UNIVERSITY OF SCIENCE
FALCUTY OF INFORMATION TECHNOLOGY

SUBJECT: INTRODUCTION TO MACHINE LEARNING

TIME SERIES FORECASTING USING RECURRENT NEURAL NETWORK

CLASS 20KHMT2

Group ID: 20127005_20127013_20127370_20127610

Lecturers: **NGUYỄN NGỌC ĐỨC, NGUYỄN BẢO LONG, KIỀU VŨ MINH ĐỨC**

Mục lục

Thông tin nhóm:

STT	Họ và tên	MSSV	Email
1	Nguyễn Đức Bảo	20127005	ndbao202@clc.fitus.edu.vn
2	Đặng Nguyễn Duy	20127013	20127013@student.hcmus.edu.vn
3	Mai Quý Trung	20127370	mqtrung20@clc.fitus.edu.vn
4	Trương Samuel	20127610	20127610@student.hcmus.edu.vn

I. Papers & Publications

1. Nguồn gốc

- Tựa đề: Time Series Forecasting (TSF) Using Various Deep Learning Models
- Tác giả: Jimeng Shi, Mahek Jain, Giri Narasimhan - Florida International University
- Ngày đăng tải: 23/04/2022
- Nhà xuất bản: Semantic Scholar
- Mã bài báo: eprint arXiv:2204.11115
- Bipcode: 2022arXiv220411115S
- DOI: 10.48550/ arXiv:2204.11115
- Corpus ID: 248377041
- Keywords: Computer Science - Machine Learning, Time Series Forecasting
- Link bài báo: Time-Series-Forecasting-(TSF)-Using-Various-Deep-Shi-Jain

2. Chi tiết bài báo

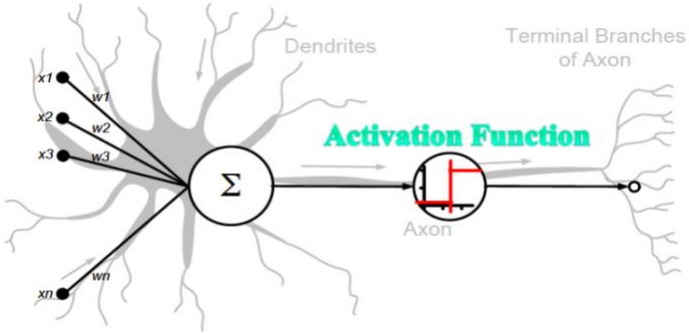
- Dự đoán chất lượng không khí ở Bắc Kinh thông qua dữ liệu chuỗi thời gian về tình trạng môi trường với 7 đặc trưng khác nhau.
- Bộ dữ liệu (hàng giờ) được sử dụng là Bộ dữ liệu chất lượng không khí Bắc Kinh từ trang web UCI, bao gồm một chuỗi thời gian đa biến gồm nhiều yếu tố được đo trên cơ sở hàng giờ trong khoảng thời gian 5 năm (2010-2014).
- Bài báo đã áp dụng qua 4 thuật toán để dự đoán và so sánh kết quả với nhau để tìm ra giải thuật tối ưu nhất.
- Trong đó có 3 thuật toán quan trọng trong DNN: Recurrent Neural Network và 2 biến thể của nó LSTM và GRU.
- Bài báo nghiên cứu hiệu suất của các mô hình dự đoán thay đổi thông qua các phương pháp sliding window có kích thước cửa sổ trượt khác nhau và các mốc thời gian khác nhau để dự đoán trong tương lai

II. Introduction to Neural Network

1. Tế bào thần kinh trong não

Bộ não con người có khoảng 100 tỷ tế bào thần kinh và hàng nghìn tỷ kết nối khớp thần kinh, do đó cung cấp cho nó khả năng lưu trữ khổng lồ.

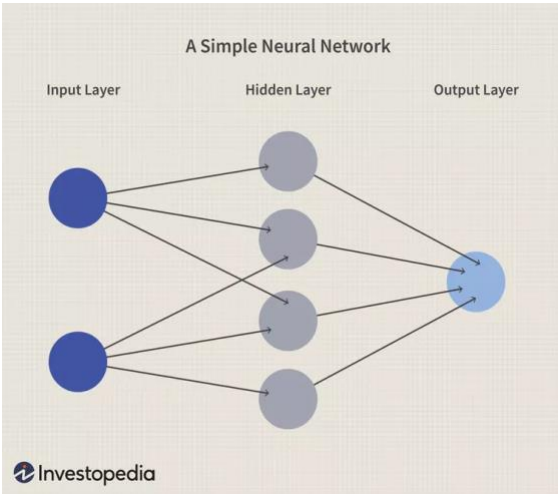
+ Tế bào thần kinh (neurons) nhận tín hiệu điện ở đuôi gai và gửi chúng đến sợi trục (axon).
+ Sợi nhánh (dendrites) có thể thực hiện các tính toán phi tuyến tính phức tạp.
+ Các khớp thần kinh (Synapses) không phải là một trọng lượng đơn lẻ mà là một hệ thống động lực học phi tuyến tính phức tạp.



Hình II.1.1: Sơ đồ minh họa mạng thần kinh trong não

2. Neural Network (Mạng Neuron nhân tạo)

Lấy cảm hứng từ bộ não con người, Neural Network là một phương pháp trong trí tuệ nhân tạo dạy máy tính xử lý dữ liệu. Đó là một loại quy trình máy học, được gọi là học sâu, sử dụng các nút hoặc tế bào thần kinh được kết nối với nhau trong một cấu trúc phân lớp giống như bộ não con người. Nó tạo ra một hệ thống thích ứng mà máy tính sử dụng để học hỏi từ những sai lầm của chúng và cải thiện liên tục. Do đó, mạng lưới thần kinh nhân tạo cố gắng giải quyết các vấn đề phức tạp, như tóm tắt tài liệu hoặc nhận dạng khuôn mặt, với độ chính xác cao hơn.



Hình II.2.1: Cấu trúc của Neural Network

a. Layer (các lớp)

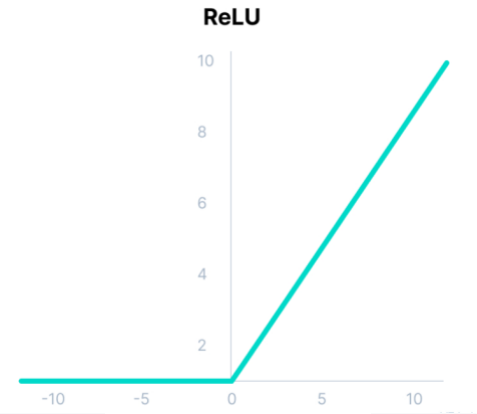
- Input Layer (lớp đầu vào): Thông tin từ thế giới bên ngoài đi vào mạng thần kinh nhân tạo từ lớp đầu vào. Các nút đầu vào xử lý dữ liệu, phân tích hoặc phân loại dữ liệu và chuyển dữ liệu đó sang lớp tiếp theo. Không có tính toán được thực hiện ở lớp này. Các nút ở đây chỉ truyền thông tin (tính năng) cho lớp ẩn.
- Hidden Layer (lớp ẩn): Các lớp ẩn lấy đầu vào của chúng từ lớp đầu vào hoặc các lớp ẩn khác. Mạng thần kinh nhân tạo có thể có một số lượng lớn các lớp ẩn. Mỗi lớp ẩn phân tích đầu ra từ lớp trước đó, xử lý thêm và chuyển nó sang lớp tiếp theo.
- Output Layer (lớp đầu ra): Lớp đầu ra đưa ra kết quả cuối cùng của tất cả quá trình xử lý dữ liệu của mạng thần kinh nhân tạo. Nó có thể có một hoặc nhiều nút. Chẳng hạn, nếu chúng ta gặp vấn đề về phân loại nhị phân (có/không), lớp đầu ra sẽ có một nút đầu ra, nút này sẽ cho kết quả là 1 hoặc 0. Tuy nhiên, nếu chúng ta gặp vấn đề về phân loại nhiều lớp, thì tầng đầu ra có thể bao gồm nhiều hơn một nút đầu ra.

b. Activation function (hàm kích hoạt)

- Quyết định liệu một nơron có nên được kích hoạt hay không. Điều này có nghĩa là nó sẽ quyết định xem đầu vào của nơron vào mạng có quan trọng hay không trong quá trình dự đoán bằng các phép toán đơn giản hơn. Chức năng của hàm kích hoạt là thêm tính phi tuyến tính vào mạng thần kinh.
- Có 3 loại hàm kích hoạt: Binary Step Function, Linear Activation Function và Non-Linear Activation Functions.
- Binary step function phụ thuộc vào giá trị ngưỡng quyết định liệu một nơron có nên được kích hoạt hay không.
- Linear Activation Function kích hoạt tỷ lệ thuận với đầu vào. Hàm không làm bất cứ điều gì với tổng trọng số của đầu vào, nó chỉ đưa ra giá trị mà nó được đưa ra.
- Non-Linear Activation Functions giải quyết các hạn chế của Linear Activation Function (không cho phép mô hình tạo các ánh xạ phức tạp giữa đầu vào và đầu ra của mạng) cho phép back propagation (lan truyền ngược) vì bây giờ các hàm đạo hàm sẽ liên quan đến đầu vào, và nó có thể quay trở lại và hiểu được trọng số nào của nơron đầu vào đưa ra dự đoán tốt hơn.
- Sigmoid Activation Function: Hàm này lấy bất kỳ giá trị thực nào làm giá trị đầu vào và đầu ra trong khoảng từ 0 đến 1. Đầu vào càng lớn càng gần 1, trong khi càng nhỏ thì càng gần 0. Nó thường được sử dụng cho các mô hình mà chúng ta phải dự đoán xác suất như một đầu ra. Vì xác suất của mọi thứ chỉ tồn tại trong phạm vi từ 0 đến 1.

$$f(x) = \frac{1}{1 + e^{-x}}$$

- ReLU Function: viết tắt của Rectified Linear Unit (Đơn vị tuyến tính chỉnh lưu). ReLU có một hàm đạo hàm và cho phép lan truyền ngược đồng thời làm cho nó hiệu quả về mặt tính toán. ReLU không kích hoạt tất cả các nơ-ron cùng một lúc. Các nơ-ron sẽ chỉ bị vô hiệu hóa nếu đầu ra của phép biến đổi tuyến tính nhỏ hơn 0. Do chỉ có một số lượng tế bào thần kinh nhất định được kích hoạt nên hàm ReLU hiệu quả hơn nhiều về mặt tính toán khi so sánh với hàm sigmoid.



$$f(x) = max\,(0, x)$$

c. Gradient descent

- Là một thuật toán tối ưu hóa, dữ liệu được đào tạo giúp các mô hình này học theo thời gian và hàm chi phí trong độ dốc giảm dần, đánh giá độ chính xác của nó với mỗi lần lặp cập nhật tham số. Cho đến khi hàm gần bằng hoặc bằng 0, mô hình sẽ tiếp tục điều chỉnh các tham số của nó để mang lại sai số nhỏ nhất có thể. Sau khi các mô hình máy học được tối ưu hóa về độ chính xác, chúng có thể trở thành công cụ mạnh mẽ cho Neural network.

d. Loss function

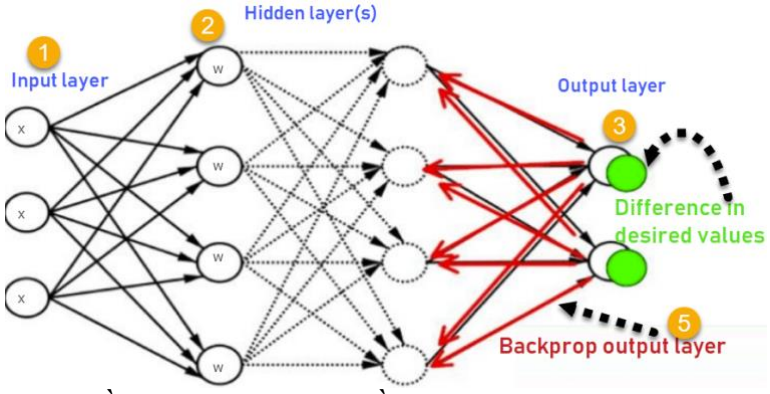
- Hàm mất mát là hàm tính toán khoảng cách giữa đầu ra hiện tại của thuật toán và đầu ra dự kiến. Đó là một phương pháp để đánh giá cách thuật toán của bạn mô hình hóa dữ liệu. Nó có thể được phân loại thành hai nhóm. Một để phân loại (các giá trị rời rạc, 0,1,2…) và một để hồi quy (các giá trị liên tục).
- Nói một cách đơn giản, Loss function được sử dụng để tính toán độ dốc. Và độ dốc được sử dụng trong Gradient descent để cập nhật trọng số của Mạng thần kinh. Đây là cách một Mạng lưới thần kinh được đào tạo.

e. Back propagation

- Backpropagation là bản chất của đào tạo mạng lưới thần kinh. Đó là phương pháp tinh chỉnh các trọng số của mạng thần kinh dựa trên tỷ lệ lỗi thu được trong kỷ nguyên trước đó (tức là phép lặp). Việc điều chỉnh đúng các trọng số cho phép bạn

giảm tỷ lệ lỗi và làm cho mô hình trở nên đáng tin cậy bằng cách tăng tính tổng quát của nó.

- Lan truyền ngược trong mạng thần kinh là một dạng viết tắt của "lan truyền ngược lỗi". Đó là một phương pháp tiêu chuẩn để đào tạo mạng lưới thần kinh nhân tạo. Phương pháp này giúp tính toán độ dốc của hàm mất mát đối với tất cả các trọng số trong mạng.
- Thuật toán lan truyền ngược trong mạng thần kinh tính toán độ dốc của hàm mất mát cho một trọng số theo quy tắc chuỗi. Nó tính toán hiệu quả từng lớp một, không giống như tính toán trực tiếp gốc. Nó tính toán độ dốc, nhưng nó không xác định cách sử dụng độ dốc. Nó khái quát hóa tính toán trong quy tắc delta.



Hình II.2.2: Sơ đồ minh họa lan truyền ngược trong Neural Network

1. Đầu vào X, đến qua đường dẫn được kết nối trước
2. Đầu vào được mô hình hóa bằng trọng số thực W. Các trọng số thường được chọn ngẫu nhiên.
3. Tính toán đầu ra cho mọi nơron từ lớp đầu vào, đến các lớp ẩn, đến lớp đầu ra.
4. Tính toán lỗi trong đầu ra với công thức: Lỗi = Đầu ra thực tế – Đầu ra mong muốn
5. Di chuyển trở lại từ lớp đầu ra đến lớp ẩn để điều chỉnh các trọng số sao cho giảm lỗi.

- Tiếp tục lặp lại quy trình cho đến khi đạt được đầu ra mong muốn

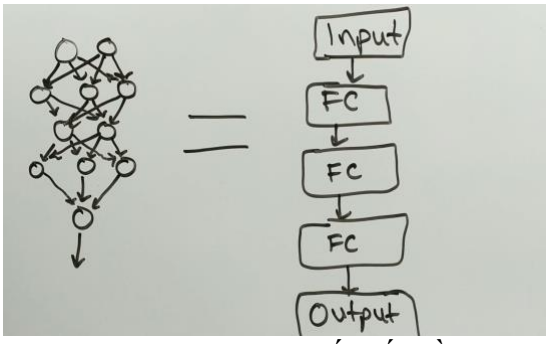
III. Deep Neural Network (DNN)

1. Định nghĩa

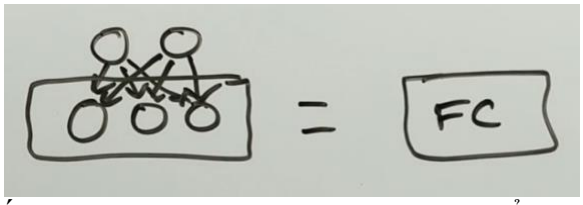
- Ở mức đơn giản nhất, một mạng nơ-ron với một số mức độ phức tạp, thường có ít nhất hai lớp, đủ điều kiện là mạng nơ-ron sâu (DNN) hay gọi tắt là mạng sâu.
- Một mạng phải có bao nhiêu lớp để đủ điều kiện là sâu? Không có câu trả lời chắc chắn cho điều này (hơi giống như hỏi có bao nhiêu hạt tạo thành một đống), nhưng thường có hai hoặc nhiều lớp ẩn được tính là sâu. Ngược lại, một mạng chỉ có một lớp ẩn duy nhất được gọi là "nông". Một cách không chính thức, "sâu" cho thấy rằng mạng rất khó xử lý.

2. Sự khác nhau giữa Artificial Neural Network (ANN) và DNN

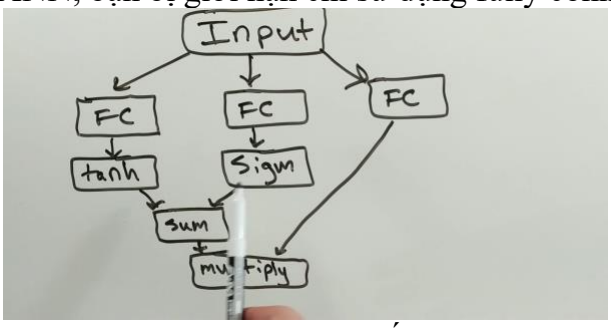
- Về mặt kỹ thuật, mạng thần kinh nhân tạo (ANN) có nhiều lớp được gọi là Mạng thần kinh sâu (DNN).



- Với ANN, fully connected layers (lớp được kết nối đầy đủ) chỉ là một lớp bình thường trong đó lớp được tạo thành từ các nơ-ron, mỗi nơron lấy tất cả các nơ-ron của lớp trước đó làm đầu vào.



- DNN không nhất thiết phải là ANN. Với DNN, bạn có thể sử dụng bất kỳ công thức và kỹ thuật nào bạn muốn để định hình từng lớp hoặc phần trong mạng của mình, nhưng trong ANN, bạn bị giới hạn chỉ sử dụng fully connected layers.



- DNN là một trong những công cụ hiệu quả nhất trong lĩnh vực học sâu. Đầu vào quy trình DNN thông tin theo cách phân cấp, trong đó mỗi cấp độ xử lý tiếp theo trích xuất trừu tượng hơn/toàn cầu/bất biến các tính năng và đặc điểm. Nói cách khác, DNN (bán) tự động tìm hiểu các tính năng chính từ dữ liệu và sau đó tổng hợp chúng cho một số mục đích, chẳng hạn như nhận dạng các đối tượng trong hình ảnh.

3. Các thuật toán DNN điển hình

Đây là 3 trong số những thuật toán DNN phổ biến nhất:

- Convolution Neural Network (CNN)
- Autoencoder
- Recurrent Neural Network (RNN)

a. Convolution Neural Network (CNN)

CNN được dựa theo vỏ não thị giác của con người và là một mạng lưới thần kinh giúp lựa chọn trong thị giác máy tính (nhận dạng hình ảnh) và nhận dạng video. CNN là 1 thuật toán học có giám sát.

b. Autoencoder

Autoencoder là mạng thần kinh sử dụng phương thức học không giám sát. Autoencoder học các biểu diễn của dữ liệu đầu vào và sẽ giảm chiều dữ liệu của nó rồi cuối cùng sẽ tái tạo lại tập dữ liệu gốc. Autoencoder được ứng dụng trong nén ảnh, giảm nhiễu ảnh, tìm kiếm hình ảnh.

c. Recurrent Neural Network (RNN)

Ứng dụng trong việc dự đoán Time Series. Chi tiết về RNN sẽ được nói chi tiết ở dưới.

IV. Time Series (Chuỗi thời gian)

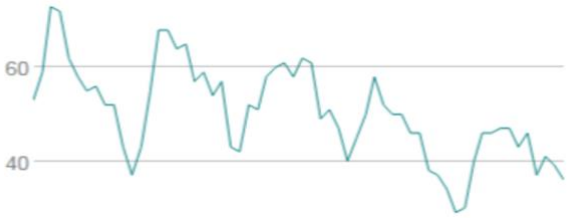
1. Khái niệm

Time Series là một chuỗi các quan sát lặp lại của một đại lượng nào đó trong khoảng thời gian nhất định. Ví dụ như giá cổ phiếu, lượng mưa, mật độ giao thông trong thông tin liên lạc hoặc giao thông vận tải.

2. Đặc trưng

a. Trend (Xu hướng)

Là đặc trưng thể hiện xu hướng tăng hoặc giảm giá trị của chuỗi theo thời gian.
Trend có thể mang tính cục bộ hoặc toàn cục, nhưng một Time Series có thể thể hiện cả 2



Time Series có trend giảm dần

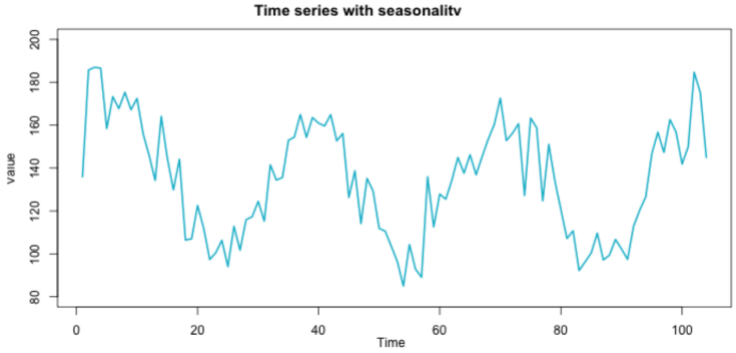
Time Series có trend tăng dần

Time Series không có trend

b. Seasonal (Tính mùa)

Là đặc trưng thể hiện các pattern có tính lặp lại (theo tháng, quý), đoán trước được trong Time Series.

Ví dụ: Lượng tiêu thụ điện sẽ tăng ở một số quốc gia vào mùa hè (máy lạnh) hoặc mùa đông (sưởi ấm)

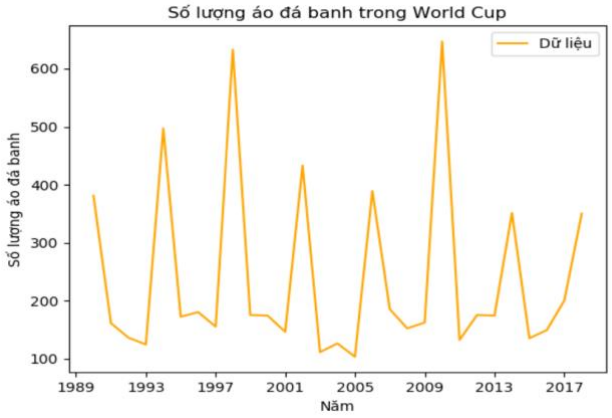


Time Series có tính mùa

c. Cyclical (Tính chu kì)

Tương tự như Seasonal nhưng khác ở chỗ là đặc trưng này có sự quan sát trong khoảng thời gian dài hơn (nhiều năm)

Ví dụ: Lượng áo đá banh bán ra tăng trong mùa World Cup

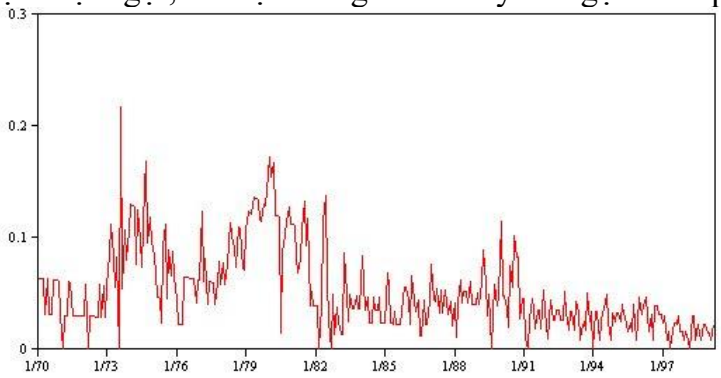


Time Series có tính chu kì

d. Pulse và Step

Trong chuỗi đôi khi sẽ có những thời điểm mà giá trị trung bình thay đổi đột ngột, chúng được chia làm 2 loại:

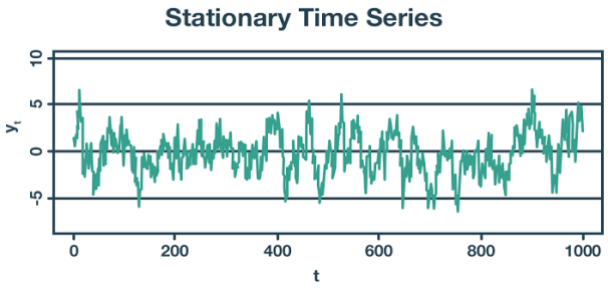
- Sự chuyển dịch đột ngột, tạm thời trong chuỗi hay còn gọi là Pulse
- Sự chuyển dịch đột ngột, dài hạn trong chuỗi hay còn gọi là Step



Một chuỗi có Pulse

e. Stationary (Tính tĩnh)

Là một chuỗi thời gian mà tính chất của nó không phụ thuộc vào thời điểm được quan sát (tức giá trị trung bình và phương sai luôn không đổi)



Một chuỗi tĩnh

3. Ứng dụng

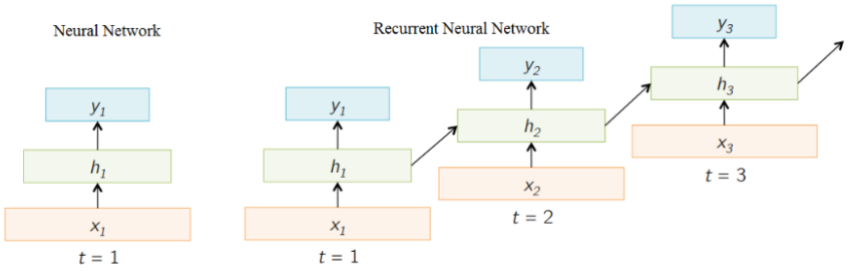
Vì bản chất Time Series là một chuỗi các quan sát lặp lại của tập hợp các đại lượng nào đó trong khoảng thời gian nhất định nên nó được ứng dụng vào việc dự đoán dựa trên những dữ liệu đã thu thập được. RNN là mạng thần kinh phù hợp nhất với kiểu dữ liệu chuỗi thế này nên chúng ta sẽ áp dụng RNN để dự đoán Time Series.

V. Recurrent Neural Network (RNN) - Algorithms & Variants

1. Khái niệm

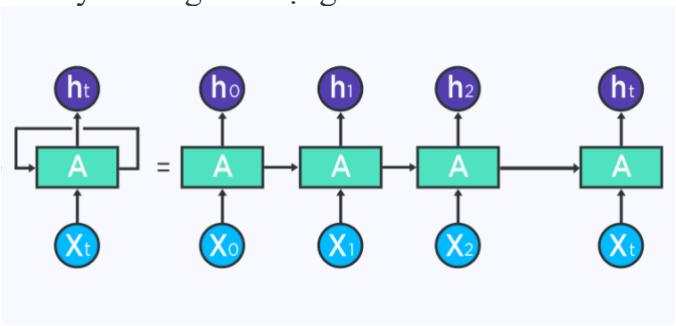
- RNN là 1 loại thuật toán trong Deep Neural Network, được phát triển từ Neural Network
- Được áp dụng cho Apple Siri và công cụ tìm kiếm bằng giọng nói của Google

- Một thuật toán ghi nhớ đầu vào của nó nhờ vào bộ nhớ trong
- Rất phù hợp với các bài toán học máy liên quan đến dữ liệu tuần tự, ứng dụng trong xử lý ngôn ngữ tự nhiên, nhận dạng giọng nói, nhận dạng giọng nói và đặc biệt là dự đoán chuỗi thời gian



Hình V.1.1: Sự khác nhau giữa Neural Network và RNN

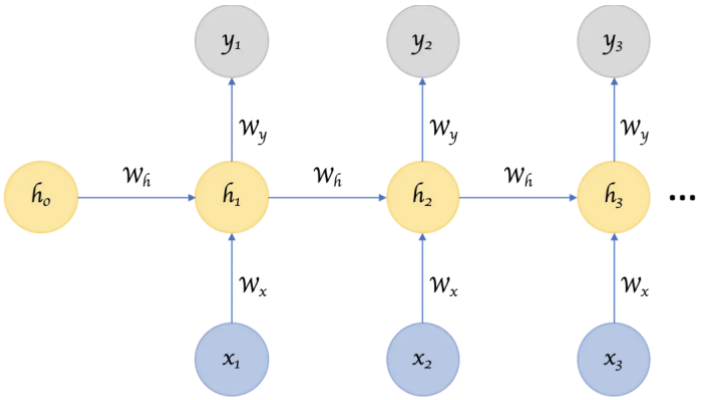
- Ở Neural Network thông thường, chỉ có sự liên kết giữa input layer – hidden layer và hidden layer – output layer. Còn với Recurrent Neural Network, giữa các node trong hidden layer đều có 1 mối liên kết thứ 3, tạo nên 1 chuỗi tuần tự cho input, output và hidden layer trong các trạng thái khác nhau.



Hình V.1.2: Trực quan mô hình Recurrent Neural Network

2. Nguyên lý thuật toán RNN

- Bản chất của thuật toán này là trạng thái phía sau sẽ phụ thuộc rất nhiều vào trạng thái trước đó. Hay nói cách khác, tại trạng thái t bất kì, nó sẽ phụ thuộc vào 2 yếu tố chính: input tại thời điểm t và trạng thái trước đó t – 1. Do đó, so với Neural Network, mô hình RNN sẽ cần thêm một mối liên kết nữa để lưu lại trạng thái phía trước. Thay vì input → hidden → ouput thì mô hình RNN là (input + prev_input) → hidden → ouput.
- Ngoài ra, RNN cũng sẽ điều chỉnh trọng số thông qua gradient descent bằng lan truyền ngược (back propagation) theo thời gian (trạng thái t, t+1, t+2, …).



Hình V.2.1: Nguyên lý mô hình Recurrent Neural Network

Công thức:

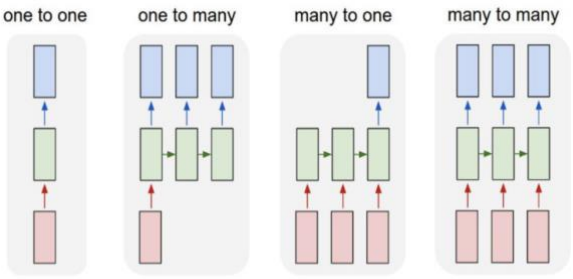
- **h(t) = f (Wx * x(t) + Wh * h(t-1) + bias)**
- **y(t) = g (Wy * h(t) + bias)**

Trong đó:

- x(t) và h(t): input đầu vào và trạng thái ẩn tại bước t.
- h(t) được tính dựa trên các trạng thái ẩn phía trước và đầu vào tại thời điểm đó.
- f: activation function, thường là một hàm non-linear như sigmoid (hoặc ReLU).
- y(t): đầu ra tại thời điểm t.
- g: hàm tùy vào yêu cầu bài toán mà áp dụng, điển hình là hàm sigmoid hoặc softmax, ...
- Thông thường để bắt đầu bước lan truyền tiến, người ta cho trạng thái ẩn đầu tiên được khởi tạo bằng 0.

Có 4 loại RNN phổ biến:

- Một-một: lan truyền tiến
- Một-nhiều: dự đoán từ kế tiếp
- Nhiều-nhiều: phiên dịch
- Nhiều-một: dự đoán chuỗi thời gian, phân loại giọng nói



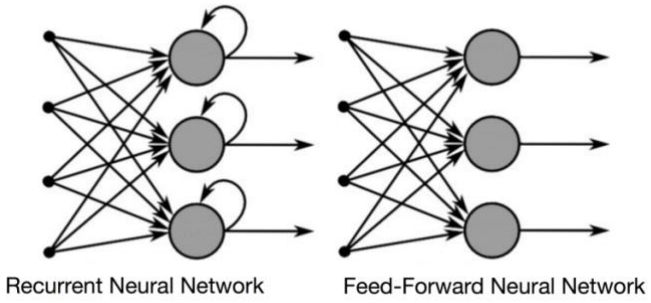
Hình V.2.2: Phân loại RNN

3. Ứng dụng

- Xử lý ngôn ngữ tự nhiên: dự đoán từ kế tiếp, gợi ý tìm kiếm
- Time Series: các bài toán dự đoán như thời tiết, giá vàng, chứng khoán, tiền ảo, …

4. So sánh Recurrent Neural Network với Feed-Forward Neural Network

Recurrent Neural Network	Feed-Forward Neural Network
Có bộ nhớ để lưu trữ đầu vào: (input + prev_input) -> hidden -> output	input -> hidden -> output
Đưa ra quyết định cho bước tiếp theo dựa trên đầu vào hiện tại và đầu vào trước đó	Không có bộ nhớ cho input nên output ở mỗi trạng thái mang tính độc lập



Hình V.4.1: Recurrent Neural Network và Feed-Forward Neural Network

5. Hạn chế của Recurrent Neural Network

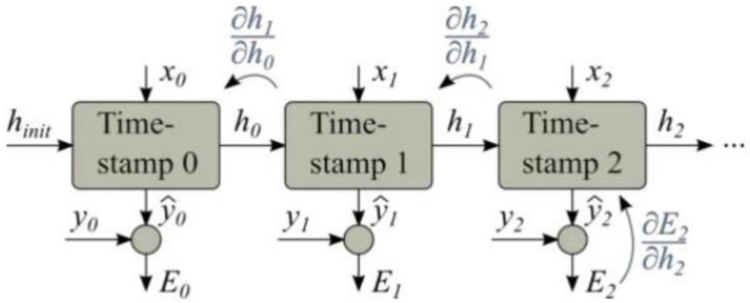
- Tốc độ thực thi: tiến trình làm việc tuần tự không song song.
- Thực hiện theo 1 chiều cho RNN nguyên bản, các phiên bản nâng cấp thường có cả hai chiều giữa các trạng thái.
- Vanishing gradient (mất mát đạo hàm): Với các bộ dữ liệu có nhiều trạng thái, RNN có xu hướng "quên" các đầu vào trước đó.

a. Hàm mất mát & lan truyền ngược

Ta cần chú ý đến 3 giá trị h, Wx và Wy để tính độ lỗi và cập nhật trọng số theo Gradient Descent:

- Wx phụ thuộc vào đầu vào x(t) và h(t).
- Wy phụ thuộc vào trạng thái h(t), output y(t) và hàm chuyển đổi f.

Vậy còn h(t) thì phụ thuộc vào đâu? Ta xét sự lan truyền ngược của từng trạng thái h



Hình V.4.1: Sơ đồ lan truyền ngược của trạng thái h trong RNN

- Dạo hàm độ lỗi ở thời điểm bất kỳ:

$$W \to W - \alpha \frac{\partial E}{\partial W} \; với \frac{\partial E}{\partial W} = \sum_{i=1}^{t} \frac{\partial E_i}{\partial W}$$

$$\frac{\partial E_i}{\partial W} = \sum_{k=1}^{i} \frac{\partial E_i}{\partial h_k} \frac{\partial h_k}{\partial W}$$

- Sử dụng quy tắc "chain rule" trong đạo hàm:

$$\frac{\partial E_i}{\partial W} = \sum_{k=1}^{i} \frac{\partial E_i}{\partial y_i} \frac{\partial y_i}{\partial h_i} \frac{\partial h_i}{\partial h_k} \frac{\partial h_k}{\partial W}$$

- Đạo hàm độ lỗi Wh tổng quát:

$$\frac{\partial E}{\partial W_R} = \sum_{j=0}^{t-1} \sum_{k=1}^{j} \frac{\partial E_j}{\partial y_j} \frac{\partial y_j}{\partial h_j} \left(\prod_{m=k+1}^{j} \frac{\partial h_m}{\partial h_{m-1}} \right) \frac{\partial h_k}{\partial W_R}$$

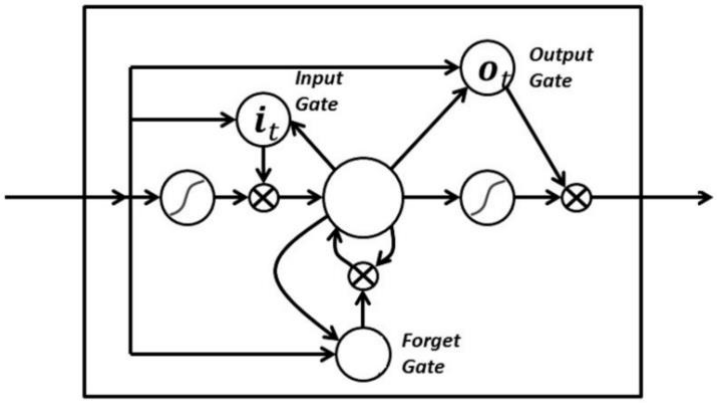
b. Vanishing Gradients (mất mát đạo hàm)

- Hiện tượng mất mát đạo hàm xảy ra khi các giá trị của độ dốc quá nhỏ và kết quả là mô hình ngừng học hoặc mất quá nhiều thời gian để học.
- Theo quy tắc lan truyền ngược, trạng thái h(t) phụ thuộc vào Wh ở t-1 và trạng thái h(t-1) mà trạng thái h(t-1) lại phụ thuộc vào Wh ở t-2 và h(t-2),... Cứ thế, ở trạng thái ht bất kì thì chúng sẽ phụ thuộc vào các kết quả trước đó và đệ quy cho đến khi về trạng thái đầu tiên.
- Nếu mạng RNN càng được mở rộng và những yếu tố này có giá trị nhỏ hơn 1 thì nó sẽ gây ra tình trạng mất mát đạo hàm do tích đạo hàm tiến về 0.
- Chính vì vậy, tình trạng vanishing gradient chắc chắn sẽ xảy ra nếu như bộ dữ liệu có đủ nhiều trạng thái, dẫn đến việc đạo hàm độ lỗi khi được tính bằng lan truyền ngược sẽ gần như bằng 0 và khi đó mô hình sẽ không còn cập nhật trọng số cho Wh, dẫn đến các trọng số giống nhau các trang thái sau đó.
- Chính vì vậy, RNN thuần túy không phù hợp cho các bộ dữ liệu TimeSeries có tính phụ thuộc xa.
- Từ đó, người ta đã phát triển lên 1 phiên bản mới cao cấp hơn RNN, được gọi là Long Short-Term Memory, viết tắt là LSTM.

6. Long Short-Term Memory Networks (LSTMs)

a. Khái niệm

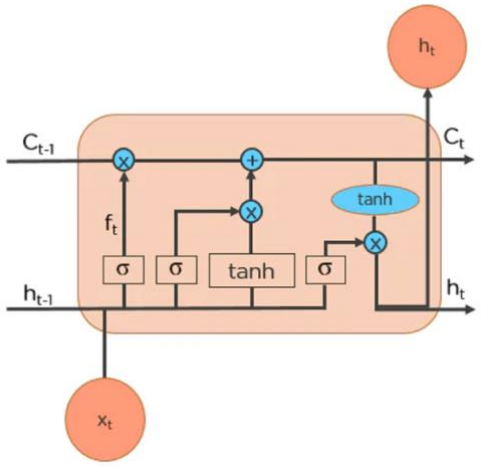
- LTSM là một phiên bản cập nhật của RNN, về cơ bản là mở rộng bộ nhớ, cho phép các RNN gán và ghi nhớ các input trong một khoảng thời gian dài. Điều này là do LSTM chứa thông tin trong bộ nhớ riêng. LSTM có thể đọc, ghi và xóa thông tin khỏi bộ nhớ của nó. Do đó, nó rất phù hợp với dữ liệu TimeSeries có tính phụ thuộc xa.
- Bộ nhớ này có thể được coi là 1 cell có cổng, cell sẽ quyết định lưu trữ hoặc xóa thông tin hay không (mở/đóng cổng), dựa vào tầm quan trọng của thông tin. Việc gán tầm quan trọng của thông tin cũng được học bởi thuật toán. Điều này có nghĩa là nó học theo thời gian thông tin nào quan trọng, thông tin nào không.



Hình V.6.1: Sơ đồ mô hình Long Short-Term Memory (LSTM)

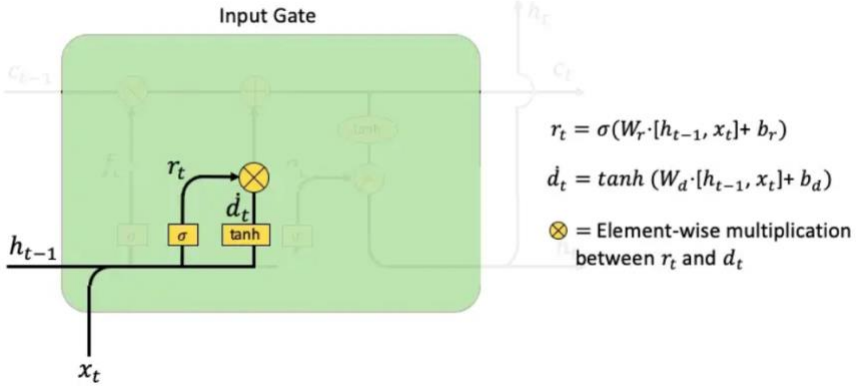
b. Nguyên lý

- Một ô nhớ của mô hình LSTM có ba cổng: cổng vào (input gate), cổng quên (forget gate) và cổng ra (output).
  + Input gate: cổng này xác định có cho phép đầu vào mới vào hay không.
  + Forget gate: xóa thông tin nếu nó không quan trọng.
  + Output gate: tác động đến đầu ra ở dấu thời gian hiện tại.



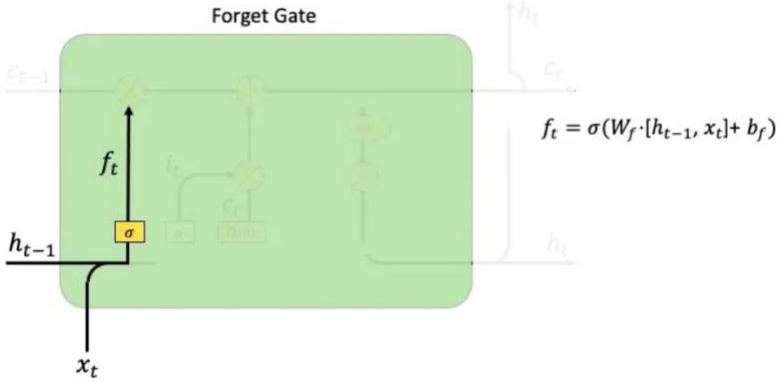
Hình V.6.2: Nguyên lý 3 cổng trong LSTM

- Input gate (cổng đầu vào): Ở cổng này sẽ có 2 tầng di chuyển. Tầng rt để quyết định bao nhiêu phần trăm chấp nhận giá trị đầu vào để cập nhật, sử dụng hàm sigmoid. Ngoài ra còn có 1 tầng dt dùng để tạo ra 1 vector mới nhằm để thêm vào cho trạng thái Ct tiếp theo, sử dùng hàm tanh.



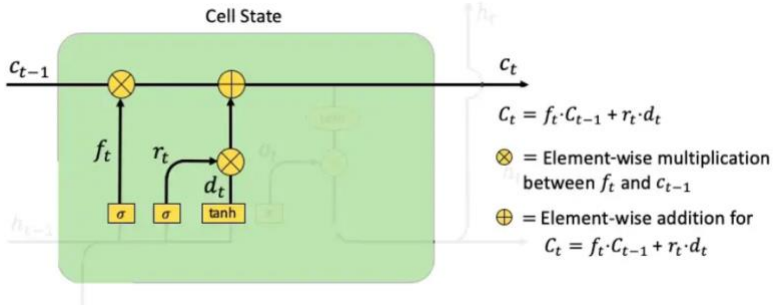
Hình V.6.3: Cổng đầu vào trong LSTM

- Forget gate (cổng quên): Cổng này sẽ quyết định xem thông tin nào cần được bỏ đi khỏi cell trung tâm, được tiến hành bởi hàm sigmoid, cho giá trị đầu ra từ 0 đến 1, tương ứng với phần trăm thông tin cần được quên.



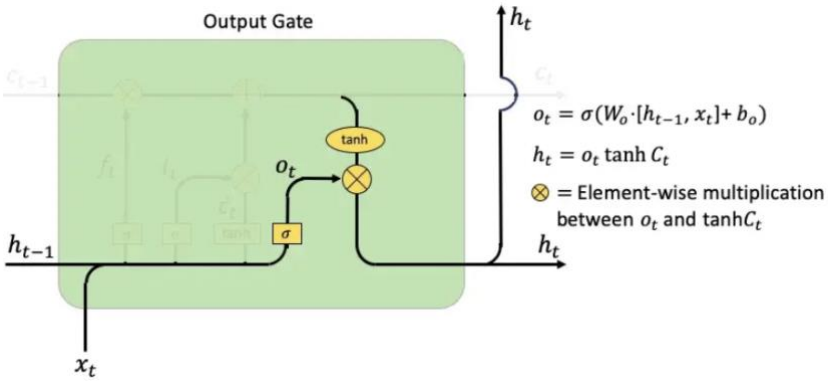
Hình V.6.4: Cổng quên trong LSTM

- Cell state (trạng thái bộ nhớ trung tâm): Ở bộ nhớ trung tâm, giá trị mới sẽ được cập nhật bằng sự kết hợp của 2 giá trị ở cổng quên và cổng đầu vào, đó là tích của trạng thái bộ nhớ trước đó Ct-1 và giá trị cổng quên cộng với tích của 2 tầng rt và dt của cổng đầu vào.



Hình V.6.5: Bộ nhớ trung tâm trong LSTM

- Output gate (cổng đầu ra): Cổng này sẽ quyết định bao nhiêu phần trăm giá trị được tác động và kết quả cuối cùng ht bằng cách sử dùng hàm sigmoid. Giá trị của Ct được đi qua 1 tầng nữa, sử dụng hàm tanh, sau đó nhân với giá trị output ở cổng đầu ra để ra được ht. Giá trị này có thể sử dụng làm kết quả output hoặc trạng thái cho node hidden tiếp theo.



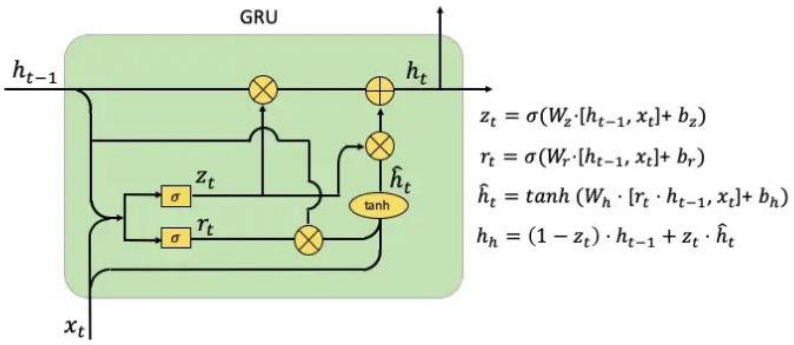
Hình V.6.6: Cổng đầu ra trong LSTM

Nhận xét:

- Các cổng trong LSTM được tính bằng sigmoid, nghĩa là giá trị chạy từ 0 đến 1, điều đó cho phép chúng thực hiện lan truyền ngược.
- Vanishing gradients được giải quyết thông qua LSTM vì nó giữ cho độ dốc đủ dốc, giúp quá trình train ngắn và độ chính xác cao hơn.

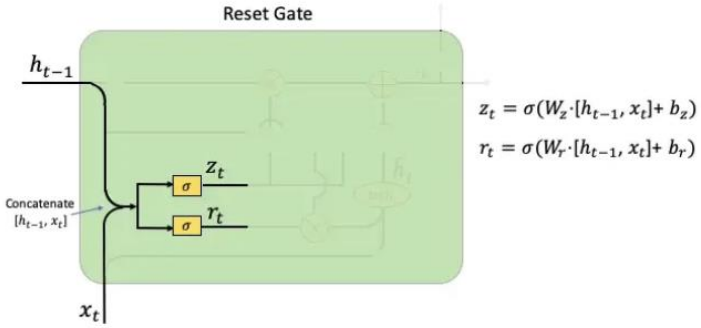
7. Gated Recurrent Units (GRU)

- GRU là 1 phiên bản khác cũng được nâng cấp từ RNN, gần tương tự với LSTM, chỉ có điều kiến trúc xây dựng của mô hình này chỉ gồm 1 cell và 2 cổng chính: reset gate và update gate.
- Chính vì có ít cổng hơn so với LSTM nên kiến trúc GRU phức tạp hơn nhiều so với LSTM.



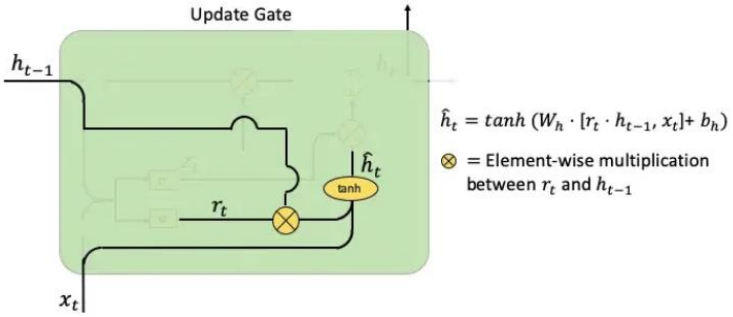
Hình V.7.1: Sơ đồ kiến trúc mô hình GRU

- Reset gate (cổng reset): Cổng này gồm 2 tầng zt và rt, đều được quyết định bởi hàm sigmoid cho giá trị đầu ra từ 0 đến 1.



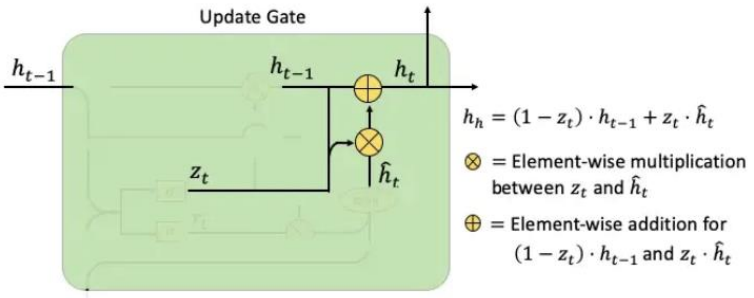
Hình V.7.2: Cổng reset trong mô hình GRU

- Update gate I (cổng update I): Cổng update sẽ bao gồm 2 giai đoạn hoạt động tuần tự. Ở cổng I, có 1 tầng h^t và 1 tầng rt nhỏ hơn, tầng rt này sẽ nhân với h(t-1), rồi nhân với trọng số, sau cùng được quyết định bởi hàm tanh ra kết quả cho h^t.



Hình V.7.3: Cổng update I trong mô hình GRU

- Update gate II (cổng update II): Cổng update II sẽ tính ra giá trị ht cuối cùng, được tính bởi sự kết hợp của h^t, zt của cổng reset và trạng thái trước đó h(t-1). Giá trị ht tùy tình huống sẽ được sử dụng làm output hoặc trạng thái tiếp theo trong hidden layer.



Hình V.7.4: Cổng update II trong mô hình GRU

VI. Time Series Forecasting with Recurrent Neural Network

1. Phương pháp luận

- Các mô hình RNN sử dụng dữ liệu trong lịch sử để tìm ra phụ thuộc hàm giữa các đặc trưng (đầu vào) và các giá trị tiềm năng của biến mục tiêu (đầu ra). Mô hình sau khi được huấn luyện có khả năng đưa ra dự đoán cho giá trị của biến mục tiêu tại các thời điểm trong tương lai.
- Cho dữ liệu chuỗi thời gian:

$$\left\{x_1, x_2, \ldots, x_T\right\}$$

Với $\boldsymbol{x_t}$ là vector của m đặc trưng đầu vào quan sát hoặc tính toán được tại thời điểm $\boldsymbol{t}$. Nhiệm vụ đặt ra là phát triển 1 mô hình để dự đoán giá trị đầu ra $\mathbf{yt_{+k}}$ tại thời điểm $t + k$ trong tương lai, sử dụng dữ liệu trong lịch sử, đồng nghĩa với việc tập dữ liệu chuỗi thời gian kết thúc tại t-1:

$$\left\{\ldots, x_{t-2}, x_{t-1}\right\}$$

- Để làm các vector đầu vào của mô hình có kích thước đồng đều, chúng tôi đã sử dụng cửa sổ thời gian trượt có chiều dài cố định có kích thước w. Dữ liệu được chuyển đổi bằng cách sử dụng tỷ lệ tối thiểu - tối đa (scale các giá trị về tập giá trị có min là 0 và max là 1):

$$x' = \frac{2.\left(x - x_{min}\right)}{x_{max} - x_{min}} - 1$$

Về mặt toán học, phụ thuộc hàm được học thông qua các mô hình RNN có thể viết dưới dạng:

$$\hat{y}_{t+k} = f_k\left(x_{t-w}, \ldots, x_{t-1}, y_{t-w}, \ldots, y_{t-1}\right)$$

với:

$\hat{y}_{t+k}$ là giá trị đầu ra tại thời điểm $t + k$

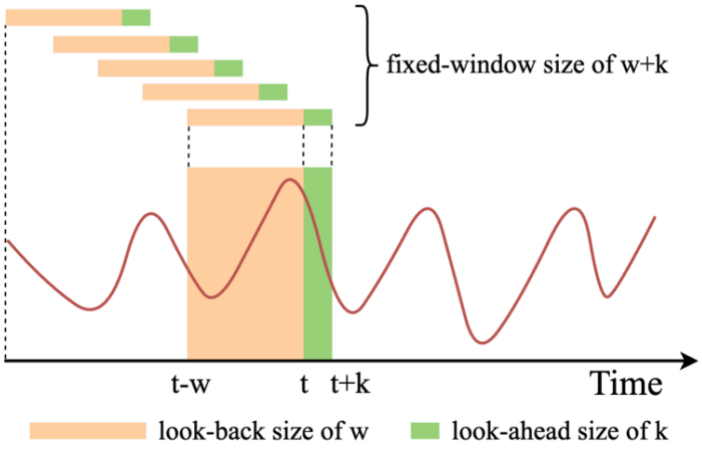
$\boldsymbol{k}$ là độ dài khoảng thời gian từ hiện tại tới thời điểm trong tương lai mà giá trị biến mục tiêu được dự đoán.

$\boldsymbol{y_{t-w}, \ldots, y_{t-1}}$ là các giá trị đầu ra quan sát được từ thời điểm t-w tới t-1

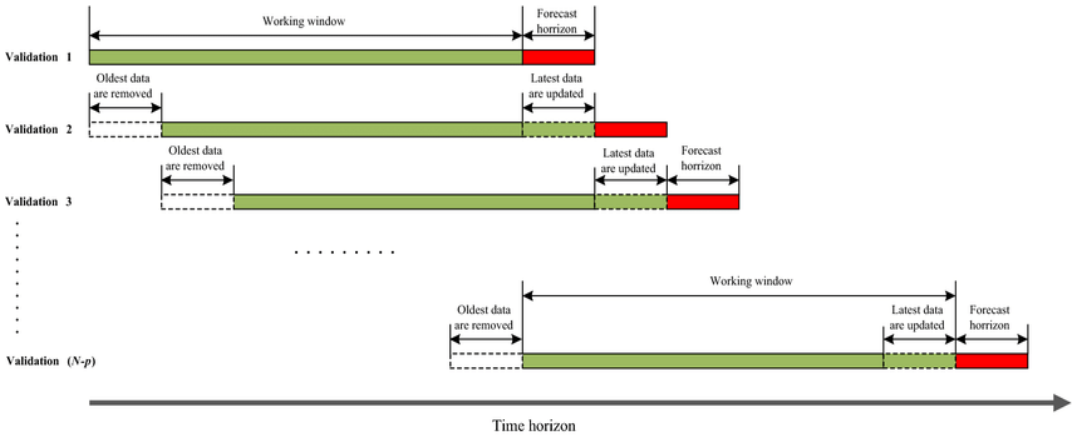
$\boldsymbol{x_{t-w}, \ldots, x_{t-1}}$ là các vector của m đặc trưng đầu vào quan sát được từ thời điểm t-w tới t-1

$\boldsymbol{f_k}$ là phụ thuộc hàm học được từ mô hình RNN.

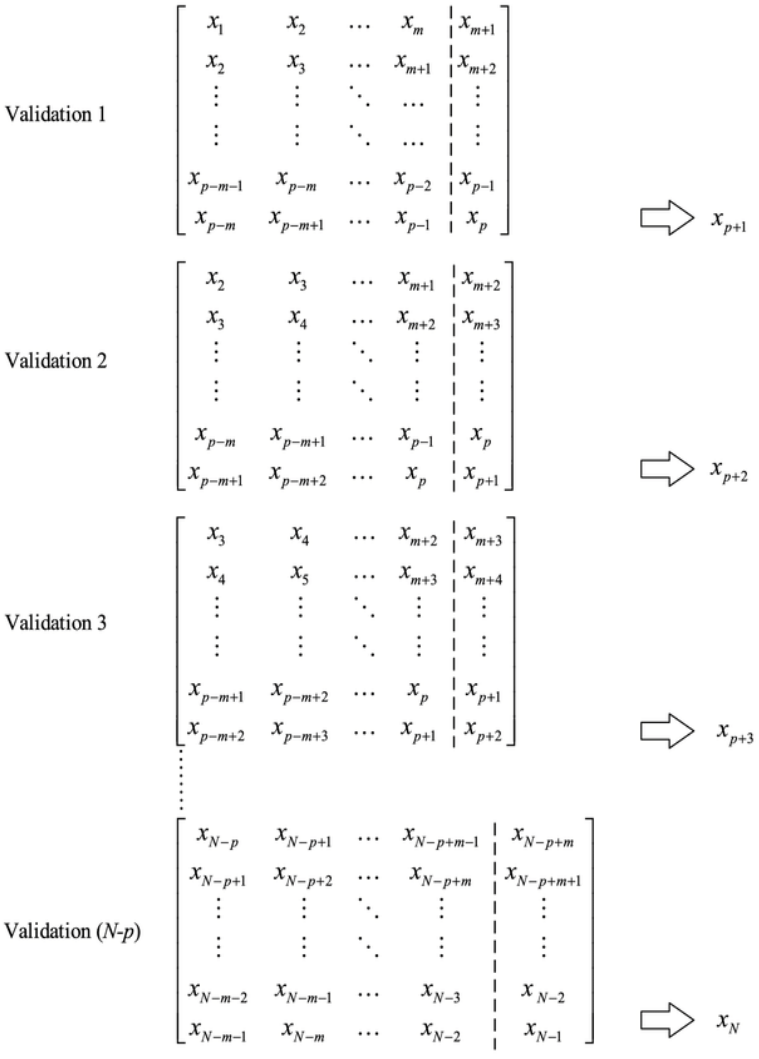
m là số lượng đặc trưng đầu vào
w là kích thước cửa sổ dùng cho đầu vào.



Hình VI.1.1 và VI.1.2:
Sử dụng cửa sổ thời gian trượt có kích thước w để xây dựng tập huấn luyện cho mô hình để dự đoán k bước thời gian trong tương lai



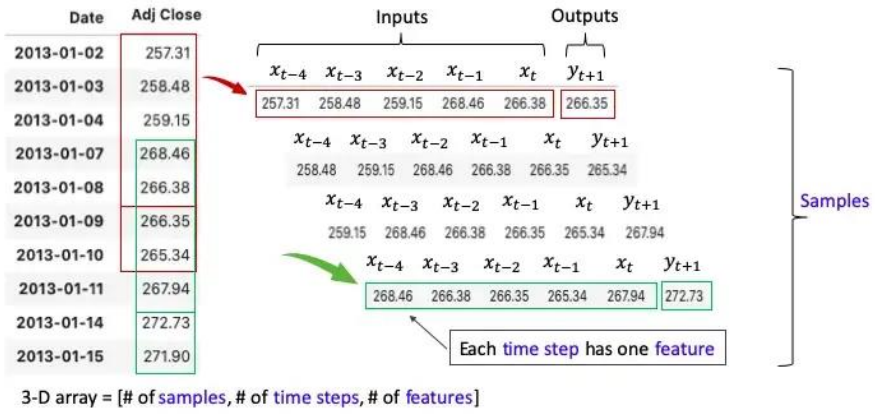
(a) Sliding window representation



(b) State reconstruction for time series analysis

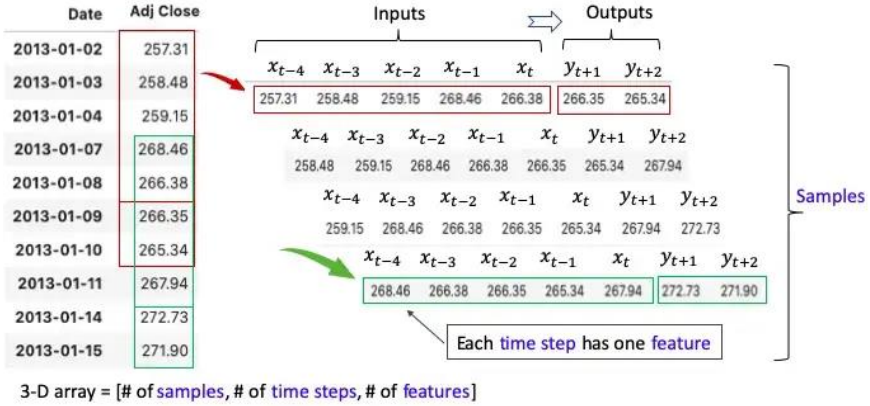
<u>Ví dụ 1:</u> điển hình đối với bài toán dự đoán stock price:

- Many-to-one:



Hình VI.1.3: Cấu trúc Many-to-one của bộ dữ liệu stock price

- Cấu trúc Many-to-one sử dụng giá đóng phiên của 5 ngày để dự đoán giá của ngày tiếp theo.
- Many-to-many:



Hình VI.1.4: Cấu trúc Many-to-many của bộ dữ liệu stock price

- Cấu trúc Many-to-many sử dụng giá đóng phiên của 5 ngày để dự đoán giá của 2 ngày tiếp theo.
- Ví dụ tạo các mẫu từ dữ liệu chuỗi thời gian đơn đặc trưng (univariate) khi cửa sổ màu đỏ trượt dọc theo chuỗi thời gian. Mỗi mẫu có 5 đầu vào và 1 hoặc 2 đầu ra. Mỗi đầu vào của một mẫu được gọi là bước thời gian (thời điểm) và mỗi bước thời gian có một giá trị, được xem là một đặc trưng của bước thời gian đó.

2. Xây dựng mô hình

Chúng ta sẽ cung cấp cho mỗi giá trị input tương ứng với trọng số của chúng bằng cách việc khởi tạo ngẫu nhiên. Sau đó chúng ta sẽ bắt đầu công việc huấn luyện thông qua 6 bước sau:

1. Chuẩn hóa dữ liệu
2. Khởi tạo trọng số
3. Lấy input từ tập dữ liệu, sau đó nhân chúng với bộ trọng số tương ứng và cộng với trạng thái trước đó rồi đưa chúng qua hàm chuyển đổi để tính giá trị output của neural.

4. Chuẩn hóa ngược đầu ra (giá trị dự đoán)
5. Tính lỗi hay tính sự khác biệt giữa giá trị dự đoán và giá trị thực tế.
6. Tùy thuộc vào độ lỗi, ta sẽ tiến hành lan truyền ngược.
7. Cập nhật trọng số và lặp lại quá trình này nhiều lần

Cuối cùng trọng số của các neuron sẽ đạt được tối ưu với các giá trị trong tập huấn luyện. Chúng ta sẽ sử dụng bộ trọng số này cho việc dự đoán tình huống mới.

Để minh hoạ, xét tập dữ liệu stock price như sau:

Date	Adj Close
2013-01-02	257.31
2013-01-03	258.48
2013-01-04	259.15
2013-01-07	268.46
2013-01-08	266.38
2013-01-09	266.35

- Mô hình xây dựng bằng cấu trúc Many-to-one giống ví dụ 1. Chúng ta minh hoạ với mẫu đầu tiên, mà mỗi mẫu của mô hình huấn luyện của ta sẽ có 5 input xt-4, xt-3, xt-2, xt-1, xt và 1 output yt+1 tương ứng với 5 time steps input và 1 time step output khi cửa sổ trượt di chuyển. Mỗi time step là 1 vector chứa các giá trị (tức là các đặc trưng) và ở đây mỗi time step đều chỉ có 1 giá trị nên chiều của các vector này là (1x1). Tập input tổng quát của mô hình sẽ là 1 ma trận có chiều như sau:

  [số mẫu, số time step, số đặc trưng mỗi time step]
- Vì thế, tập input với mẫu dữ liệu minh hoạ trên sẽ là ma trận có chiều **[1x5x1]**

Bước 1: Scale dữ liệu với minmax scaler, ta được tập dữ liệu mới:

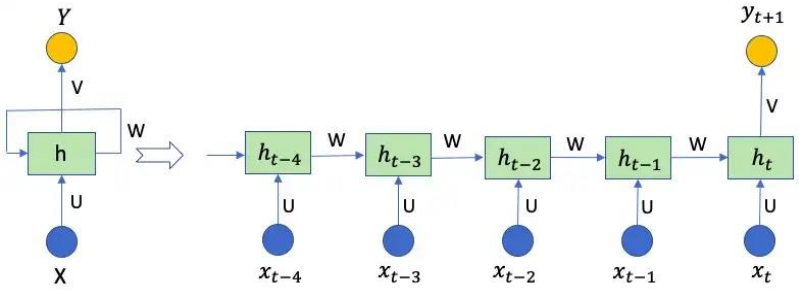
Date	Adj Close
2013-01-02	-1
2013-01-03	-0.7901
2013-01-04	-0.67
2013-01-07	1
2013-01-08	0.6269
2013-01-09	**266.35**

Input: [1x5x1] = [[-1]

[-0.7901]
[-0.67]
[1]
[0.6269]]

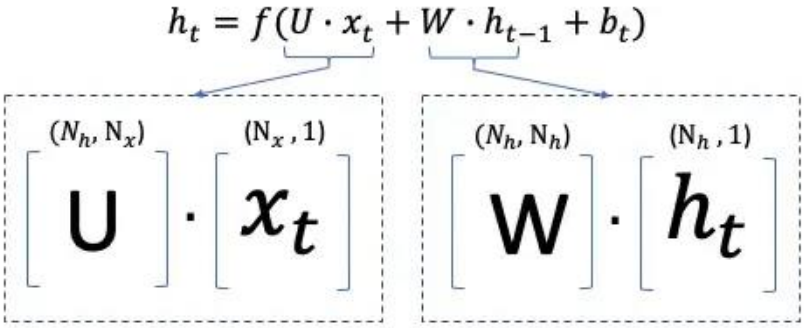
Output: [1x1] = [266.35]

Mô hình RNN sử dụng để huấn luyện sẽ có cấu trúc như sau:



Hình VI.2.1: Cấu trúc RNN

Bước 2: Khởi tạo trọng số U, W và V với U bộ trọng số nối giữa input và hidden, W là bộ trọng số nối giữa hidden và hidden có chiều tuỳ ý nên ta chọn 2 là chiều và V là bộ trọng số nối giữa hidden và output. Chiều của U, W được chọn như sau:



Hình VI.2.2: Quy tắc chọn kích cỡ cho các trọng số

Với N_h là chiều của ma trận trọng số nối các hidden layers, N_x là chiều của vector chứa giá trị đặc trưng của các time step. Nên chiều của U, W lần lượt là (2x1), (2x2) và chiều của V là (1x2). Các giá trị của các ma trận trọng số được khởi tạo ngẫu nhiên bởi giá trị thuộc bảng phân phối chuẩn.

U = [0.9804
-0.6080]

W = [-0.5620 0.3111
0.4655 -0.2602]

V = [0.8718 -0.5877]

Bước 3: Thực hiện tính toán với công thức với độ lỗi bias là 0

$$h_t = f(U.x_t + W.h_{t-1})$$

Chọn hàm *tanh* làm hàm kích hoạt:

$$f(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

Với *t = 1*:

$$h_1 = f(U.x_1 + W.h_0) = f(U.x_1 + 0) = f(U.x_1)$$
$$h_1 = [-0.7532 \quad 0.5427]$$
$$(x_1 = [-1], h_0 = 0)$$

Với $t = 2$:

$$h_2 = f(U.x_2 + W.h_1)$$
$$h_2 = [-0.0345 \quad -0.1334]$$
$$(x_2 = [-0.7901])$$

Với $t = 3$:

$$h_3 = f(U.x_3 + W.h_2)$$
$$h_3 = [-0.5910 \quad 0.4021]$$
$$(x_3 = [-0.67])$$

Với $t = 4$:

$$h_4 = f(U.x_4 + W.h_3)$$
$$h_4 = [0.9042 \quad -0.7759]$$
$$(x_4 = [1])$$

Với $t = 5$:

$$h_5 = f(U.x_5 + W.h_4)$$
$$h_5 = [-0.4986 \quad 0.5255]$$
$$(x_5 = [0.6269])$$

Tính output:

$$\hat{y}_{t+1} = g\left(V.h_t\right)$$

Chọn hàm *sigmoid* cho *g* để làm hàm kích hoạt:

$$g(z) = \frac{1}{1 + e^{-z}}$$

$=> \hat{y}_6 = g(V.h_5) = 0.3057$

Bước 4**:** Chuẩn hóa ngược output:

$$y = \frac{\left((y' + 1).\left(x_{max} - x_{min}\right)\right)}{2} + x_{min}$$

$=> y = 264.5893$

Bước 5**:** Tính sai số giữa output dự đoán và output thực tế:

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{\left(\hat{y}_i - y_i\right)^2}{n}}$$

$=> rmse = 1.7607$

Bước 6**:** Sau khi tính được sai số giữa kết quả dự đoán và kết quả thực tế, ta tiến hành lan truyền ngược để tối ưu hóa trọng số, thông qua các công thức sau:

Trọng số V**:**

$$\frac{dL}{dV} = \frac{dL}{d\hat{y}} \cdot \frac{d\hat{y}}{dV}$$
$$V' = V - \alpha . \frac{dL}{dV}$$

Trọng số U**:**

$$\frac{dL}{dU} = \frac{dL}{d\hat{y}} \cdot \frac{d\hat{y}}{dh_w} \cdot \frac{dh_w}{dU} = \sum_{i=0}^{w} \frac{dL}{d\hat{y}} \cdot \frac{d\hat{y}}{dh_w} \cdot \left(\prod_{j=i}^{w-1} \frac{dh_{j+1}}{dh_j} \right) \cdot \frac{dh_i'}{dU}$$
$$U' = U - \alpha . \frac{dL}{dU}$$

Trọng số W**:**

$$\frac{dL}{dW} = \frac{dL}{d\hat{y}} \cdot \frac{d\hat{y}}{dh_w} \cdot \frac{dh_w}{dW} = \sum_{i=0}^{w} \frac{dL}{d\hat{y}} \cdot \frac{d\hat{y}}{dh_w} \cdot \left(\prod_{j=i}^{w-1} \frac{dh_{j+1}}{dh_j} \right) \cdot \frac{dh_i'}{dW}$$
$$W' = W - \alpha . \frac{dL}{dW}$$

Với:

w là kích thước cửa sổ thời gian trượt (số time steps đầu vào)

α là tỉ lệ học (learning rate) cho trước

$\frac{dh_i'}{dU}$ và $\frac{dh_i'}{dW}$ lần lượt là đạo hàm của s_i với ***U*** và ***W*** khi coi s_{i-1} là hằng số với ***U, W***.

Bước 7: Lặp lại quá trình trên *epochs* lần.

VII.Kết luận

1. Dữ liệu và kết quả thực nghiệm

- Bài báo ứng dụng 4 kỹ thuật học máy với bộ dữ liệu chất lượng không khí theo giờ ở Bắc Kinh trong khoảng thời gian 5 năm từ ngày 1 tháng 1 năm 2020 đến ngày 31 tháng 12 năm 2014 từ trang web UCI, thông qua bài toán dự đoán trên tập dữ liệu chuỗi thời gian để dự đoán chất lượng không khí tại đây trong tương lai.
- Bài báo phân loại thực nghiệm thành hai mô hình, 1 mô hình dự đoán 1 bước thời gian (1 thời điểm) và 1 mô hình dự đoán 1 khoảng thời gian liên tiếp gồm nhiều bước thời gian, sử dụng dữ liệu chuỗi thời gian lấy từ những bước thời gian liên tục trước đó.

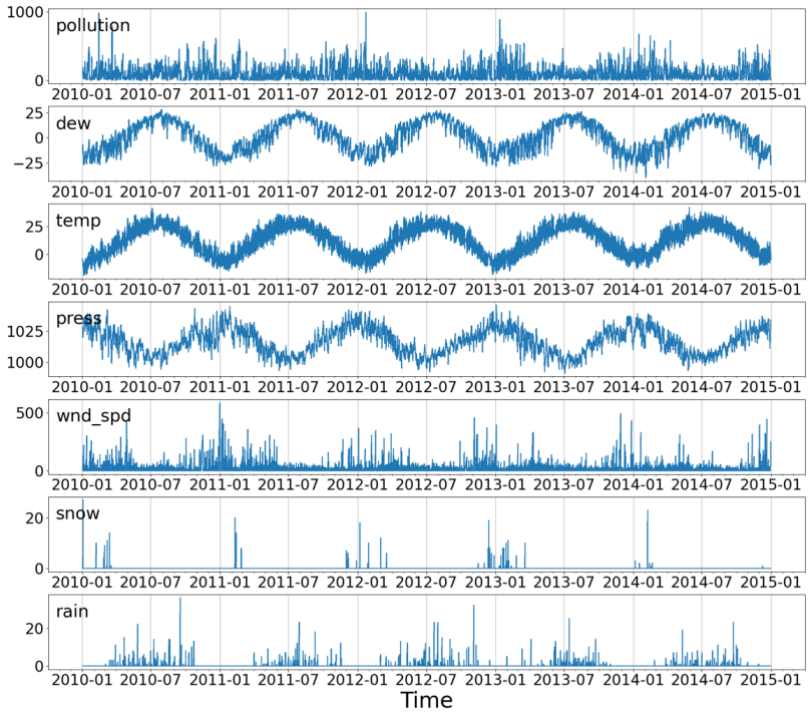
a. Về tập dữ liệu

Bộ dữ liệu được thu thập theo giờ, bao gồm 43,824 dòng and 13 cột. Cột đầu tiên đơn thuần là chỉ mục và được bỏ qua trong quá trình phân tích. Bốn cột được tiếp theo là năm, tháng, ngày và giờ, được kết hợp thành một đặc trưng duy nhất đặt tên là “time”. Cột “PM2.5” là biến mục tiêu. Tất cả các biến khác được sử dụng như các đặc trưng đầu vào. Tên cột và ý nghĩa của chúng được chú thích trong bảng sau:

Tên cột	Ý nghĩa	Ghi chú

No	Số chỉ hàng	Bỏ qua
Year	Năm	Kết hợp thành cột "Time"
Month	Tháng	
Day	Ngày	
Hour	Giờ	
PM2.5	Mức độ bụi tập trung	Đặc trưng đầu ra
DEWP	Điểm ngưng sương	Đặc trưng đầu vào
TEMP	Nhiệt độ	
PRES	Áp suất	
cbwd	Hướng gió	
Iws	Tốc độ gió	
Is	Giờ tuyết tích luỹ	
Ir	Giờ mưa tích lũy	

- Nhóm nghiên cứu tiến hành loại bỏ đặc trưng 'cbwd' với lí do đặc trưng này vừa thuộc dạng phân loại, vừa không mang ý nghĩa gì đối với bài toán đặt ra. Chuỗi thời gian cho tất cả đặc trưng đầu vào và đặc trưng đầu ra trừ cột 'time' và cột 'cbwd' được minh hoạ trong hình dưới đây:



Hình VII.1.1: Mô tả các đặc trưng trong bộ dữ liệu

b. Về thực nghiệm

- Các thí nghiệm với $\boldsymbol{k}$ = 1 dự đoán một time step trong tương lai được gọi là dự đoán đơn bước (single-step prediction), trong khi các thí nghiệm với $\boldsymbol{k}$ > 1 dự đoán một hoặc nhiều mốc thời gian xa hơn trong tương lai được gọi là dự đoán đa bước (multi-step prediction).
- Cả hai bộ thí nghiệm được thực hiện với các giá trị khác nhau của $\boldsymbol{w}$ - kích thước của cửa sổ thời gian trượt đại diện cho một khoảng thời gian của quá khứ gần đây được sử dụng làm đầu vào. Kích thước cửa sổ $\boldsymbol{w}$ = 1, 2, 4, 8 và 16 ngày (tức 24, 48, 96, 192, và 384 giờ) được sử dụng trong cả hai bộ thực nghiệm. Các sự lựa chọn theo cấp số nhân của kích thước cửa sổ được chọn để hiểu tác động của kích thước cửa sổ lên độ chính xác của bài toán dự đoán.
- Các dự đoán đa bước được sử dụng để dự đoán giá trị chất lượng không khí tại các điểm thời gian $\boldsymbol{k}$ = 1, 2, 4, 8 và 16 giờ tới trong tương lai.
- 4 mô hình học máy được sử dụng gồm có RNN, LSTM, GRU và Transformer. Các mô hình được khởi tạo các giá trị *epochs, learning rate, batch size* và *optimizer* như sau:

SETTINGS OF VARIOUS DEEP LEARNING MODELS

	Epoch	LR	Batch	Optimizer
RNN	100; 200	0.0005; 0.00001	256; 512	Adam
LSTM	100; 200	0.0005; 0.00001	256; 512	Adam
GRU	100; 200	0.0005; 0.00001	256; 512	Adam
Transformer	200; 300	0.0005; 0.00005	256; 512	AdamW [54]

Hình VII.1.2: Giá trị khởi tạo cho epoch, learning rate và batch size

- Với hàm tính độ lỗi của kết quả dự đoán, bài báo đã sử dụng hàm *Mean Absolute Error* và *Root of Mean Squared Error* để ước lượng độ chính xác của các thuật toán trên nhiều thiết lập mô hình khác nhau.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|\hat{y}_i - y_i\right| \quad RMSE = \sqrt{\sum_{i=1}^{n}\frac{\left(\hat{y}_i - y_i\right)^2}{n}}$$

Hình VII.1.3: Công thức MAE và RMSE

c. Về kết quả thực nghiệm

Kết quả thực nghiệm với mô hình sử dụng cửa sổ có kích thước *w cố định* (4 ngày tức 96 giờ) để dự đoán đa bước các khoảng thời gian **k** trong tương lai:

PERFORMANCE (MAE AND RSME) OF MULTI-STEP PREDICTION SHOWN AS A FUNCTION OF *k*, THE NUMBER OF HOURS INTO THE FUTURE FOR WHICH THE PREDICTION IS BEING MADE. BEST RESULTS IN EACH ROW ARE IN BOLD FONT.

Future Timesteps (*k*)	RNN		LSTM		GRU		Transformer	
	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
1 hour	15.183	25.262	14.997	23.804	**14.107**	23.782	14.882	**23.573**
2 hours	18.332	29.846	16.983	28.038	16.555	28.479	**15.126**	**27.421**
4 hours	31.687	47.468	29.880	44.468	30.113	45.321	**23.459**	**38.115**
8 hours	44.599	66.166	42.964	**64.626**	42.923	65.910	**41.854**	64.641
16 hours	53.424	75.301	50.662	72.225	52.950	73.053	**50.339**	**72.218**

Hình VII.1.4: Kết quả thực nghiệm

- Đánh giá: một điều khác rõ ràng từ bảng trên là độ chính xác của mô hình dự đoán giảm đi (độ lỗi tăng) khi k tăng, và giảm càng mạnh khi dự đoán khoảng thời gian *k >= 4*. Không kể đến mô hình thuật toán Transformer, mô hình GRU cho ra kết quả chính xác nhất khi dự đoán khoảng thời gian *k < 4* nhưng với *k >= 4* thì LSTM cho ra kết quả chính xác nhất. Mô hình RNN luôn cho ra kết quả kém chính xác nhất ở mọi ***k***.

- Kết quả thực nghiệm với mô hình sử dụng nhiều cửa sổ có kích thước *w* khác nhau để dự đoán đơn bước khoảng thời gian *k = 1*:

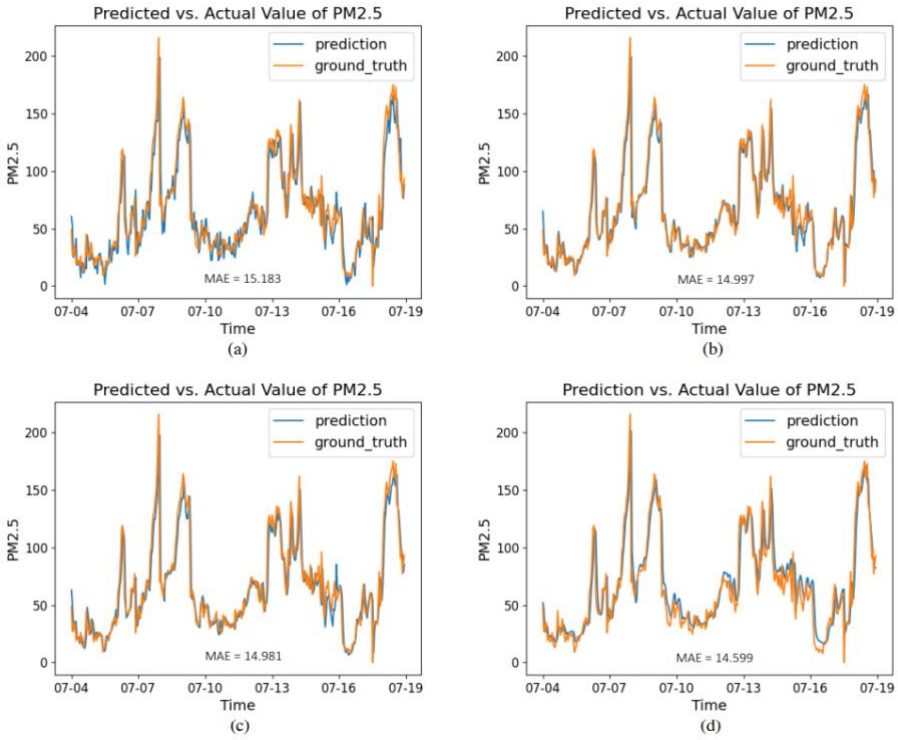
PERFORMANCE (MAE AND RSME) OF SINGLE-STEP PREDICTION (LOOK-BACK WINDOW (4 DAYS) TO PREDICT 1 HOURS AHEAD) SHOWN AS A FUNCTION OF *w*, THE SIZE OF THE LOOK-BACK WINDOW USED FOR THE PREDICTION. BEST RESULTS IN EACH ROW ARE IN BOLD FONT.

Look-back Window	RNN		LSTM		GRU		Transformer	
	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
1 day	13.895	24.109	12.893	**23.492**	**12.861**	23.745	13.538	23.950
2 days	14.878	24.958	13.566	24.090	**13.180**	**23.878**	13.946	23.896
4 days	15.183	25.262	14.997	23.804	14.981	23.782	**14.599**	**23.573**
8 days	17.031	26.998	16.089	24.379	16.892	24.886	**15.255**	**23.499**
16 days	20.538	30.511	18.991	27.924	17.811	26.892	**15.837**	**24.637**

Hình VII.1.5: Độ lỗi của các mô hình

- Đánh giá: mô hình Transformer cho ra kết quả tốt hơn rất nhiều so với các mô hình còn lại, đặc biệt là ở các giá trị kích thước cửa sổ w >= 96, ngoài ra tính ổn định của mô hình cũng được thể hiện rõ khi độ chính xác chỉ giảm nhẹ khi kích thước w tăng. Mô hình GRU cho ra kết quả chính xác nhất khi dự đoán kích thước cửa sổ w <= *96* nhưng với *w > 96* thì LSTM cho ra kết quả chính xác nhất. Mô hình RNN luôn cho ra kết quả kém chính xác nhất ở mọi ***w***.

- Các biểu đồ dưới mô tả sự khác biệt giữa giá trị dự đoán và giá trị thực tế của mô hình trên với *w=96, k=1* sử dụng (a) RNN, (b) LSRM, (c) GRU, (d) Transformer models:



Hình VII.1.6: Trực quan kết quả dự đoán so với giá trị thực tế của các mô hình

- Kết quả thực nghiệm với mô hình sử dụng nhiều cửa sổ có kích thước *w* khác nhau để dự đoán đa bước khoảng thời gian *k = 3*:

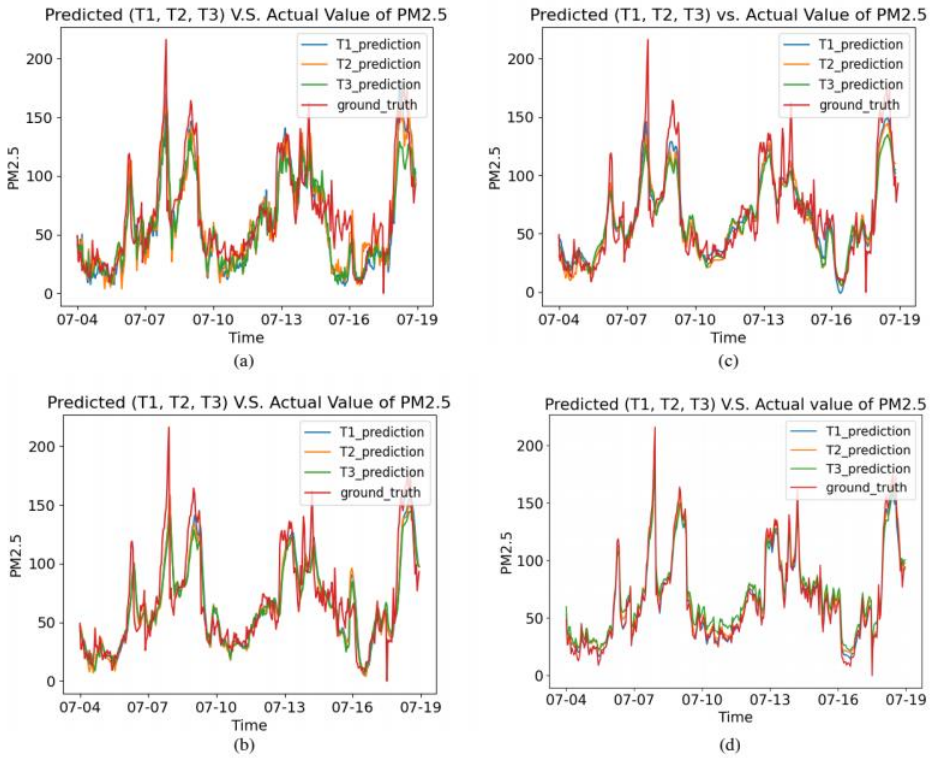
PERFORMANCE (MAE AND RSME) OF MULTI-STEPS PREDICTION (LOOK-BACK WINDOW (4 DAYS) TO PREDICT 3 HOURS AHEAD) SHOWN AS A FUNCTION OF *w*, THE SIZE OF THE LOOK-BACK WINDOW USED FOR THE PREDICTION. BEST RESULTS IN EACH ROW ARE IN BOLD FONT.

Look-back Window	RNN		LSTM		GRU		Transformer	
	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
1 day	27.732	43.645	26.225	42.564	25.360	40.625	**23.998**	**38.767**
2 days	26.068	42.008	25.976	42.091	25.107	39.266	**23.676**	**38.131**
4 days	27.729	43.429	25.994	41.872	25.818	39.501	**23.273**	**39.889**
8 days	30.002	47.191	26.272	42.685	25.340	40.519	**24.221**	**39.443**
16 days	35.958	53.742	29.991	45.816	28.887	44.284	**26.192**	**43.889**

Hình VII.1.7: Độ lỗi của mô hình ở k=3

- Đánh giá: mô hình Transformer vẫn cho ra kết quả tốt hơn rất nhiều so với các mô hình còn lại. RNN vẫn là mô hình cho ra kết quả kém nhất, tuy nhiên, GRU lại cho ra kết quả dự đoán tốt hơn ở mọi ***w*** so với LSTM thay vì chỉ tốt hơn ở *w <= 96* như thí nghiệm với *k = 1*. Nhưng, điều đáng chú ý nhất lại là giá trị độ lỗi cho ra từ kết quả dự đoán của các mô hình đạt nhỏ nhất với *w = 48 hoặc 96* thay vì là ở *w = 24* ở thí nghiệm trước đó. Điều này chứng tỏ giá trị ***w*** này có thể là giá trị tối ưu cho mô hình thuật toán.

- Các biểu đồ dưới mô tả sự khác biệt giữa giá trị dự đoán và giá trị thực tế của mô hình trên với *w=96* sử dụng (a) RNN, (b) GRU, (c) LSTM, (d) Transformer models với T1, T2, T3 ứng với kết quả của thí nghiệm với *k = 1, 2, và 3*:



Hình VII.1.8: Kết quả dự đoán so với giá trị thực tế của các mô hình với k=1,2,3

2. Đánh giá mô hình

- Mô hình Transformer cho ra kết quả tốt nhất khi dự đoán khoảng thời gian càng xa.
- LSTM và GRU cho ra kết quả tốt vượt trội hơn hẳn RNN khi dự đoán khoảng thời gian gần và cũng luôn duy trì hiệu suất tốt hơn khi dự đoán khoảng thời gian càng xa, điều này có thể chứng tỏ GRU và LSTM có bộ nhớ dài hạn hơn RNN và đã giải quyết một phần vấn đề mất mát đạo hàm (Vanishing Gradient) ...
- Đối với dự đoán đa bước, Transformer vượt trội các phương pháp khác. Đối với dự đoán đơn bước, Transformer cho ra kết quả tốt hơn nếu cửa sổ thời gian trượt càng dài; GRU và LSTM sẽ tốt hơn khi giá trị ***w*** nhỏ.
- Về sự ảnh hưởng của kích thước cửa sổ thời gian trượt, độ lỗi hiển thị mức tối thiểu cục bộ bởi ta luôn xác định được các giá trị ***w*** tối thiểu (giá trị tối ưu) trong các thí nghiệm nhưng chưa thể tìm ra giá trị ***w*** tối ưu nhất cho bài toán dự đoán trên tập dữ liệu này. Đối với dự đoán đơn bước, giá trị tối ưu của kích thước cửa sổ là w = 24. Đối với dự đoán đa bước, giá trị tối ưu là w = 48 hoặc 96 (khi dự đoán với k = 3).

3. Tổng kết - Đánh giá bài báo

- Bài toán dự đoán trên dữ liệu chuỗi thời gian chia làm ba giai đoạn chính: xây dựng mô hình dự đoán, tiến hành dự đoán và kiểm định mô hình.
- Về vấn đề xây dựng mô hình dự đoán thì có nhiều phương pháp khác nhau để xây dựng, bài báo tiến hành xây dựng mô hình RNN, LSTM, GRU và Transformer. Và chúng em chỉ đề cập đến các mô hình liên quan đến Deep Neural Network đó là RNN, LSTM và GRU.
- RNN được thiết kế để làm việc với dữ liệu kiểu chuỗi (Sequential) bằng việc sử dụng thông tin xuất hiện trước đó trong chuỗi làm đầu vào để đem lại kết quả đầu ra tại thời điểm hiện tại hoặc tương lai. Tuy nhiên, RNN gặp vấn đề khi quá trình trải qua quá dài dẫn đến mô hình quên đi những gì xa xôi trong quá khứ. Chính vì thế, LSTM và GRU xuất hiện để phần nhiều giải quyết được vấn đề này. Chi tiết về các mô hình đã được trình bày ở các chương trên.
- Nhóm nghiên cứu đã chia tập dữ liệu thành tập huấn luyện (70%) và tập kiểm thử (30%). Sau khi mô hình đã được huấn luyện, nhóm nghiên cứu đã tiến hành dự đoán dựa trên tập kiểm thử.
- Bước cuối là tiến hành kiểm định mô hình để xem xét mô hình đã phù hợp chưa và so sánh kết quả tập kết quả dự đoán với tập kiểm thử thông qua chỉ số MAE và RMSE để từ đó rút ra kết luận và đánh giá.
- Bài báo này cho chúng ta một hướng tiếp cận với bài toán dự đoán trên dữ liệu chuỗi thời gian, biết rằng ứng dụng mô hình RNN, LSTM và GRU để dự đoán mặc dù đã có lâu nhưng thông qua đây chúng ta đã biết cách dự đoán dựa vào các mô hình trên. Những ý nghĩa bài báo mang lại có thể kể đến như sau:
- Tìm hiểu bài toán dự đoán trên dữ liệu chuỗi thời gian và các phương pháp hiện có để dự đoán trên dữ liệu chuỗi thời gian
- Nghiên cứu các mô hình mạng neural hồi quy nói chung (bao gồm 2 biến thể) và ứng dụng của chúng trong việc xử lý các dữ liệu kiểu chuỗi, cụ thể là dự đoán trên dữ liệu chuỗi thời gian.
- Dùng chỉ số đánh giá MAE và RMSE để so sánh giữa các mô hình RNN, LSTM và GRU trên tập dữ liệu với kích thước cửa sổ thời gian trượt và khoảng thời gian dự đoán khác nhau để tìm ra các giá trị tối ưu nhất.

VIII. Nguồn tham khảo

360DigiTMG Team. “RNN and its Variants in AI.” *360DigiTMG*, 23 July 2021,

https://360digitmg.com/blog/rnn-and-its-variants-in-artificial-intelligence

Biswal, Avijeet. “Recurrent Neural Network (RNN) Tutorial: Types and Examples [Updated].” *Simplilearn*, 30 November 2022, https://www.simplilearn.com/tutorials/deep-learning-tutorial/rnn

Kalita, Debasish. “A Brief Overview of Recurrent Neural Networks (RNN).” *Analytics Vidhya*, 11 March 2022, https://www.analyticsvidhya.com/blog/2022/03/a-brief-overview-of-recurrent-neural-networks-rnn/

Kuo, Chris. “A Technical Guide on RNN/LSTM/GRU for Stock Price Prediction.” *Medium*, 6 December 2020, https://medium.com/swlh/a-technical-guide-on-rnn-lstm-gru-for-stock-price-prediction-bce2f7f30346

Shi, Jimeng, et al. “Time Series Forecasting (TSF) Using Various Deep Learning Models.” 23 April 2022, https://arxiv.org/abs/2204.11115

Wang, Jay, et al. “CNN Explainer.” *Polo Club of Data Science*, 1 May 2020, https://poloclub.github.io/cnn-explainer/