng số
trong một lần huấn luyện.
validation_split = 0.2 #tỷ lệ phần trăm dữ liệu được sử dụng để
đánh giá mô hình sau mỗi epoch trong quá trình huấn luyện.
```

Trong đó: `x_train` và `y_train` là các mảng numpy chứa dữ liệu huấn luyện. `x_test` và `y_test` là các mảng numpy chứa dữ liệu kiểm tra. `y_train` và `y_test` là các mảng numpy chứa nhãn tương ứng của từng ảnh.

Chuẩn hóa dữ liệu bằng cách chia giá trị của mỗi pixel cho 255, để giá trị pixel nằm trong khoảng từ 0 đến 1. Chuyển đổi nhãn từ dạng số sang dạng one-hot vector.

Sau đó xây dựng mô hình dựa trên các phương pháp tránh Overfitting.

- Basic Model

```
model_basic = keras.models.Sequential([
    keras.layers.Flatten(input_shape=(28, 28)),
    keras.layers.Dense(128, activation='relu'),
    keras.layers.Dense(64, activation='relu'),
    keras.layers.Dense(10, activation='softmax')
])
model_basic.compile(optimizer='adam',
                    loss='categorical_crossentropy',
                    metrics=['accuracy'])

history_basic = model_basic.fit(x_train, y_train,
                                epochs=epochs,
                                batch_size=batch_size,
                                validation_split=validation_split)
```

- Cross-validation

```
from sklearn.model_selection import KFold

# Sử dụng phương pháp Cross-validation
kf = KFold(n_splits=5, shuffle=True)

model_cross_validation = keras.models.Sequential([
    keras.layers.Flatten(input_shape=(28, 28)),
    keras.layers.Dense(128, activation='relu'),
    keras.layers.Dense(64, activation='relu'),
    keras.layers.Dense(10, activation='softmax')
```

```

    ])

model_cross_validation.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])

train_acc = []
val_acc = []
train_loss = []
val_loss = []

for train_index, val_index in kf.split(x_train):
    x_train_fold, x_val_fold = x_train[train_index],
x_train[val_index]
    y_train_fold, y_val_fold = y_train[train_index],
y_train[val_index]

    # Huấn luyện mô hình
    history = model_cross_validation.fit(x_train_fold,
y_train_fold,
                                     epochs=epochs,
                                     batch_size=batch_size,
                                     validation_data=(x_val_fold,
y_val_fold),
                                     verbose=0)

    # Lưu trữ độ chính xác trên tập huấn luyện và tập
validation của mỗi lần chia
    train_acc.append(history.history['accuracy'])
    val_acc.append(history.history['val_accuracy'])
    train_loss.append(history.history['loss'])
    val_loss.append(history.history['val_loss'])

# Tính trung bình độ chính xác của các lần chia
Cross_train_acc = np.mean(train_acc, axis=0)
Cross_val_acc = np.mean(val_acc, axis=0)

# Tính trung bình độ Loss của các lần chia
Cross_train_loss = np.mean(train_loss, axis=0)
Cross_val_loss = np.mean(val_loss, axis=0)

```


- Early Stopping

[illegible]

```

Epoch 1/10
1500/1500 [=====] - 5s 3ms/step - loss: 0.2645 - accuracy: 0.9229 - val_loss: 0.1467 - val_accuracy: 0.9538
Epoch 2/10
1500/1500 [=====] - 5s 3ms/step - loss: 0.1131 - accuracy: 0.9657 - val_loss: 0.1068 - val_accuracy: 0.9663
Epoch 3/10
1500/1500 [=====] - 4s 3ms/step - loss: 0.0776 - accuracy: 0.9758 - val_loss: 0.1053 - val_accuracy: 0.9679
Epoch 4/10
1500/1500 [=====] - 4s 3ms/step - loss: 0.0578 - accuracy: 0.9815 - val_loss: 0.0954 - val_accuracy: 0.9715
Epoch 5/10
1500/1500 [=====] - 4s 3ms/step - loss: 0.0456 - accuracy: 0.9856 - val_loss: 0.0947 - val_accuracy: 0.9732
Epoch 6/10
1500/1500 [=====] - 4s 3ms/step - loss: 0.0356 - accuracy: 0.9884 - val_loss: 0.0961 - val_accuracy: 0.9743
Epoch 7/10
1500/1500 [=====] - 4s 3ms/step - loss: 0.0298 - accuracy: 0.9904 - val_loss: 0.0859 - val_accuracy: 0.9749
Epoch 8/10
1500/1500 [=====] - 4s 3ms/step - loss: 0.0257 - accuracy: 0.9914 - val_loss: 0.1035 - val_accuracy: 0.9727
Epoch 9/10
1500/1500 [=====] - 4s 3ms/step - loss: 0.0212 - accuracy: 0.9932 - val_loss: 0.0994 - val_accuracy: 0.9761
Epoch 9: early stopping

```

Hình 1.6 Thực hiện chạy Early stopping

- Network Reduction

```

model_network_reduction = keras.models.Sequential([
    keras.layers.Flatten(input_shape=(28, 28)),
    keras.layers.Dense(64, activation='relu'),
    keras.layers.Dense(10, activation='softmax')
])

model_network_reduction.compile(optimizer='adam',
                                loss='categorical_crossentropy',
                                metrics=['accuracy'])

history_network_reduction =
model_network_reduction.fit(x_train, y_train,
                            epochs=
epochs,
                            batch_s
ize=batch_size,
                            validat
ion_split=validation_split)

```

- Expansion of the training data

```

#thực hiện các phép biến đổi ảnh ngẫu nhiên để tạo ra các dữ
liệu mới từ các ảnh gốc.
expansion = keras.Sequential([
    keras.layers.experimental.preprocessing.RandomRotation(0.1)
,

```

```

        keras.layers.experimental.preprocessing.RandomZoom(0.1),
        keras.layers.experimental.preprocessing.RandomContrast(0.1)
    ])

model_expansion = keras.models.Sequential([
    expansion,
    keras.layers.Flatten(input_shape=(28, 28)),
    keras.layers.Dense(128, activation='relu'),
    keras.layers.Dense(64, activation='relu'),
    keras.layers.Dense(10, activation='softmax')
])

model_expansion.compile(optimizer='adam',
                        loss='categorical_crossentropy',
                        metrics=['accuracy'])
x_train_expanded = expansion(x_train.reshape(-1, 28, 28,
1)).numpy()
history_expansion = model_expansion.fit(x_train_expanded,
y_train,
                                     epochs=epochs,
                                     batch_size=batch_size,
                                     validation_split=validation_split)

```

- L1 regularization

```

model_l1 = keras.models.Sequential([
    keras.layers.Flatten(input_shape=(28, 28)),
    keras.layers.Dense(128, activation='relu',
kernel_regularizer=keras.regularizers.l1(0.01)), #hệ số lambda
bằng 0.01
    keras.layers.Dense(64, activation='relu',
kernel_regularizer=keras.regularizers.l1(0.01)),
    keras.layers.Dense(10, activation='softmax')
])

model_l1.compile(optimizer='adam',
                 loss='categorical_crossentropy',
                 metrics=['accuracy'])

```

```
history_l1 = model_l1.fit(x_train, y_train,
                          epochs=epochs,
                          batch_size=batch_size,
                          validation_split=validation_split)
```

- L2 regularization

```
model_l2 = keras.models.Sequential([
    keras.layers.Flatten(input_shape=(28, 28)),
    keras.layers.Dense(128, activation='relu',
    kernel_regularizer=keras.regularizers.l2(0.01)),
    keras.layers.Dense(64, activation='relu',
    kernel_regularizer=keras.regularizers.l2(0.01)),
    keras.layers.Dense(10, activation='softmax')
])

model_l2.compile(optimizer='adam',
                 loss='categorical_crossentropy',
                 metrics=['accuracy'])

history_l2 = model_l2.fit(x_train, y_train,
                          epochs=epochs,
                          batch_size=batch_size,
                          validation_split=validation_split)
```

- Dropout

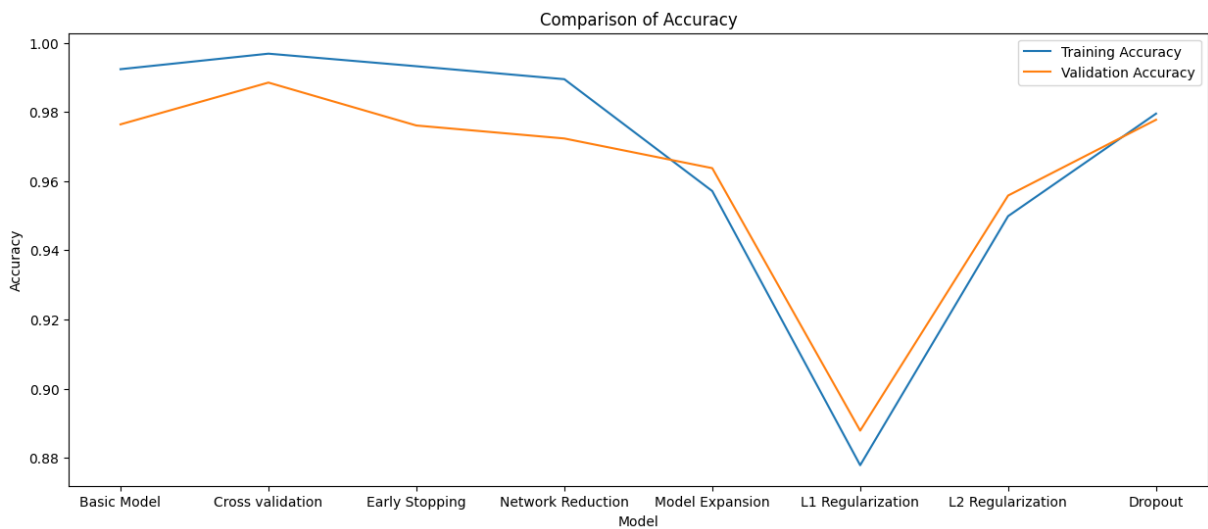
```
model_dropout = keras.models.Sequential([
    keras.layers.Flatten(input_shape=(28, 28)),
    keras.layers.Dense(128, activation='relu'),
    keras.layers.Dropout(0.2), #dropout rate (0.2->0.5)
    keras.layers.Dense(64, activation='relu'),
    keras.layers.Dropout(0.2),
    keras.layers.Dense(10, activation='softmax')
])

model_dropout.compile(optimizer='adam',
                     loss='categorical_crossentropy',
```

```
metrics=['accuracy'])

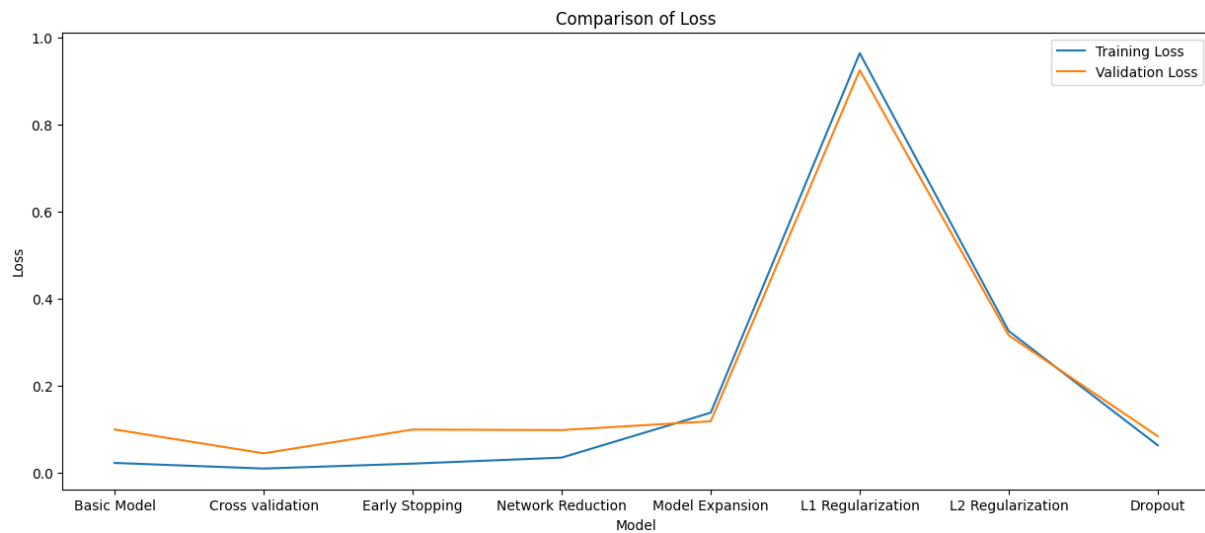
history_dropout = model_dropout.fit(x_train, y_train,
                                     epochs=epochs,
                                     batch_size=batch_size,
                                     validation_split=validation
                                     _split)
```

Thực hiện vẽ biểu đồ so sánh giá trị Accuracy trên tập huấn luyện và tập validation của mỗi mô hình.



Hình 1.7 So sánh giá trị Accuaracy

Thực hiện vẽ biểu đồ so sánh giá trị Loss trên tập huấn luyện và tập validation của mỗi mô hình.



Hình 1.8 So sánh giá trị Loss

PHẦN 2 – XÂY DỰNG RECOMMENDATION SYSTEM

Tổng quan: Trong bài viết này, chúng em sẽ xây dựng một Recommendation System cho bộ dữ liệu Amazon Fine Food Reviews. Chúng em sẽ sử dụng phương pháp Collaborative Filtering để đưa ra các đề xuất sản phẩm cho người dùng dựa trên lịch sử mua hàng và xếp hạng của họ.

2.1 Bộ dữ liệu Amazon Fine Food Reviews

Bộ dữ liệu Amazon Fine Food Reviews chứa hơn 500.000 đánh giá của người dùng về các sản phẩm thực phẩm trên Amazon từ tháng 10 năm 1999 đến tháng 10 năm 2012. Bộ dữ liệu này bao gồm các trường như tên người dùng, tên sản phẩm, xếp hạng và bình luận.

Dữ liệu bao gồm:

- Các nhận xét từ Tháng 10 năm 1999 - Tháng 10 năm 2012
- Tổng cộng 568,454 nhận xét
- 256,059 người dùng 74,258 sản phẩm
- 260 người dùng có hơn 50 nhận xét.

Bộ dữ liệu: Amazon Fine Food Reviews [\[11\]](#)

2.2 Mô hình Recommendation Systems

Mô hình Recommendation Systems là một mô hình dự đoán được sử dụng để đưa ra các gợi ý cho người dùng về các sản phẩm mà họ có thể quan tâm dựa trên các thông tin về lịch sử mua hàng và xếp hạng của họ. Mô hình này có thể được xây dựng bằng các phương pháp khác nhau như Collaborative Filtering, Content-Based Filtering và Hybrid Filtering.

Trong Collaborative Filtering, mô hình sử dụng thông tin về lịch sử mua hàng và xếp hạng của các người dùng khác để đưa ra các gợi ý cho người dùng hiện tại. Trong

Content-Based Filtering, mô hình sử dụng các đặc trưng của sản phẩm (như tên, mô tả, danh mục) để đưa ra các gợi ý cho người dùng.

Mô hình Recommendation Systems có rất nhiều ứng dụng trong thực tế, ví dụ như trong các trang thương mại điện tử để đưa ra các gợi ý sản phẩm cho khách hàng, trong các trang web chia sẻ video để đưa ra các gợi ý video cho người dùng, và trong các trang web tin tức để đưa ra các gợi ý bài viết cho người đọc.

2.2.1 Phương pháp Collaborative Filtering

Collaborative Filtering là phương pháp khá phổ biến trong xây dựng các hệ thống đề xuất sản phẩm (Recommendation Systems). Nó hoạt động dựa trên sự tương tác giữa các người dùng và sản phẩm, tức là những gì mà một người dùng đã mua hoặc xem trước đó. Với một người dùng mới, hệ thống sẽ dựa trên các đánh giá, xếp hạng của các người dùng khác có sở thích tương tự để đưa ra các sản phẩm được đề xuất cho người dùng đó.

Phương pháp này sử dụng ma trận đánh giá sản phẩm của các người dùng để xác định sự tương đồng giữa các người dùng. Sau đó, các sản phẩm được đề xuất dựa trên các sản phẩm mà người dùng có sở thích tương tự đã mua hoặc xem trước đó.

2.3 Thư viện *Surprise* trong Recommendation Systems

Thư viện *surprise* là một thư viện mã nguồn mở được phát triển trên ngôn ngữ Python, được sử dụng cho các bài toán Recommendation System (Hệ thống đề xuất sản phẩm). Thư viện này cung cấp các thuật toán phổ biến trong lĩnh vực này như Collaborative Filtering, Singular Value Decomposition (SVD), Neighborhood-based algorithms, Matrix Factorization,.. Nó cũng cung cấp các công cụ để đọc, xử lý và đánh giá dữ liệu đầu vào cho các bài toán recommendation system. Thư viện surprise được sử dụng rộng rãi trong cộng đồng Machine Learning và là một trong những thư viện phổ biến nhất cho bài toán recommendation system.

2.4 Recommendation Systems với Amazon Fine Food Reviews

Tiến hành import thư viện cần thiết và đọc file *reviews.csv* sau đó hiển thị mô tả của bộ dữ liệu.

```
import pandas as pd
from surprise import Dataset, Reader
from surprise.model_selection import train_test_split
from surprise import SVD
from surprise import accuracy

# Đọc dữ liệu từ tập tin csv
filename = 'reviews.csv'
df = pd.read_csv(filename)
df.info()
df.describe()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 568454 entries, 0 to 568453
Data columns (total 10 columns):
#   Column                      Non-Null Count  Dtype
---  -
0   Id                          568454 non-null int64
1   ProductId                  568454 non-null object
2   UserId                     568454 non-null object
3   ProfileName                568438 non-null object
4   HelpfulnessNumerator       568454 non-null int64
5   HelpfulnessDenominator     568454 non-null int64
6   Score                      568454 non-null int64
7   Time                      568454 non-null int64
8   Summary                   568427 non-null object
9   Text                      568454 non-null object
dtypes: int64(5), object(5)
memory usage: 43.4+ MB
```

	Id	HelpfulnessNumerator	HelpfulnessDenominator	Score	Time
count	568454.000000	568454.000000	568454.000000	568454.000000	5.684540e+05
mean	284227.500000	1.743817	2.22881	4.183199	1.296257e+09
std	164098.679298	7.636513	8.28974	1.310436	4.804331e+07
min	1.000000	0.000000	0.00000	1.000000	9.393408e+08
25%	142114.250000	0.000000	0.00000	4.000000	1.271290e+09
50%	284227.500000	0.000000	1.00000	5.000000	1.311120e+09
75%	426340.750000	2.000000	2.00000	5.000000	1.332720e+09
max	568454.000000	866.000000	923.00000	5.000000	1.351210e+09

Hình 2.1 Thông tin và mô tả tổng quan về bộ dữ liệu

2.4.1 Tiền xử lý dữ liệu

Loại bỏ các giá trị NaN

```
# loại bỏ giá trị NaN
df = df.dropna()
print(df.isnull().sum())
```

Id	0	Id	0
ProductId	0	ProductId	0
UserId	0	UserId	0
ProfileName	16	ProfileName	0
HelpfulnessNumerator	0	HelpfulnessNumerator	0
HelpfulnessDenominator	0	HelpfulnessDenominator	0
Score	0	Score	0
Time	0	Time	0
Summary	27	Summary	0
Text	0	Text	0
dtype: int64		dtype: int64	

Hình 2.2 Kết quả trước và sau khi xử lý missing values

```
# Xử lý dữ liệu thành ma trận rating
reader = Reader(rating_scale=(1, 5)) # được sử dụng để đọc dữ
liệu đầu vào và định dạng chúng cho phù hợp với các phương thức
của Surprise

data = Dataset.load_from_df(df[['UserId', 'ProductId',
'Score']], reader)

# Chia dữ liệu thành tập huấn luyện và tập kiểm tra
trainset, testset = train_test_split(data, test_size=0.25)
```

Đầu tiên tạo một **Reader** với **rating_scale** được đặt là (1, 5), mô tả khoảng giá trị của ratings trong tập dữ liệu (từ 1 đến 5). (Trong thư viện Surprise, Reader là một lớp đại diện cho cách dữ liệu đầu vào được đọc và phân tích. Nó cung cấp các phương thức để đọc dữ liệu từ tệp và chuyển đổi chúng thành dạng mà các lớp khác của Surprise có thể sử dụng để xây dựng các mô hình Recommendation Systems)

Sau đó sử dụng *Reader* để đọc dữ liệu được lấy từ cột *UserId*, *ProductId* và *Score* của *dataframe df*.

Dataset sau đó được tạo bằng cách sử dụng phương thức *load_from_df* và *Reader* đã được tạo trước đó. *Dataset* là dùng để lưu trữ dữ liệu và thực hiện các phương pháp học máy trong Surprise.

Train_test_split() được sử dụng để chia tập dữ liệu *data* thành 2 phần: tập huấn luyện (trainset) và tập kiểm tra (testset). Tham số *test_size* được sử dụng để xác định tỷ lệ phần trăm dữ liệu được chọn để làm tập kiểm tra. Ở đây, chúng em chọn tỷ lệ là 0.25, tức là 25% dữ liệu được chọn làm tập kiểm tra.

2.4.2 Xây dựng và đánh giá mô hình Singular Value Decomposition

Mô hình SVD hoạt động bằng cách phân rã ma trận đánh giá thành các ma trận con thấp chiều hơn, giúp giảm số chiều của ma trận và tối ưu hóa việc tính toán đánh giá dự đoán. Sau khi phân tích ma trận, mô hình sử dụng các yếu tố ẩn để tạo ra dự đoán xếp hạng cho các mặt hàng chưa được đánh giá bởi người dùng.

Điểm mạnh của mô hình SVD là khả năng xử lý được dữ liệu lớn và khả năng dự đoán chính xác các đánh giá của người dùng cho các mặt hàng mà họ chưa xem. Tuy nhiên, điểm yếu của mô hình này là không thể xử lý được dữ liệu thưa và không thể giải quyết các vấn đề liên quan đến bất đối xứng của đánh giá sản phẩm giữa các người dùng.

Với đánh giá chỉ số Mean Absolute Error độ đo đánh giá sự chênh lệch giữa các giá trị dự đoán và giá trị thực tế trong các bài toán dự đoán. Nó được tính bằng cách lấy trung bình giá trị tuyệt đối của sự chênh lệch giữa các giá trị dự đoán và giá trị thực tế.

```
# Xây dựng mô hình Recommendation Systems
model = SVD() #mô hình SVD (Singular Value Decomposition)
model.fit(trainset)

# Đánh giá mô hình trên tập kiểm tra và tính toán độ chính xác của mô hình
predictions = model.test(testset)
```

```
# chỉ số RMSE
accuracy.rmse(predictions)
# Chỉ số MAE
accuracy.mae(predictions)
```

RMSE: 1.0914

MAE: 0.8007

0.8007009477056345

Hình 2.3 Kết quả các chỉ số RMSE và MAE của mô hình

2.4.3 Tạo recommendations cho một người dùng

- Với 1 người mới

```
# Tạo recommendations Cho một người dùng mới
user_id = 'A1B2C3D40A2'
items = df['ProductId'].unique()
predictions = []
for item_id in items:
    pred = model.predict(user_id, item_id)
    predictions.append((item_id, pred.est))

# sắp xếp predictions bởi tỷ lệ rating
predictions.sort(key=lambda x: x[1], reverse=True)

# Tạo bản đồ từ điển ánh xạ ID sản phẩm với Summary
id_to_name = {}
for row in df[['ProductId',
'Summary']].drop_duplicates().itertuples():
    id_to_name[row.ProductId] = row.Summary

# In ra top 10 sản phẩm được đề xuất bằng tên sản phẩm
print("Top 10 recommended cho Thiên An:")
for i in range(10):
    product_id = predictions[i][0]
    Summary = id_to_name[product_id]
    print(f"{i+1}. {predictions[i][0]} . {Summary}")
```

```

... Top 10 recommended cho Thiên An:
1. B000ED9L9E . Multi-Use, Nutricious, Easy, Perfect!
2. B005EL6VOY . I love this oatmeal
3. B000NMJWZO . Best gluten-free baking mix ever
4. B003OZX4ME . Questionable ingredients. This is junk food disguised as "spring water". Blech.
5. B000LKXJW0 . Severely allergic son gives two itchy thumbs up!
6. B004AFODLI . I will never eat regular pancakes again!!!!!!!!!!!!
7. B001IZ9ME6 . The best mint you can't find in store
8. B003LECIDE . Sodium Benzoate is Dangerous- It's in EZ sweetz
9. B001E5E10A . if you ever go outdoors, this tea is for you
10. B004JLRYC8 . Ideal drink

```

Hình 2.4 Kết quả 10 đề xuất cho người mới

- Với người dùng đã có trong cơ sở dữ liệu

```

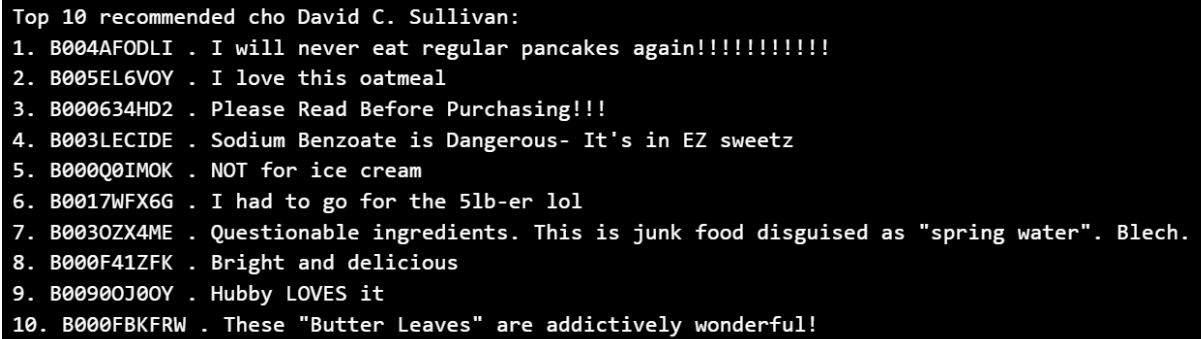
# recommendations cho 1 người dùng trong data
user_id = 'A1SP2KVKFXXRU1'
items = df['ProductId'].unique()
predictions = []
for item_id in items:
    pred = model.predict(user_id, item_id)
    predictions.append((item_id, pred.est))

# sắp xếp predictions bởi tỷ lệ rating
predictions.sort(key=lambda x: x[1], reverse=True)

# Tạo bản đồ từ điển ánh xạ ID sản phẩm với Summary
id_to_name = {}
for row in df[['ProductId',
'Summary']].drop_duplicates().itertuples():
    id_to_name[row.ProductId] = row.Summary

# In ra top 10 sản phẩm được đề xuất bằng tên sản phẩm
print("Top 10 recommended cho David C. Sullivan:")
for i in range(10):
    product_id = predictions[i][0]
    Summary = id_to_name[product_id]
    print(f"{i+1}. {predictions[i][0]} . {Summary}")

```



```
Top 10 recommended cho David C. Sullivan:
1. B004AFODLI . I will never eat regular pancakes again!!!!!!!!!!!!
2. B005EL6VOY . I love this oatmeal
3. B000634HD2 . Please Read Before Purchasing!!!
4. B003LECIDIE . Sodium Benzoate is Dangerous- It's in EZ sweetz
5. B000Q0IMOK . NOT for ice cream
6. B0017WFX6G . I had to go for the 5lb-er lol
7. B0030ZX4ME . Questionable ingredients. This is junk food disguised as "spring water". Blech.
8. B000F41ZFK . Bright and delicious
9. B00900J00Y . Hubby LOVES it
10. B000FBKFRW . These "Butter Leaves" are addictively wonderful!
```

Hình 2.5 Kết quả 10 đề xuất cho người đã có trong cơ sở dữ liệu

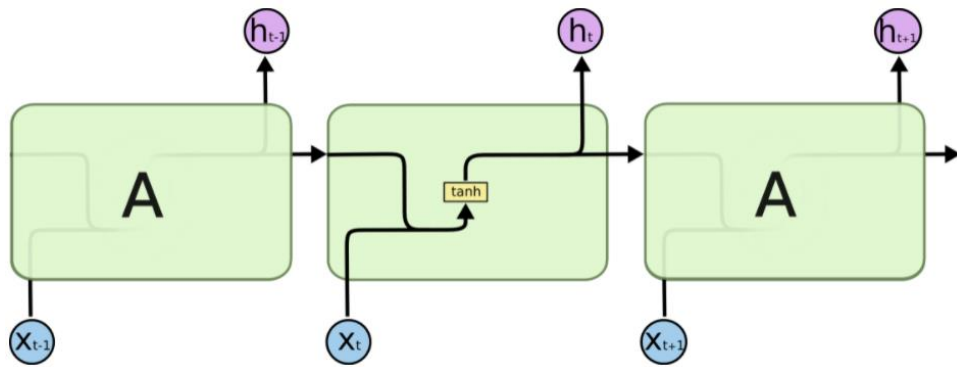
PHẦN 3 – MÔ HÌNH HỌC SÂU LONG SHORT TERM MEMORY

3.1 Cơ sở lý thuyết

Long Short Term Memory (LSTM) là dạng cải tiến của mạng RNN, một mạng tuần tự cho phép thông tin tồn tại. Nó có khả năng xử lý vấn đề gradient biến mất mà RNN phải đối mặt. Mạng nơ-ron tuần hoàn được còn được gọi là RNN được sử dụng cho bộ nhớ liên tục.

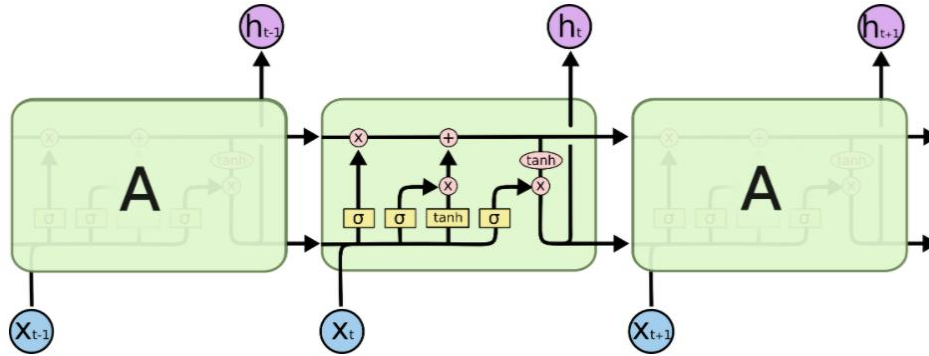
Các RNN hoạt động bằng cách ghi nhớ thông tin trước đó và sử dụng nó để xử lý đầu vào cho hiện tại. Thiếu sót của mạng RNN là không thể nhớ các phụ thuộc lâu dài do gradient biến mất. LSMT được thiết kế để tránh các vấn đề phụ thuộc lâu dài.

Mọi mạng hồi quy đều có dạng là một chuỗi các mô-đun lặp đi lặp lại của mạng nơ-ron. Với mạng RNN chuẩn thì các mô-đun có cấu trúc rất đơn giản, thường là một tầng *tanh*.



Hình 3.1 Minh họa mạng RNN

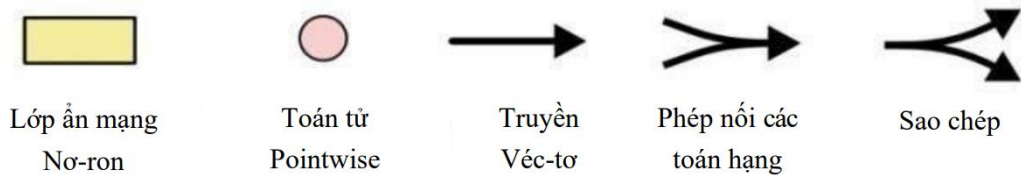
LSTM cũng có kiến trúc dạng chuỗi như vậy, nhưng các mô-đun trong nó có cấu trúc khác với mạng RNN chuẩn. Thay vì chỉ có một mạng nơ-ron, chúng có tới 4 tầng tương tác với nhau một cách rất đặc biệt.



Hình 3.2 Cấu trúc mạng LSTM

LSTM có các thành phần cơ bản sau:

- Trạng thái tế bào (cell state)
- Cổng (gates)
- Sigmoid
- Tanh

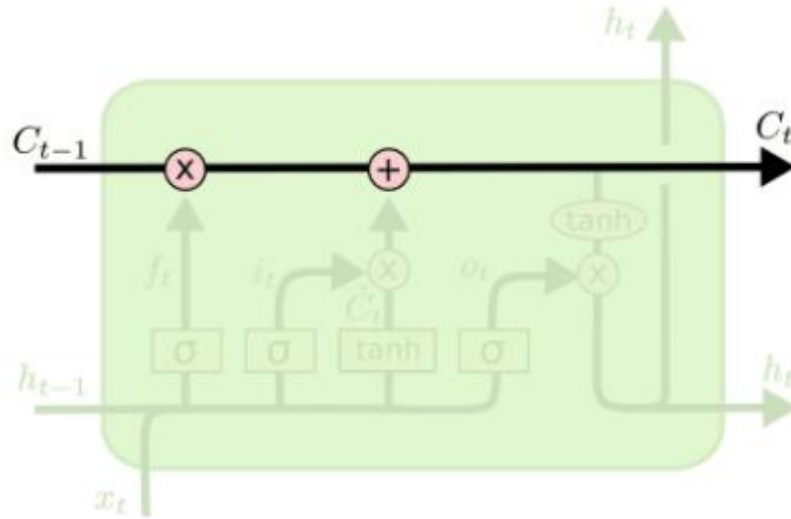


Hình 3.3 Các kí hiệu sử dụng trong mạng LSTM

3.1.1 Ý tưởng cốt lõi

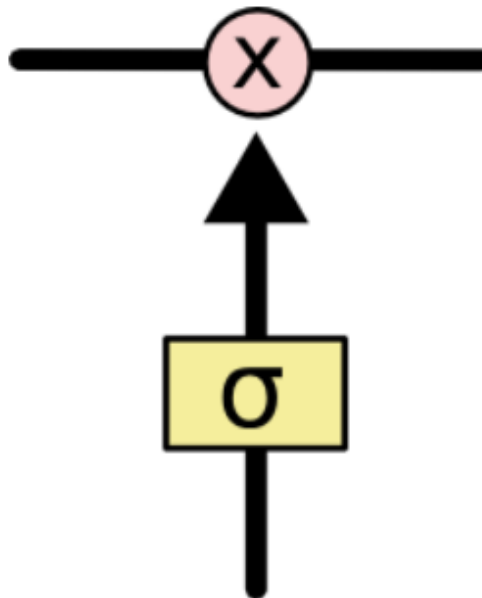
Chìa khóa của LSTM là trạng thái của tế bào (cell-state) chính là đường chạy thông qua của sơ đồ hình vẽ.

Trạng thái tế bào là một dạng giống như băng truyền. Nó chạy xuyên suốt tất cả các mắt xích (các nút mạng) và chỉ tương tác tuyến tính đôi chút. Vì vậy mà các thông tin có thể dễ dàng truyền đi thông suốt mà không sợ biến đổi.



Hình 3.4 Minh họa trạng thái tế bào

LSTM có khả năng bỏ đi hoặc thêm vào các thông tin cần thiết cho trạng thái tế bào, chúng được chỉnh cân thận bởi các nhóm được gọi là cổng (gate). Các cổng là nơi sàng lọc thông tin đi qua nó, chúng được kết hợp bởi một tầng mạng sigmoid và một phép nhân.



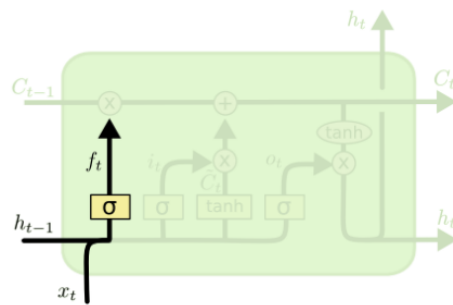
Hình 3.5 Cổng trạng thái LSTM

Tầng sigmoid sẽ cho đầu ra là một số trong đoạn $[0,1]$, mô tả có bao nhiêu thông tin có thể được thông qua. Khi đầu ra là 0 thì có nghĩa không cho thông tin nào qua cả, còn khi 1 thì có nghĩa là cho tất cả thông tin đi qua nó.

Một LSTM gồm có 3 công như vậy để duy trì và điều hành trạng thái của tế bào.

3.1.2 Phân tích mô hình LSTM

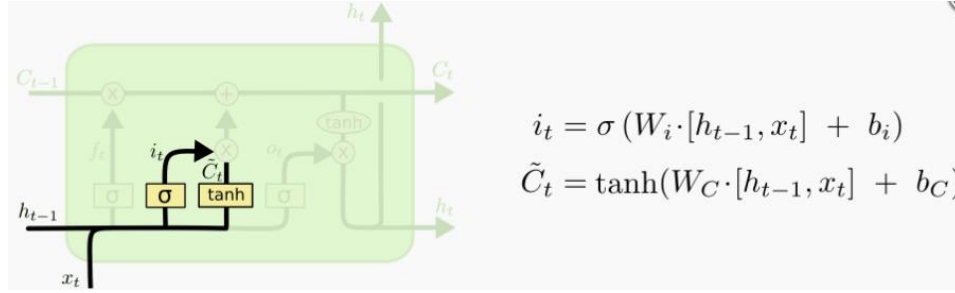
Bước đầu tiên của LSTM là quyết định xem thông tin nào cần bỏ đi từ trạng thái tế bào. Quyết định này được đưa ra bởi tầng sigmoid- gọi là **forget gate layer**. Nó sẽ lấy đầu vào là h_{t-1} và x_t rồi đưa kết quả là một số trong khoảng $[0,1]$ cho mỗi số trong trạng thái tế bào C_{t-1} . Đầu ra là một thể hiện rằng nó giữ lại toàn bộ thông tin lại, còn 0 chỉ là rằng toàn bộ thông tin sẽ bị bỏ đi.



$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

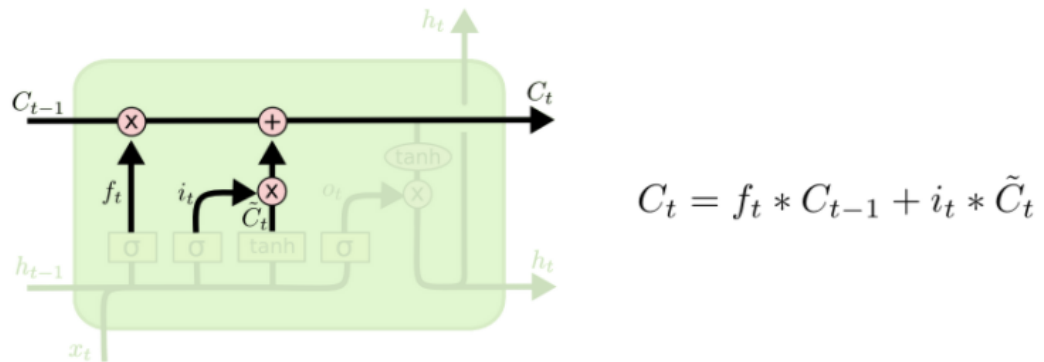
Hình 3.6 Minh họa về input gate layer

Bước tiếp theo là quyết định xem thông tin mới nào ta sẽ lưu vào trạng thái tế bào. Việc này gồm 2 phần, đầu tiên là sử dụng một tầng sigmoid được gọi là **input gate layer** để quyết định giá trị nào ta sẽ cập nhật. Tiếp theo là một tầng tanh tạo ra một vector cho giá trị mới nhằm thêm vào cho trạng thái. Trong bước tiếp theo, ta kết hợp 2 giá trị đó lại để tạo ra một cập nhật cho trạng thái.



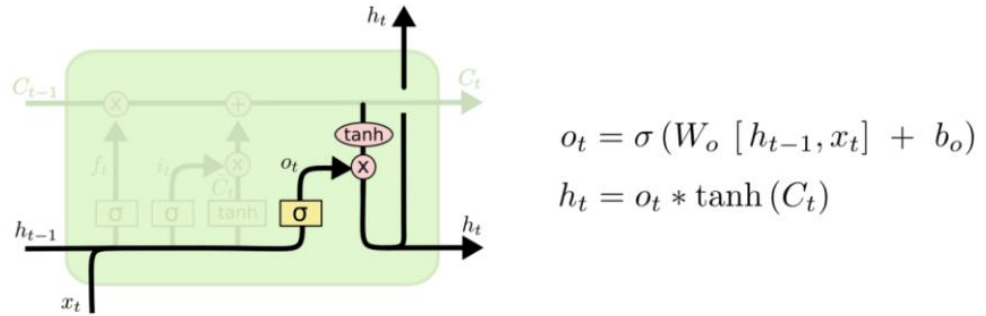
Hình 3.7 Minh họa về cập nhật

Giờ là lúc cập nhật trạng thái tế bào cũ C_{t-1} thành trạng thái mới C_t . Ở các bước trước đó đã quyết định những việc cần làm, nên giờ ta chỉ cần thực hiện là xong. Ta sẽ nhân trạng thái cũ với f_t để bỏ đi những thông tin ta quyết định quên lúc trước. Sau đó cộng thêm $i_t C_t$. Trạng thái mới thu được này phụ thuộc vào việc ta quyết định cập nhật mỗi giá trị trạng thái ra sao.



Hình 3.8 Minh họa về quyết định đầu ra

Cuối cùng, ta cần quyết định xem ta muốn đầu ra là gì. Giá trị đầu ra sẽ dựa vào trạng thái tế bào, nhưng sẽ được tiếp tục sàng lọc. Đầu tiên, ta chạy một tầng sigmoid để quyết định phần nào của trạng thái tế bào ta muốn xuất ra. Sau đó, ta đưa nó trạng thái tế bào qua một hàm tanh để co giá trị nó về khoảng $[-1, 1]$, và nhân nó với đầu ra của cổng sigmoid để được giá trị đầu ra ta mong muốn.

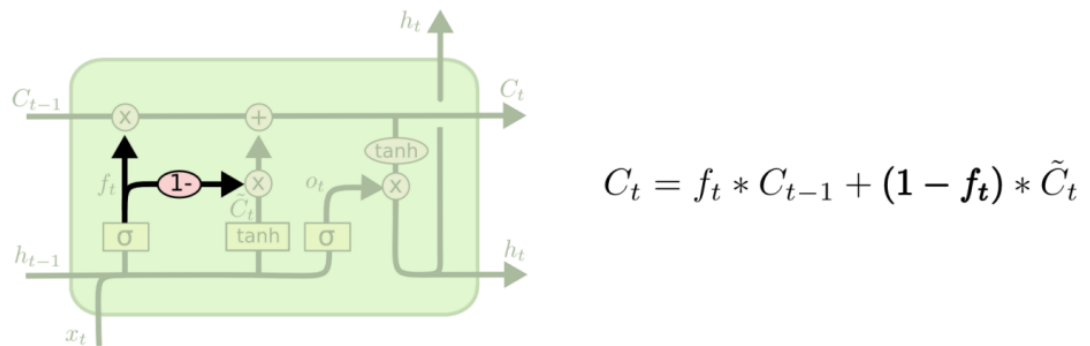


Hình 3.9 Minh họa đầu ra

3.2 Một số biến thể

3.2.1 Biến thể nối cổng loại trừ và cổng đầu vào

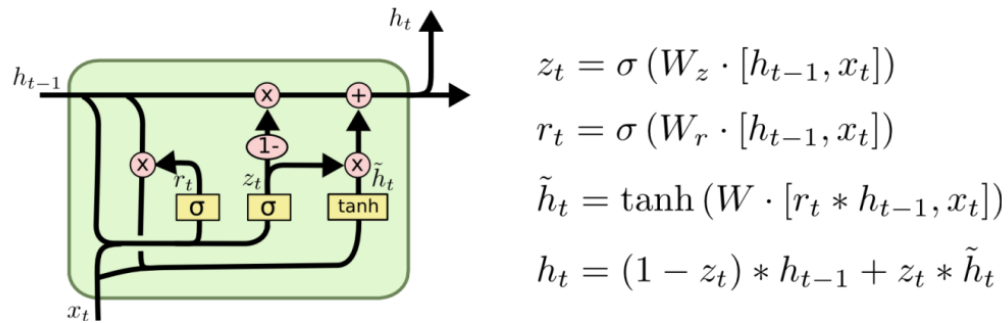
Thay vì phân tách các quyết định thông tin loại trừ và thông tin mới thêm vào, ta sẽ quyết định chúng cùng với nhau luôn. Bỏ đi thông tin khi mà ta thay thế nó bằng thông tin mới đưa vào. Đưa thông tin mới vào khi ta bỏ thông tin cũ nào đó đi.



Hình 3.10 Biến thể nối 2 cổng loại trừ và đầu vào với nhau

3.2.2 Biến thể Gated Recurrent Unit

Nó kết hợp các cổng loại trừ và đầu vào thành một cổng “cổng cập nhập” (update gate). Nó cũng hợp trạng thái tế bào và trạng thái ẩn với nhau tạo ra một thay đổi khác. Kết quả là mô hình của ta sẽ đơn giản hơn mô hình LSTM chuẩn và ngày càng trở nên phổ biến.



Hình 3.11 Biến thể Gated Recurrent Unit

3.3 Thực nghiệm mô hình LSTM

Nhóm sử dụng bộ dữ liệu “Superstore_Data” trên Kaggle để đưa ra dự đoán doanh số bán hàng. Đây là bộ dữ liệu về bán lẻ của một cửa hàng tại Mỹ, với hơn 51.000 bản ghi và 24 cột dữ liệu. Bộ dữ liệu này chứa các thông tin về sản phẩm, khách hàng, đơn đặt hàng và doanh thu của cửa hàng trong giai đoạn từ năm 2011 đến 2015.

Bộ dữ liệu Super Store Data [\[10\]](#)

Tiến hành load file dữ liệu và đọc một số thông tin cơ bản

```
import pandas as pd
with open('superstore_dataset2011-2015.csv', 'r', encoding='ISO-8859-1') as f:
    df = pd.read_csv(f)
df.head(5)
df.tail(5)
df.info()
df.describe()
```

...

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	City	State	...	Product ID	Category	Sub-Category	Product Name	Sales	Quantity	Discount	Profit	Shipping Cost	Order Priority
0	42433	AG-2011-2040	1/1/2011	6/1/2011	Standard Class	TB-11280	Toby Braunhardt	Consumer	Constantine	Constantine	...	OFF-TEN-10000025	Office Supplies	Storage	Tenex Lockers, Blue	408.300	2	0.0	106.140	35.46	Medium
1	22253	IN-2011-47883	1/1/2011	8/1/2011	Standard Class	JH-15985	Joseph Holt	Consumer	Wagga Wagga	New South Wales	...	OFF-SU-10000618	Office Supplies	Supplies	Acme Trimmer, High Speed	120.366	3	0.1	36.036	9.72	Medium
2	48883	HU-2011-1220	1/1/2011	5/1/2011	Second Class	AT-735	Annie Thurman	Consumer	Budapest	Budapest	...	OFF-TEN-10001585	Office Supplies	Storage	Tenex Box, Single Width	66.120	4	0.0	29.640	8.17	High
3	11731	IT-2011-3647632	1/1/2011	5/1/2011	Second Class	EM-14140	Eugene Moren	Home Office	Stockholm	Stockholm	...	OFF-PA-10001492	Office Supplies	Paper	Enemmax Note Cards, Premium	44.865	3	0.5	-26.055	4.82	High
4	22255	IN-2011-47883	1/1/2011	8/1/2011	Standard Class	JH-15985	Joseph Holt	Consumer	Wagga Wagga	New South Wales	...	FUR-FU-10003447	Furniture	Furnishings	Eldon Light Bulb, Duo Pack	113.670	5	0.1	37.770	4.70	Medium

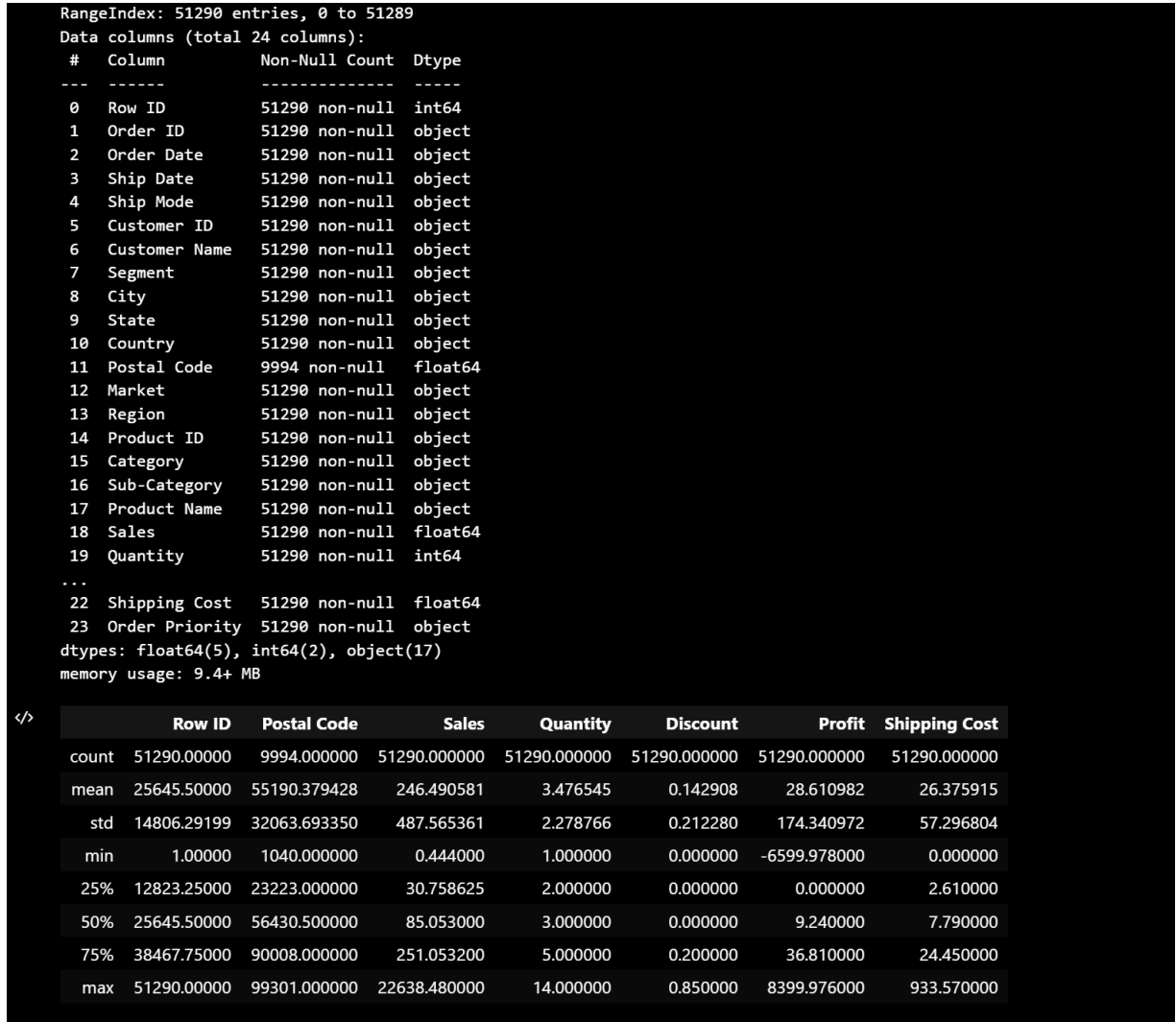
5 rows x 24 columns

Hình 3.12 Năm hàng đầu của bộ dữ liệu

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	City	State	...	Product ID	Category	Sub-Category	Product Name	Sales	Quantity	Discount	Profit	Shipping Cost	Order Priority
51285	32593	CA-2014-115427	31-12-2014	4/1/2015	Standard Class	EB-13975	Erica Bern	Corporate	Fairfield	California	...	OFF-BI-10002103	Office Supplies	Binders	Cardinal Slant-D Ring Binder, Heavy Gauge Vinyl	13.904	2	0.2	4.5188	0.89	Medium
51286	47594	MO-2014-2560	31-12-2014	5/1/2015	Standard Class	LP-7095	Liz Preis	Consumer	Agadir	Souss-Massa-Draâ	...	OFF-WIL-10001069	Office Supplies	Binders	Wilson Jones Hole Reinforcements, Clear	3.990	1	0.0	0.4200	0.49	Medium
51287	8857	MX-2014-110527	31-12-2014	2/1/2015	Second Class	CM-12190	Charlotte Melton	Consumer	Managua	Managua	...	OFF-LA-10004182	Office Supplies	Labels	Hon Color Coded Labels, 5000 Label Set	26.400	3	0.0	12.3600	0.35	Medium
51288	6852	MX-2014-114783	31-12-2014	6/1/2015	Standard Class	TD-20995	Tamara Dahlen	Consumer	Juárez	Chihuahua	...	OFF-LA-10000413	Office Supplies	Labels	Hon Legal Exhibit Labels, Alphabetical	7.120	1	0.0	0.5600	0.20	Medium
51289	36388	CA-2014-156720	31-12-2014	4/1/2015	Standard Class	JM-15580	Jill Matthias	Consumer	Loveland	Colorado	...	OFF-FA-10003472	Office Supplies	Fasteners	Bagged Rubber Bands	3.024	3	0.2	-0.6048	0.17	Medium

5 rows x 24 columns

Hình 3.13 Năm hàng cuối của bộ dữ liệu



Hình 3.14 Thông tin và mô tả tổng quan về các giá trị thống kê của các cột dữ liệu

3.3.1 Tiến hành tiền xử lý dữ liệu

```
# Kiểm tra dữ liệu bị thiếu
print(df.isnull().sum())
```

```
# Xóa các dòng có giá trị bị thiếu
df = df.dropna()
print(df.isnull().sum())
```

Row ID	0	Row ID	0
Order ID	0	Order Date	0
Order Date	0	Ship Date	0
Ship Date	0	Ship Mode	0
Ship Mode	0	Segment	0
Customer ID	0	City	0
Customer Name	0	State	0
Segment	0	Country	0
City	0	Postal Code	0
State	0	Market	0
Country	0	Region	0
Postal Code	41296	Product ID	0
Market	0	Category	0
Region	0	Sub-Category	0
Product ID	0	Product Name	0
Category	0	Sales	0
Sub-Category	0	Quantity	0
Product Name	0	Discount	0
Sales	0	Profit	0
Quantity	0	Shipping Cost	0
Discount	0	Order Priority	0
Profit	0	Order Year	0
Shipping Cost	0	Order Month	0
Order Priority	0	Order Day	0
dtype: int64		Ship Year	0
		Ship Month	0
		Ship Day	0
		dtype: int64	

Hình 3.15 Trước và sau khi xử lý cột bị thiếu dữ liệu

Xóa các cột không cần thiết

```
df = df.drop(['Order ID', 'Customer ID', 'Customer Name'],
axis=1)
```

Chuyển đổi kiểu dữ liệu của cột Order Date và Ship Date sang datetime

```
df['Order Date'] = pd.to_datetime(df['Order Date'])
df['Ship Date'] = pd.to_datetime(df['Ship Date'])
```

Tạo cột Year, Month, Day từ cột Order Date và Ship Date

```
df['Order Year'] = df['Order Date'].dt.year
df['Order Month'] = df['Order Date'].dt.month
df['Order Day'] = df['Order Date'].dt.day
df['Ship Year'] = df['Ship Date'].dt.year
df['Ship Month'] = df['Ship Date'].dt.month
df['Ship Day'] = df['Ship Date'].dt.day
```

Sử dụng phương pháp IQR (interquartile range) để xử lý các giá trị ngoại lai vì đây là các giá trị dữ liệu cực đoan, có giá trị khác biệt quá lớn so với các giá trị khác trong bộ dữ liệu. Các giá trị ngoại lai có thể gây ảnh hưởng đến việc phân tích và xử lý

dữ liệu, gây sai lệch trong các phân tích thống kê và mô hình hóa. Việc xử lý giá trị ngoại lai có thể cải thiện độ chính xác của phân tích và mô hình hóa dữ liệu.

```
# test và xử lý các giá trị ngoại lai (outliers)
# Cách thường được sử dụng là sử dụng IQR (interquartile range)
Q1 = df['Sales'].quantile(0.25)
Q3 = df['Sales'].quantile(0.75)
IQR = Q3 - Q1
df = df[(df['Sales'] >= Q1 - 1.5 * IQR) & (df['Sales'] <= Q3 + 1.5 * IQR)]
```

Chia bộ dữ liệu thành tập train (80%) và tập test (20%)

```
train_size = int(len(df) * 0.8)
train_data, test_data = df[:train_size], df[train_size:]
```

Chuẩn hóa giá trị của cột "Sales" trong tập dữ liệu huấn luyện và tập dữ liệu kiểm tra. Trong quá trình chuẩn hóa, giá trị tối thiểu và giá trị tối đa của cột "Sales" được xác định và các giá trị khác được chuyển đổi thành một khoảng giá trị mới từ 0 đến 1.

```
from sklearn.preprocessing import MinMaxScaler
# Chuẩn hóa dữ liệu cho tập train
scaler = MinMaxScaler(feature_range=(0, 1))
train_data_scaled =
scaler.fit_transform(train_data['Sales'].values.reshape(-1, 1))

# Chuẩn hóa dữ liệu cho tập test
test_data_scaled =
scaler.transform(test_data['Sales'].values.reshape(-1, 1))
```

3.3.2 Xây dựng mô hình LSTM

Viết hàm khởi tạo dữ liệu cho mô hình

```
def create_dataset(dataset, time_steps=1):
    X, y = [], []
    for i in range(len(dataset) - time_steps):
        a = dataset[i:(i + time_steps), 0]
        X.append(a)
```

```
y.append(dataset[i + time_steps, 0])
return np.array(X), np.array(y).reshape(-1, 1)
```

Chọn số time steps (kích thước cửa sổ trượt) và tạo dữ liệu cho mô hình

```
time_steps = 40
X_train, y_train = create_dataset(train_data_scaled,
time_steps)
X_test, y_test = create_dataset(test_data_scaled, time_steps)
```

Reshape dữ liệu để phù hợp với mô hình LSTM

```
X_train = np.reshape(X_train, (X_train.shape[0],
X_train.shape[1], 1))
X_test = np.reshape(X_test, (X_test.shape[0], X_test.shape[1],
1))
```

Khởi tạo mô hình LSTM

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM, Dropout
from tensorflow.keras.callbacks import EarlyStopping

# Khởi tạo mô hình LSTM
model = Sequential()

# Thêm lớp LSTM với 100 đơn vị đầu ra và kích thước đầu vào là
(time_steps, 1)
model.add(LSTM(units=100, input_shape=(time_steps, 1)))
model.add(Dropout(0.2)) # Thêm lớp Dropout

# Thêm lớp đầy đủ với 1 đơn vị đầu ra
model.add(Dense(1))

# Biên dịch mô hình với hàm mất mát là mean squared error và
thuật toán tối ưu hóa là adam
model.compile(loss='mean_squared_error', optimizer='adam')

# Sử dụng early stopping để tránh overfitting
```

```
early_stopping = EarlyStopping(monitor='val_loss', patience=10,
                                verbose=1, mode='auto')

# train mô hình
history = model.fit(X_train, y_train, epochs=100,
                    batch_size=64, verbose=1, shuffle=False,
                    validation_data=(X_test, y_test), callbacks=[early_stopping])
```

Đánh giá mô hình trên tập test sau đó vẽ biểu đồ so sánh dữ liệu thực tế và dữ liệu dự đoán của mô hình LSTM

```
import matplotlib.pyplot as plt

# Đánh giá mô hình trên tập test
test_loss = model.evaluate(X_test, y_test, verbose=0)
print(test_loss)

# Dự đoán trên tập test
lstm_y_pred = model.predict(X_test)

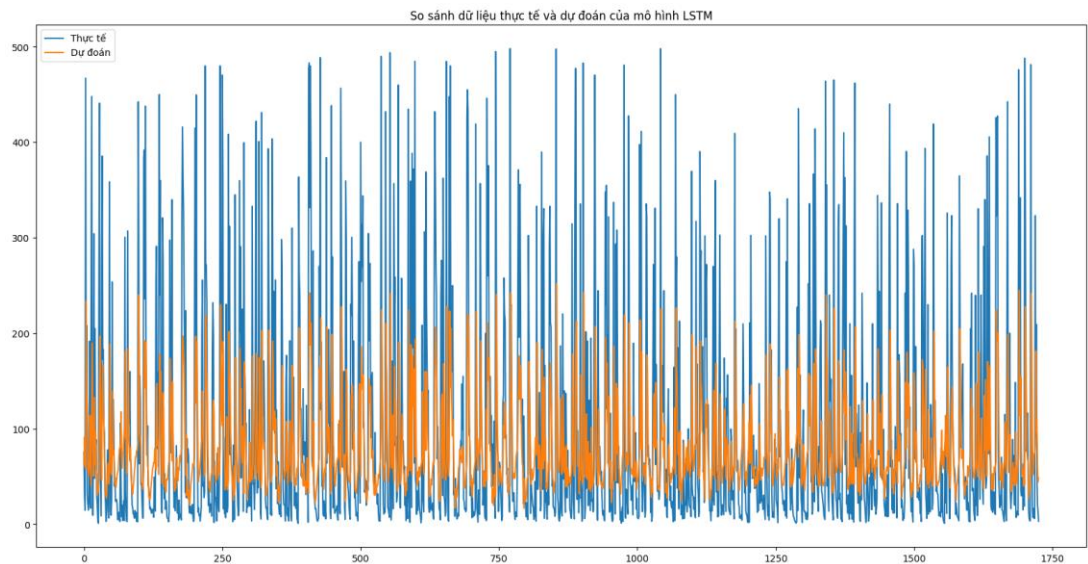
# Đưa kết quả dự đoán về đơn vị gốc và giải chuẩn hóa
y_pred_rescaled = scaler.inverse_transform(lstm_y_pred)
y_test_rescaled = scaler.inverse_transform(y_test.reshape(-1,
1))

# Tạo biểu đồ với dữ liệu thực tế và dữ liệu dự đoán của mô
hình
plt.figure(figsize=(20, 10))
plt.plot(y_test_rescaled, label='Thực tế')
plt.plot(y_pred_rescaled, label='Dự đoán')
plt.title('So sánh dữ liệu thực tế và dự đoán của mô hình
LSTM')
plt.legend()
plt.show()
```

```
0.050433430820703506
```

```
54/54 [=====] - 1s 12ms/step
```

Hình 3.16 Kết quả điểm đánh giá mô hình trên tập test



Hình 3.17 Biểu đồ so sánh dữ liệu thực tế và dữ liệu dự đoán của mô hình

3.3.3 So sánh với *LinearRegression* - *MLPRegressor* - *DecisionTreeRegressor*

Đầu tiên cần phải chuyển đổi kích thước dữ liệu đầu vào từ 3 chiều sang 2 chiều do dữ liệu đầu vào của mô hình LSTM là một mảng ba chiều với các chiều lần lượt là số lượng mẫu, số bước thời gian và số lượng đặc trưng còn đối với các mô hình học máy truyền thống thì đầu vào chỉ là một ma trận 2 chiều.

```
X_train = X_train.reshape(X_train.shape[0], -1)
y_train = y_train.reshape(-1)
X_test = X_test.reshape(X_test.shape[0], -1)
y_test = y_test.reshape(-1)
```

Tiến hành train các mô hình

```
lr_model = LinearRegression()
lr_model.fit(X_train, y_train)
lr_y_pred = lr_model.predict(X_test)

mlp_model = MLPRegressor(hidden_layer_sizes=(100,),
max_iter=1000, random_state=42)
mlp_model.fit(X_train, y_train)
```

```
mlp_y_pred = mlp_model.predict(X_test)

dt_model = DecisionTreeRegressor()
dt_model.fit(X_train, y_train)
dt_y_pred = dt_model.predict(X_test)
```

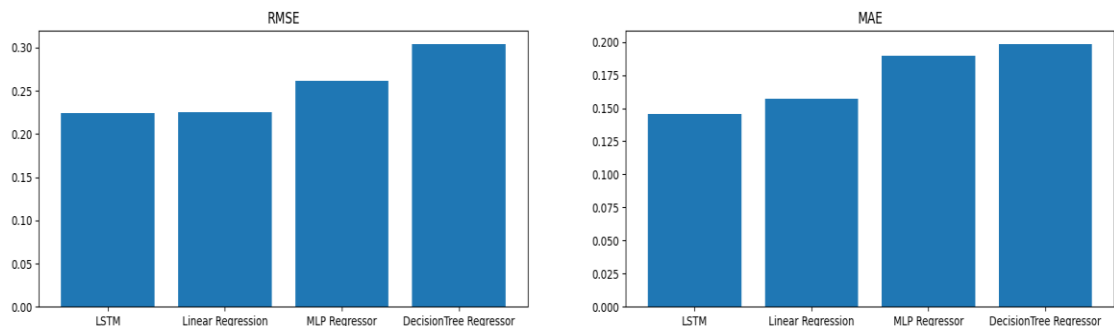
Đánh giá hiệu suất của các mô hình

```
lr_rmse = np.sqrt(mean_squared_error(y_test, lr_y_pred))
mlp_rmse = np.sqrt(mean_squared_error(y_test, mlp_y_pred))
lstm_rmse = np.sqrt(mean_squared_error(y_test, lstm_y_pred))
dt_rmse = np.sqrt(mean_squared_error(y_test, dt_y_pred))

lr_mae = mean_absolute_error(y_test, lr_y_pred)
mlp_mae = mean_absolute_error(y_test, mlp_y_pred)
dt_mae = mean_absolute_error(y_test, dt_y_pred)
lstm_mae = mean_absolute_error(y_test, lstm_y_pred)
```

```
Linear Regression - RMSE: 0.2254, MAE: 0.1575
MLP Regressor - RMSE: 0.2619, MAE: 0.1894
DecisionTree Regressor - RMSE: 0.3039, MAE: 0.1984
LSTM - RMSE: 0.2246, MAE: 0.1457
```

Hình 3.18 Kết quả đánh giá hiệu suất thông qua chỉ số RMSE và MAE



Hình 3.19 Biểu đồ đánh giá hiệu suất các mô hình

Dựa trên các giá trị RMSE và MAE của các mô hình, ta có thể đưa ra các kết luận sau:

- Mô hình LSTM cho kết quả tốt nhất với RMSE: 0.2246 và MAE: 0.1457.
- Mô hình Linear Regression cho kết quả tốt thứ hai với RMSE: 0.2254 và MAE: 0.1575.
- Mô hình MLP Regressor cho kết quả tốt thứ ba với RMSE: 0.2619 và MAE: 0.1894.
- Mô hình DecisionTree Regressor cho kết quả tốt thấp nhất với RMSE: 0.3039 và MAE: 0.1984.

Từ đó, ta có thể kết luận rằng mô hình LSTM cho kết quả tốt nhất trong việc dự đoán doanh số bán hàng so với các mô hình học máy truyền thống khác. Tuy nhiên, việc đưa ra kết luận này còn phụ thuộc vào các thông số khác như số lượng features, số lượng điểm dữ liệu, cấu trúc của mô hình, thời gian huấn luyện và các thông số khác của các mô hình.

TÀI LIỆU THAM KHẢO

Tiếng Việt

- [1] Vu, T. (2017c, March 4). Bài 15: Overfitting. Tiep Vu's Blog. <https://machinelearningcoban.com/2017/03/04/overfitting/>
- [2] Trức, T. T. (2022, November 2). Các phương pháp tránh Overfitting - Regularization, Dropout. Viblo. <https://viblo.asia/p/cac-phuong-phap-tranh-overfitting-gDVK24AmlLj>
- [3] Hai D. M. (2017, December 25). [ML] Mô hình quá khớp (Overfitting). Hai's Blog. <https://dominhhai.github.io/vi/2017/12/ml-overfitting/>
- [4] Nttuan. (2019). Bài 14: Long short term memory (LSTM). Deep Learning Cơ Bản. <https://nttuan8.com/bai-14-long-short-term-memory-lstm/>

Tiếng Anh

- [5] Seth, N. (2022, July 12). What Is Regularization in Machine Learning? Techniques & Methods. Blogs & Updates on Data Science, Business Analytics, AI Machine Learning. <https://www.analytixlabs.co.in/blog/regularization-in-machine-learning/>
- [6] GeeksforGeeks. (2023). Deep Learning Introduction to Long Short Term Memory. GeeksforGeeks. <https://www.geeksforgeeks.org/deep-learning-introduction-to-long-short-term-memory/>
- [7] Saxena, S. (2023). Learn About Long Short-Term Memory (LSTM) Algorithms. Analytics Vidhya. <https://www.analyticsvidhya.com/blog/2021/03/introduction-to-long-short-term-memory-lstm/>

- [8] Welcome to Surprise' documentation! — Surprise 1 documentation. (n.d.).
<https://surprise.readthedocs.io/en/stable/>
- [9] Giới thiệu. (n.d.). Google Developers.
<https://developers.google.com/machine-learning/recommendation?hl=vi>
- [10] superstore_data. (2019, January 30). Kaggle.
<https://www.kaggle.com/datasets/jr2ngb/superstore-data>
- [11] Amazon Fine Food Reviews. (2017, May 1). Kaggle.
<https://www.kaggle.com/datasets/snap/amazon-fine-food-reviews>