Time Series Forecasting with Deep Neural Network

using Recurrent Neural Network

Thành viên

Mai Quý Trung

20127370

Đặng Nguyễn Duy

20127013

Nguyễn Đức Bảo

20127005

Trương Samuel

20127610

Nội dung nghiên cứu

I. Introduction to Neural Network

Giới thiệu về mạng Neural Network, Deep Neural Network và các thuật ngữ, kiến thức đi kèm.

II. Time Series & Publications

Tìm hiểu về dữ liệu chuỗi thời gian (Time Series), các đặc điểm, tính chất và đặc trưng của kiểu dữ liệu này.

III. Recurrent Neural Network (RNN)

Nghiên cứu về thuật toán RNN, đặc điểm nổi bật, điểm yếu và các phiên bản nâng cấp của RNN

IV. Time Series forecasting with RNN

Áp dụng thuật toán RNN vào bài toán dự đoán trong chuỗi thời gian.

V. Conclusion

Đưa ra các kết luận về thuật toán RNN, Time Series và bài toán dự đoán trong chuỗi thời gian.

I.

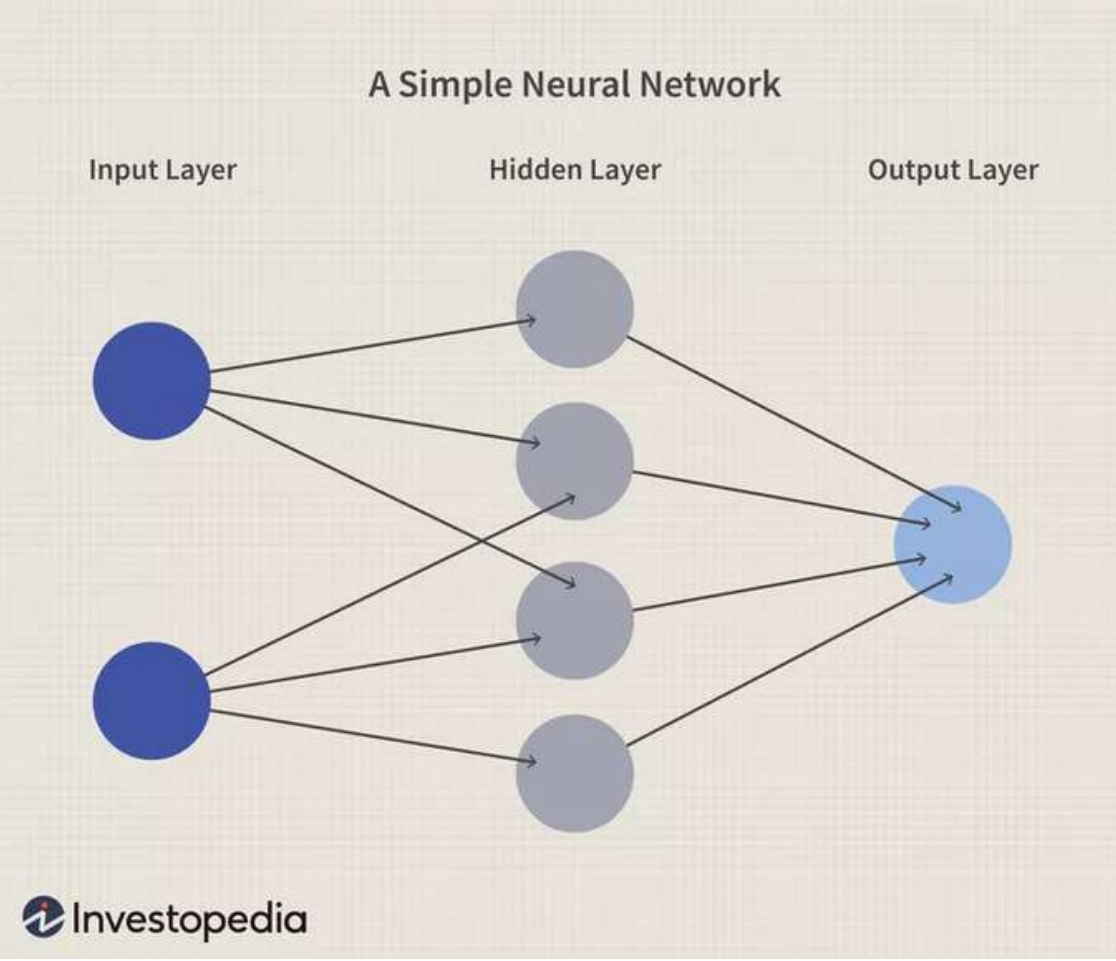
Introduction to Neural Network

Neural Network (Mạng Neuron nhân tạo)

- 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.

- Dùng để giải quyết các bài toán học máy phức tạp

Neural Network (Mạng Neuron nhân tạo)



Sơ đồ minh họa mô hình Neural Network.

Neural Network (Mạng Neuron nhân tạo)

Activation function (hàm kích hoạt)

Hàm này sẽ quyết định xem đầu vào của Mạng có quan trọng hay không trong quá trình dự đoán bằng các phép toán đơn giản hơn.

Gradient descent

Thuật toán giúp mô hình học tốt hơn theo thời gian bằng cách tìm cực trị của hàm thông qua việc tính độ lỗi của mô hình và cập nhật trọng số của hàm.

Neural Network (Mạng Neuron nhân tạo)

Loss function

Hàm tính toán khoảng cách giữa đầu ra hiện tại của thuật toán và đầu ra dự kiến.

Back propagation

Phương pháp tinh chỉnh các trọng số của mạng thần kinh dựa trên độ lỗi thu được trong kết quả trước đó.

Deep Neural Network

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.

Deep Neural Network

Một số thuật toán DNN điển hình

- Convolution Neural Network (CNN)
- Autoencoder
- Restricted Boltzmann Machine (RBM)
- **Recurrent Neural Network (RNN)**

II.

Time Series & Publications

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.

Đặc trưng

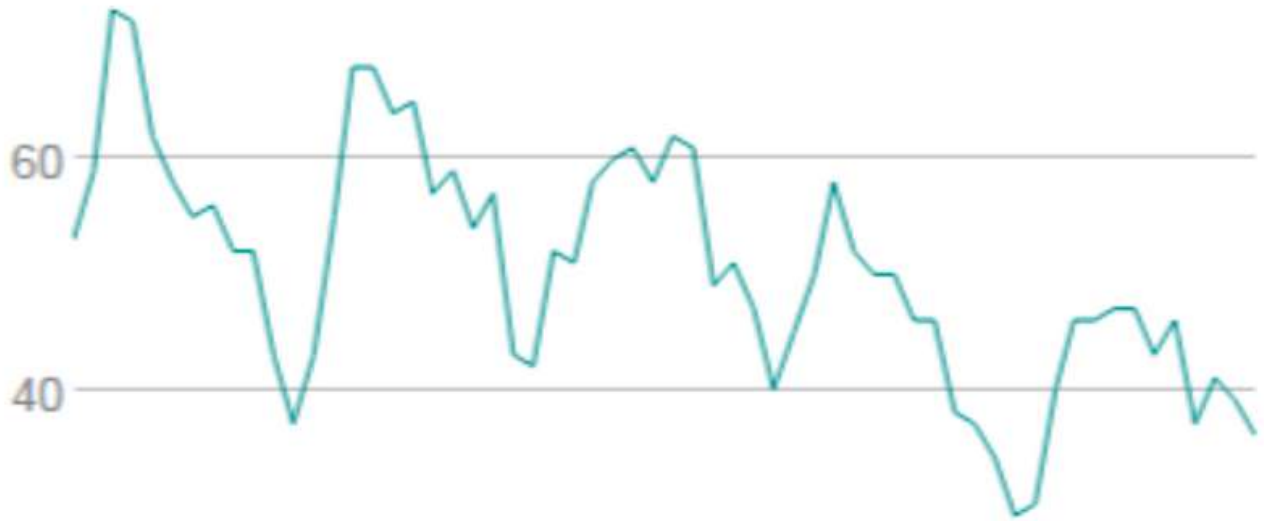
Trend (xu hướng)

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, một Time Series có thể thể hiện cả 2.

Đặc trưng

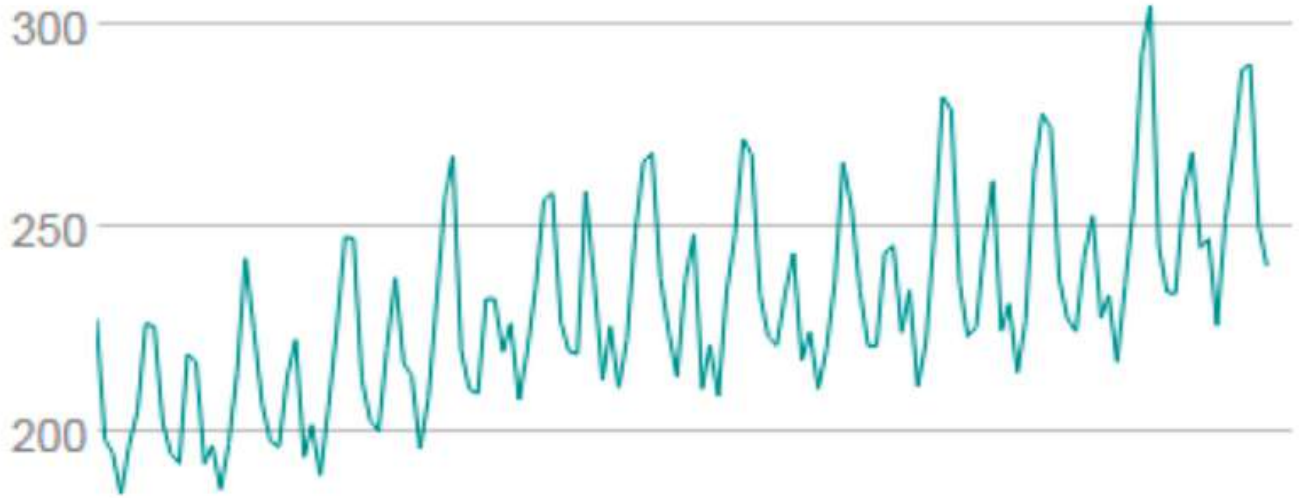
Trend (xu hướng)



Chuỗi có xu hướng giảm dần

Đặc trưng

Trend (xu hướng)



Chuỗi có xu hướng giảm dần

Đặc trưng

Trend (xu hướng)

Chuỗi không có xu hướng

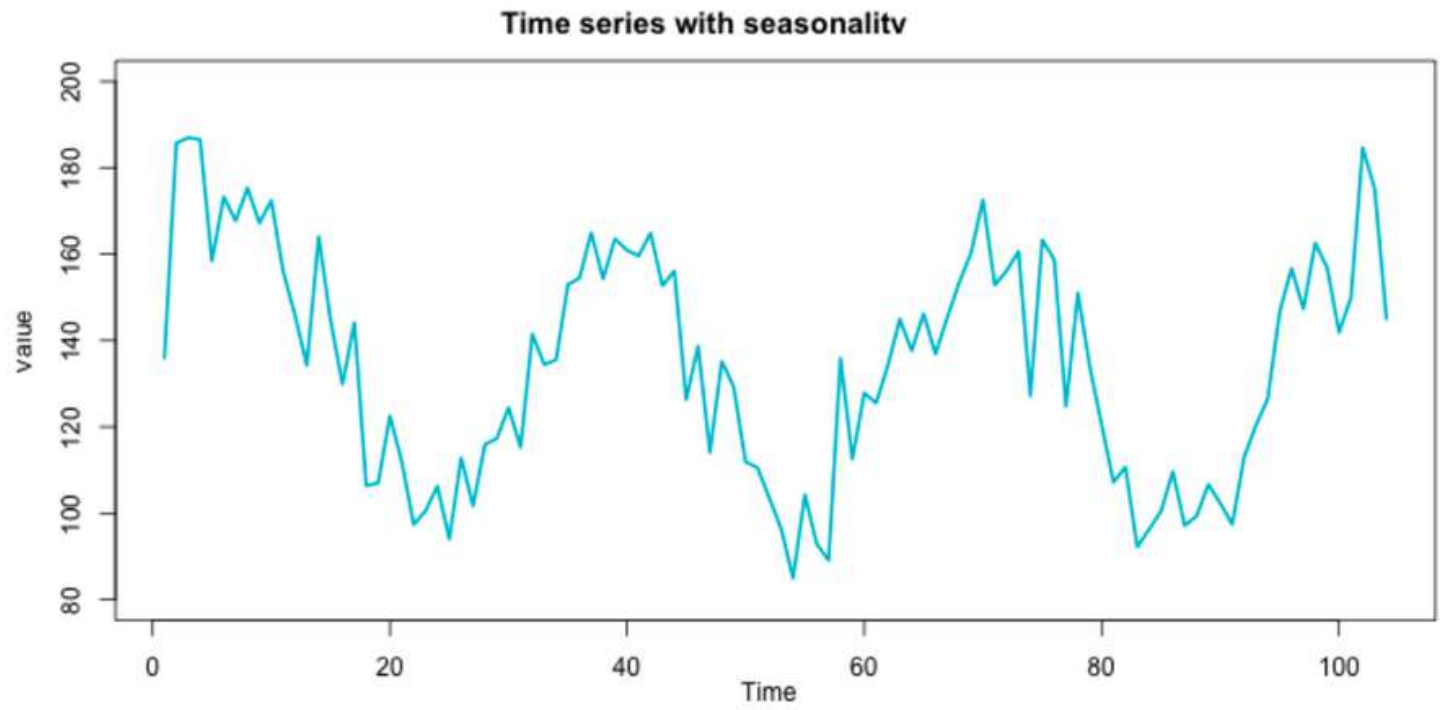
Đặc trưng

Seasonal (tính mùa)

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

Đặc trưng

Seasonal (tính mùa)



Chuỗi có tính mùa

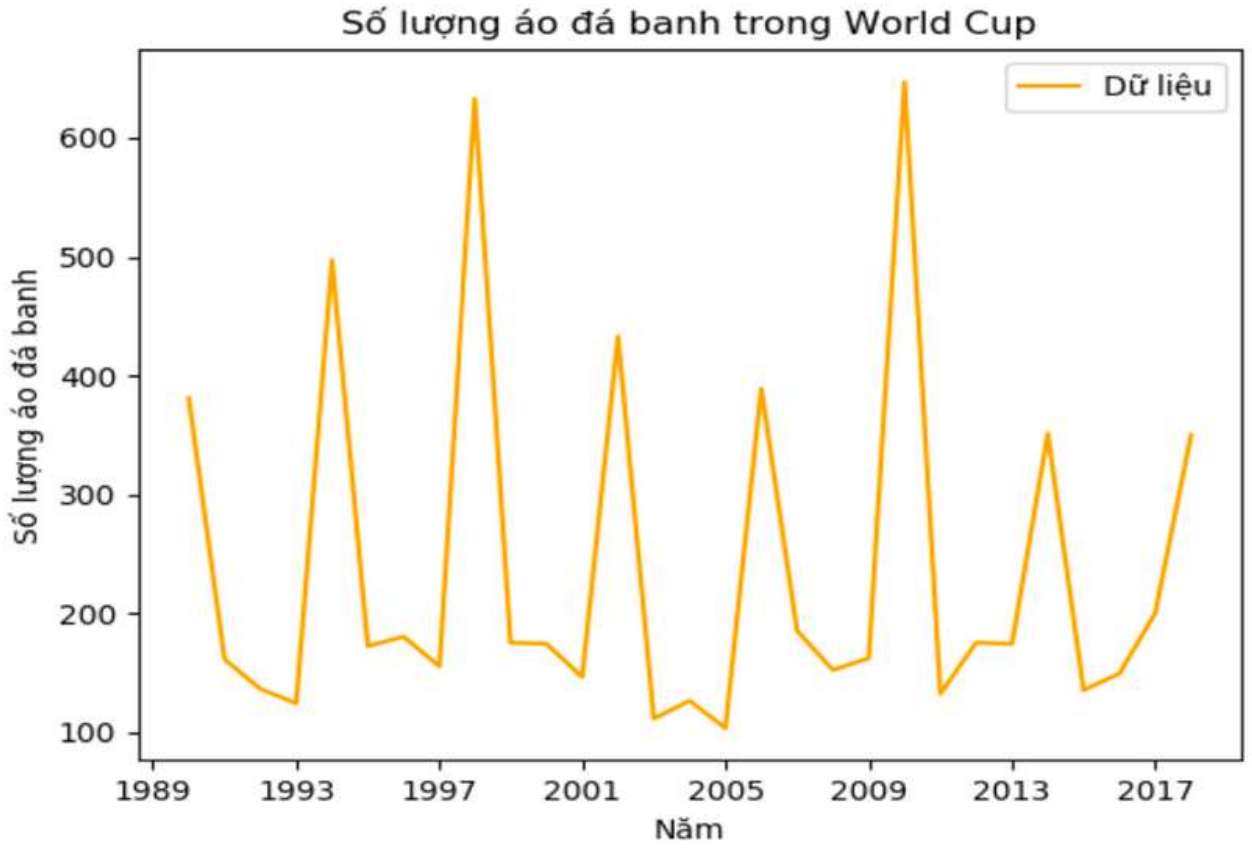
Đặc trưng

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)

Đặc trưng

Cyclical (tính chu kì)



Chuỗi có tính chu kì

Đặc trưng

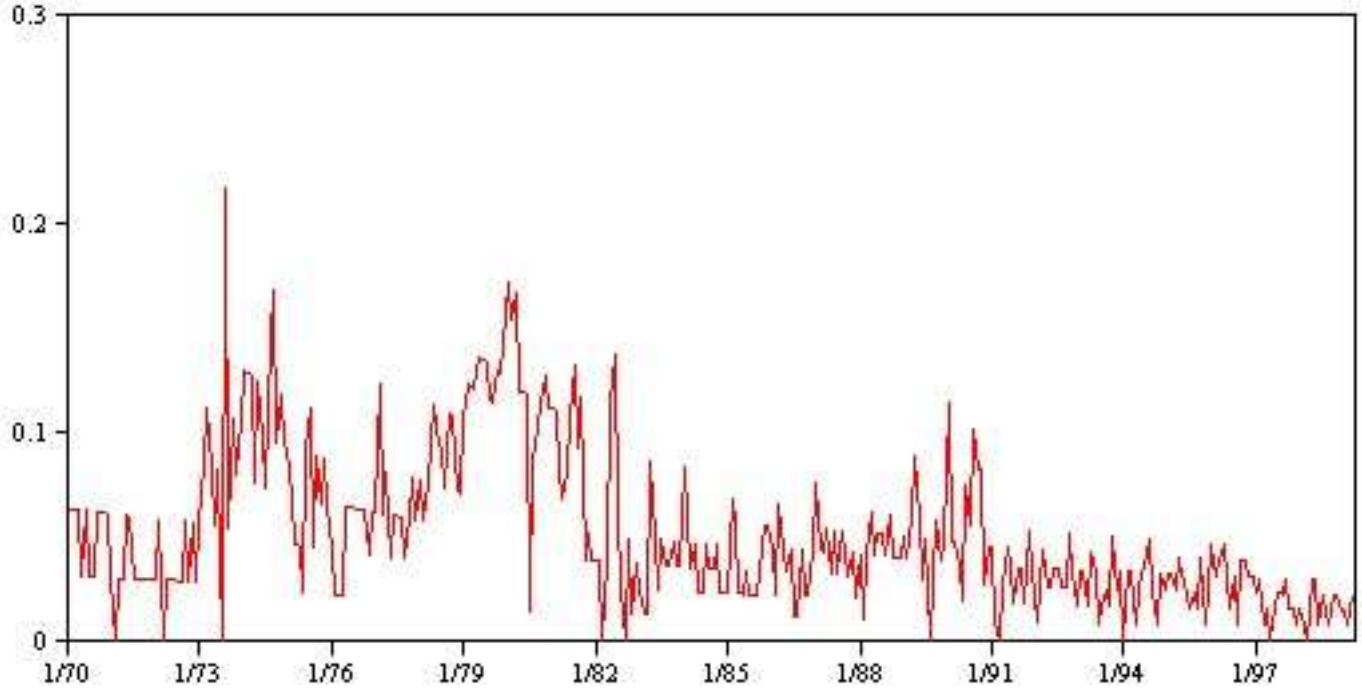
Pulse và Step

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

- Pulse: sự chuyển dịch đột ngột, tạm thời trong chuỗi.
- Step: sự chuyển dịch đột ngột, dài hạn trong chuỗi.

Đặc trưng

Pulse và Step



Chuỗi có pulse

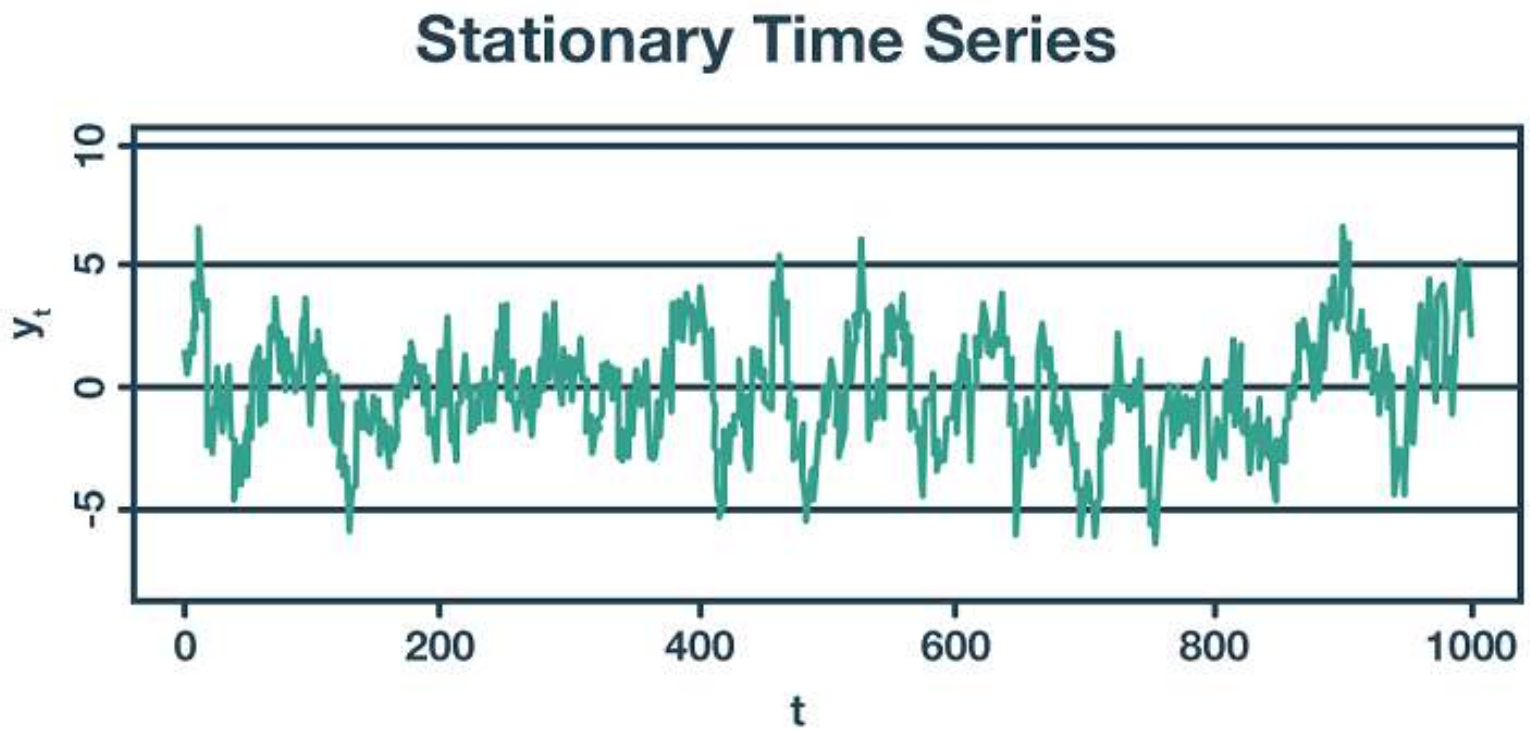
Đặc trưng

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)

Đặc trưng

Stationary (tính tĩnh)



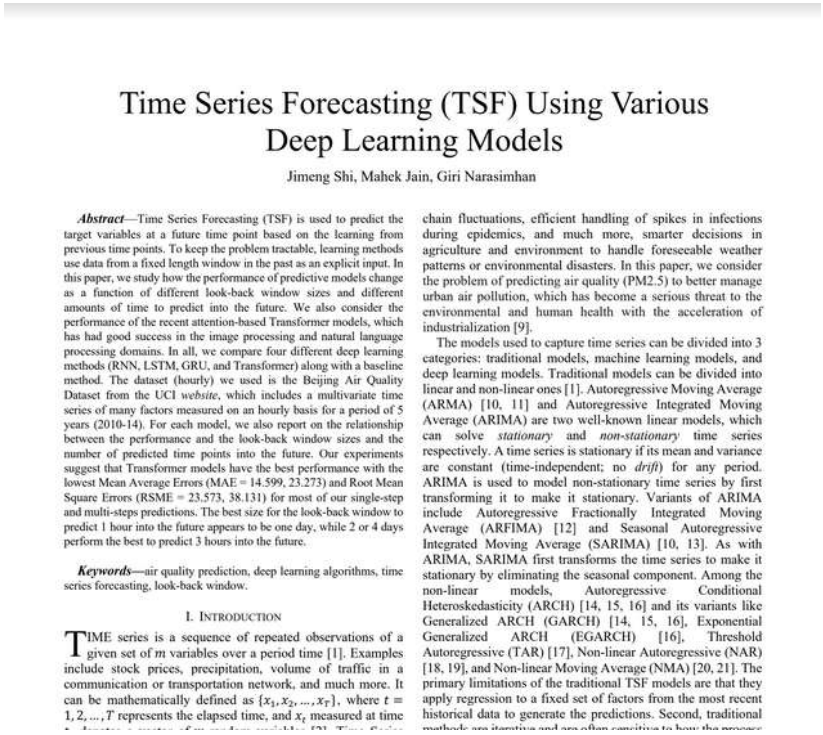
Chuỗi có tính tĩnh

Ứng dụng

- Vì bản chất Time Series là một chuỗi các quan sát lặp lại của một 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.

Về bài báo - Nguồn gốc



Time Series Forecasting (TSF) Using Various Deep Learning Models

Jimeng Shi, Mahek Jain, Giri Narasimhan

***Abstract*—Time Series Forecasting (TSF) is used to predict the target variables at a future time point based on the learning from previous time points. To keep the problem tractable, learning methods use data from a fixed length window in the past as an explicit input. In this paper, we study how the performance of predictive models change as a function of different look-back window sizes and different amounts of time to predict into the future. We also consider the performance of the recent attention-based Transformer models, which has had good success in the image processing and natural language processing domains. In all, we compare four different deep learning methods (RNN, LSTM, GRU, and Transformer) along with a baseline method. The dataset (hourly) we used is the Beijing Air Quality Dataset from the UCI *website*, which includes a multivariate time series of many factors measured on an hourly basis for a period of 5 years (2010-14). For each model, we also report on the relationship between the performance and the look-back window sizes and the number of predicted time points into the future. Our experiments suggest that Transformer models have the best performance with the lowest Mean Average Errors (MAE = 14.599, 23.273) and Root Mean Square Errors (RSME = 23.573, 38.131) for most of our single-step and multi-steps predictions. The best size for the look-back window to predict 1 hour into the future appears to be one day, while 2 or 4 days perform the best to predict 3 hours into the future.**

***Keywords*—air quality prediction, deep learning algorithms, time series forecasting, look-back window.**

I. Introduction

TIME series is a sequence of repeated observations of a given set of m variables over a period time [1]. Examples include stock prices, precipitation, volume of traffic in a communication or transportation network, and much more. It can be mathematically defined as $\{x_1, x_2, \ldots, x_T\}$, where $t = 1, 2, \ldots, T$ represents the elapsed time, and x_t measured at time

chain fluctuations, efficient handling of spikes in infections during epidemics, and much more, smarter decisions in agriculture and environment to handle foreseeable weather patterns or environmental disasters. In this paper, we consider the problem of predicting air quality (PM2.5) to better manage urban air pollution, which has become a serious threat to the environmental and human health with the acceleration of industrialization [9].

The models used to capture time series can be divided into 3 categories: traditional models, machine learning models, and deep learning models. Traditional models can be divided into linear and non-linear ones [1]. Autoregressive Moving Average (ARMA) [10, 11] and Autoregressive Integrated Moving Average (ARIMA) are two well-known linear models, which can solve *stationary* and *non-stationary* time series respectively. A time series is stationary if its mean and variance are constant (time-independent; no *drift*) for any period. ARIMA is used to model non-stationary time series by first transforming it to make it stationary. Variants of ARIMA include Autoregressive Fractionally Integrated Moving Average (ARFIMA) [12] and Seasonal Autoregressive Integrated Moving Average (SARIMA) [10, 13]. As with ARIMA, SARIMA first transforms the time series to make it stationary by eliminating the seasonal component. Among the non-linear models, Autoregressive Conditional Heteroskedasticity (ARCH) [14, 15, 16] and its variants like Generalized ARCH (GARCH) [14, 15, 16], Exponential Generalized ARCH (EGARCH) [16], Threshold Autoregressive (TAR) [17], Non-linear Autoregressive (NAR) [18, 19], and Non-linear Moving Average (NMA) [20, 21]. The primary limitations of the traditional TSF models are that they apply regression to a fixed set of factors from the most recent historical data to generate the predictions. Second, traditional

- Tác giả: Jimeng Shi, Mahek Jain, Giri Narasimhan - Florida International University
- Ngày đăng: 23/04/2022
- Nhà xuất bản: eprint arXiv:2204.11115
- Bipcode: 2022arXiv220411115S
- Keywords: Computer Science - Machine Learning

Về bài báo - Tóm tắt

- Dự đoán chất lượng không khí ở Bắc Kinh thông qua dữ liệu chuỗi thời gian về tình trạng môi trường ở thành phố này với 7 đặc trưng khác nhau.
- Bài báo đã áp dụng qua 4 thuật toán để dự đoán và so sánh kết quả với nhau để tìm ra kết quả 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.

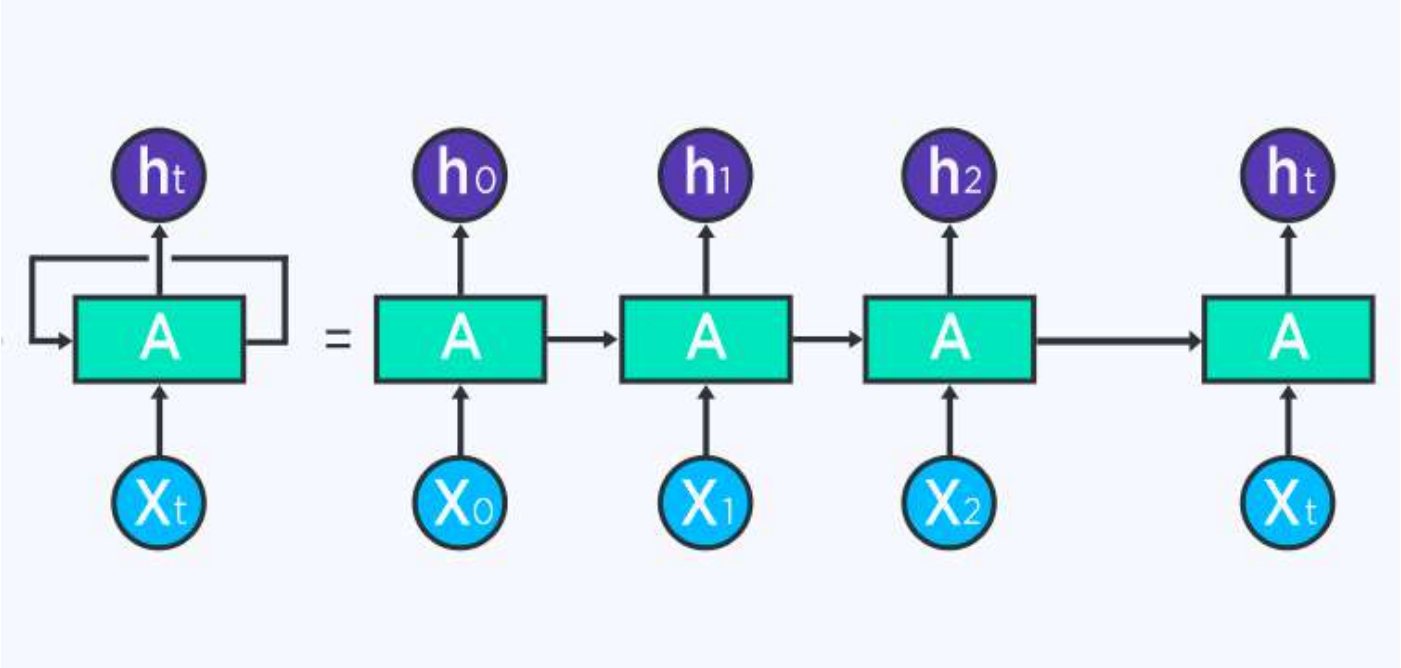
III.

Recurrent Neural Network

Khái niệm

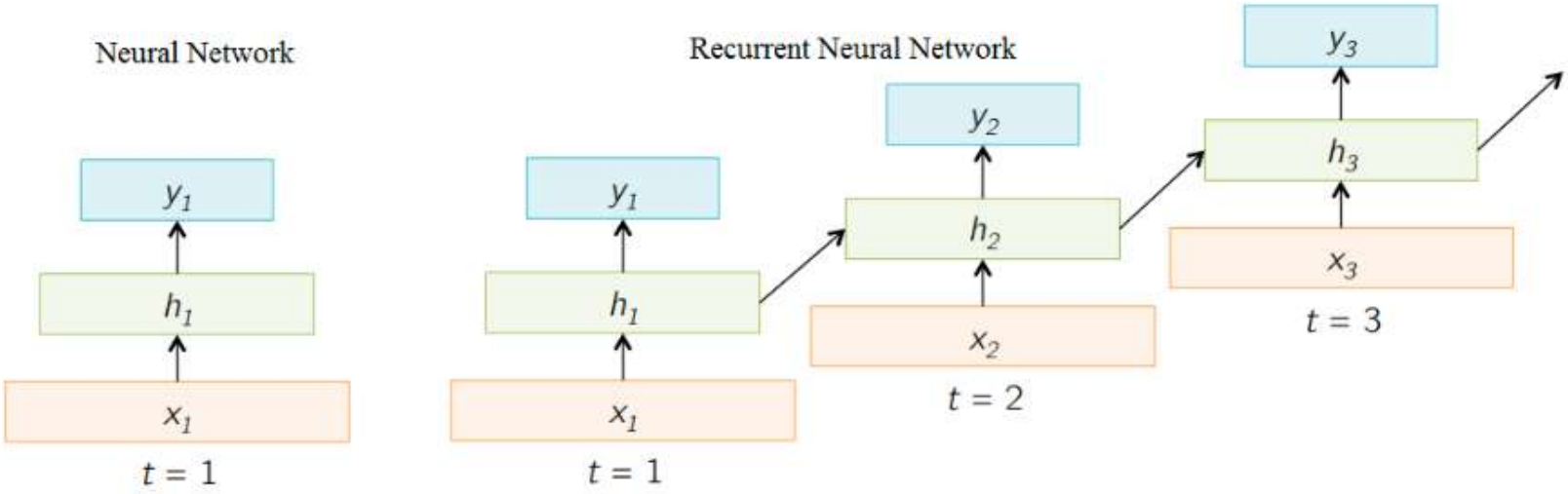
- Một thuật toán trong Deep Neural Network, phát triển từ Neural Network.
- Áp dụng cho Apple Siri và Google voice search.
- Một thuật toán ghi nhớ đầu vào của nó.
- Phù hợp với các bài toán học máy liên quan đến dữ liệu tuần tự.

Khái niệm



Trực quan mô hình Recurrent Neural Network

Khái niệm



Sự khác nhau của Recurrent Neural Network so với Neural Network

Nguyên lý

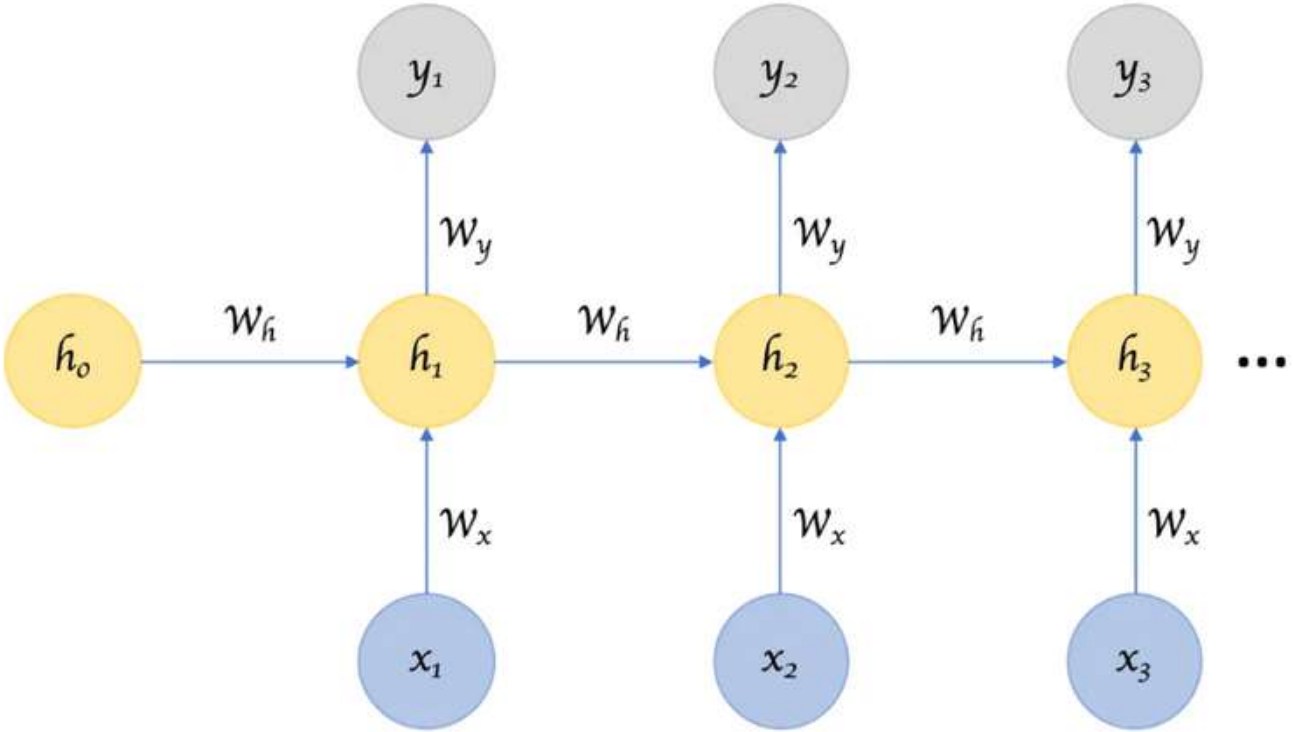
- Bản chất mô hình RNN là trạng thái phía sau sẽ phụ thuộc vào trạng thái trước đó.

- Tại trạng thái t bất kì, nó sẽ phụ thuộc vào input tại thời điểm t và trạng thái trước đó t – 1.

- Mô hình RNN sẽ cần thêm một mối liên kết để lưu lại trạng thái phía trước.
  - Neural Network: input → hidden → output
  - RNN: (input + prev_input) → hidden → output

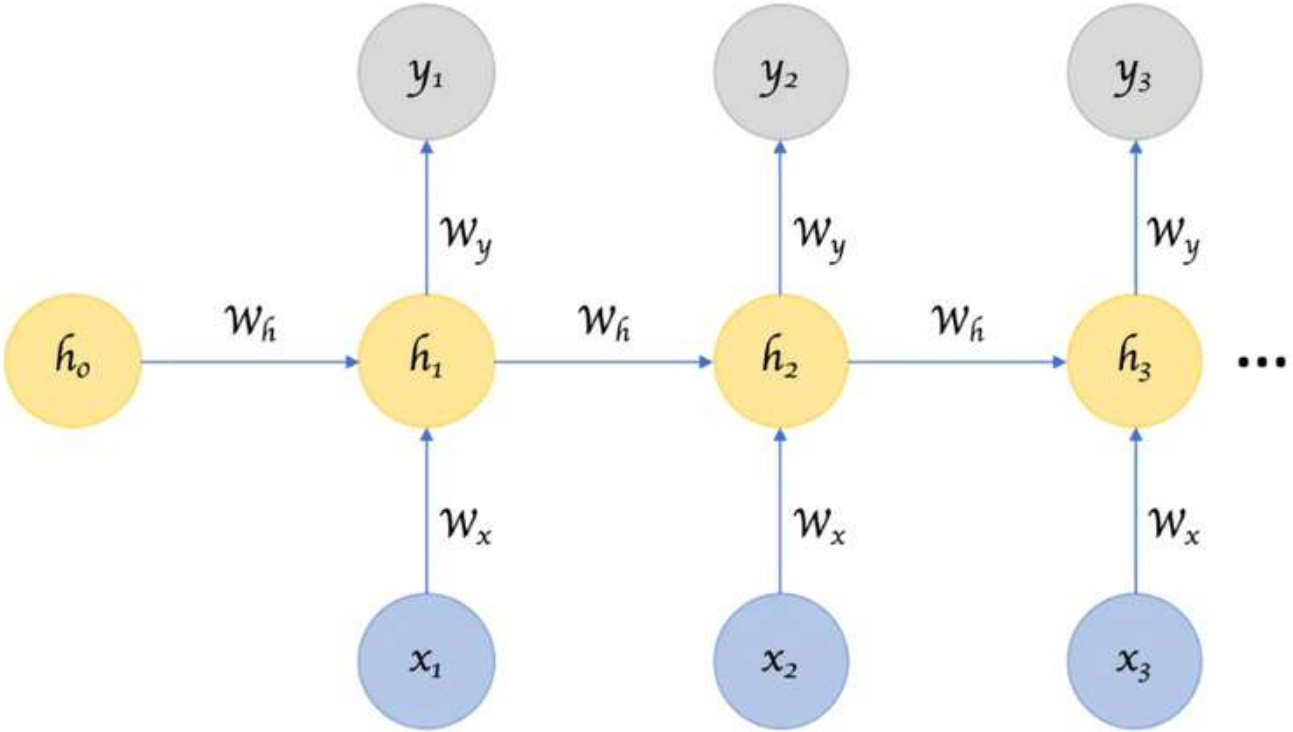
- RNN cũng điều chỉnh trọng số thông qua gradient descent bằng lan truyền ngược (back propagation) theo thời gian.

Nguyên lý



Nguyên lý của mô hình Recurrent Neural Network

Nguyên lý



$$h(t) = f(Wx * x(t) + Wh * h(t-1) + bias)$$

$$y(t) = g(Wy * h(t) + bias)$$

Nguyên lý

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

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.

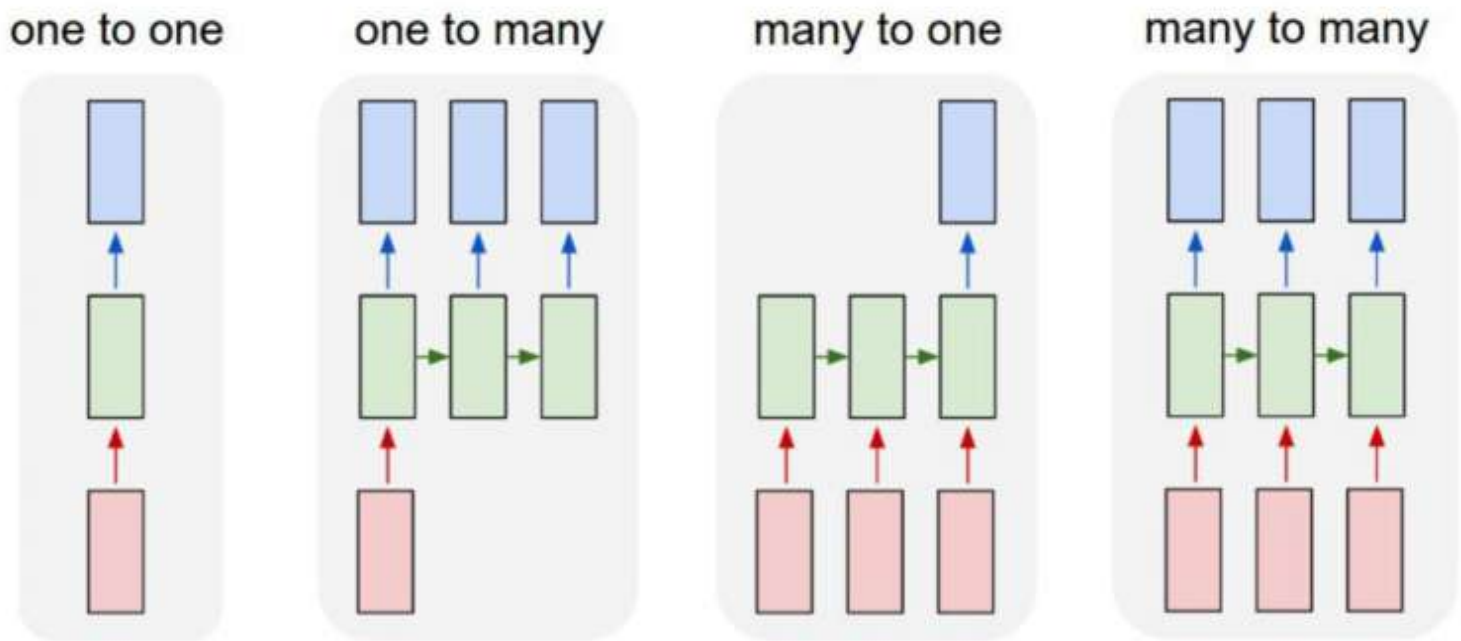
Nguyên lý

RNN được chia làm 4 loại:

- 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

Nguyên lý

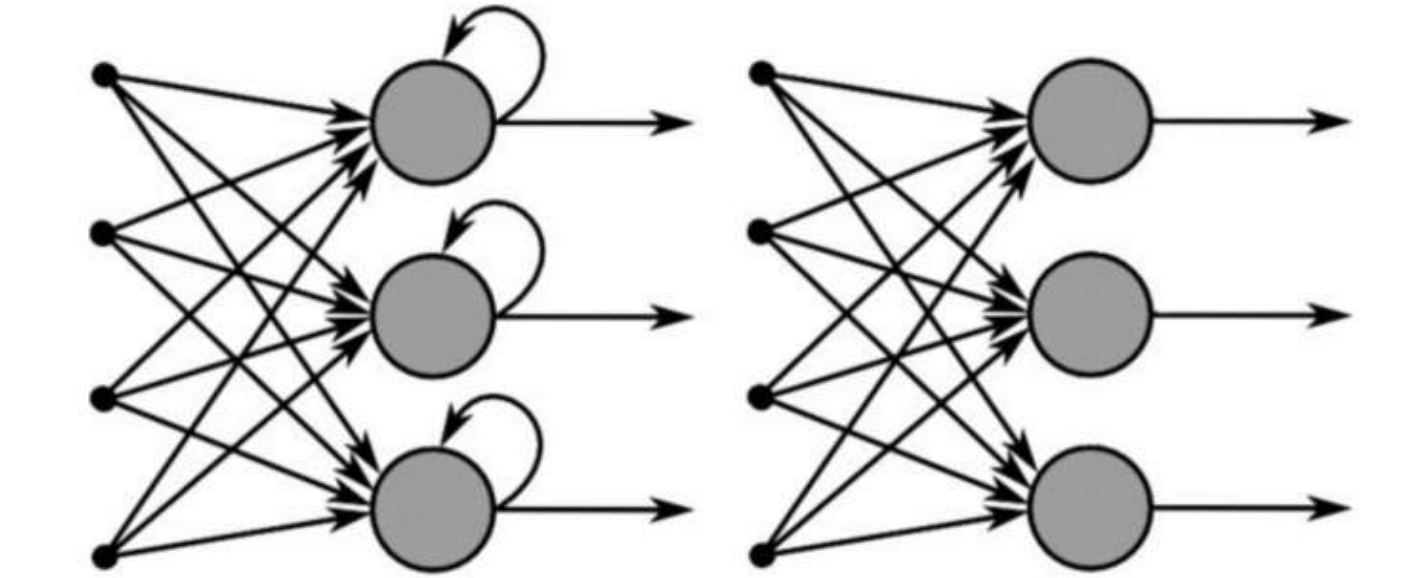
RNN được chia làm 4 loại:



Ứng dụng

- Xử lý ngôn ngữ tự nhiên: dự đoán từ kế tiếp
- Time Series: dự đoán thời tiết, giá vàng, chứng khoán, ...

RNN và Feed-Forward Neural Network



Recurrent Neural Network

Feed-Forward Neural Network

RNN và Feed-Forward Neural Network

RNN	Feed-Forward Neural Network
Có bộ nhớ để ghi nhớ đầu vào: input + prev -> 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

Hạn chế

- Tốc độ thực thi: tiến trình làm việc tuần tự không song song.
- Thực hiện theo 1 chiều cho RNN nguyên bản.
- Vanishing gradient (mất mát đạo hàm): RNN có xu hướng "quên" các đầu vào trước đó

Hạn chế

Loss function & back propagation

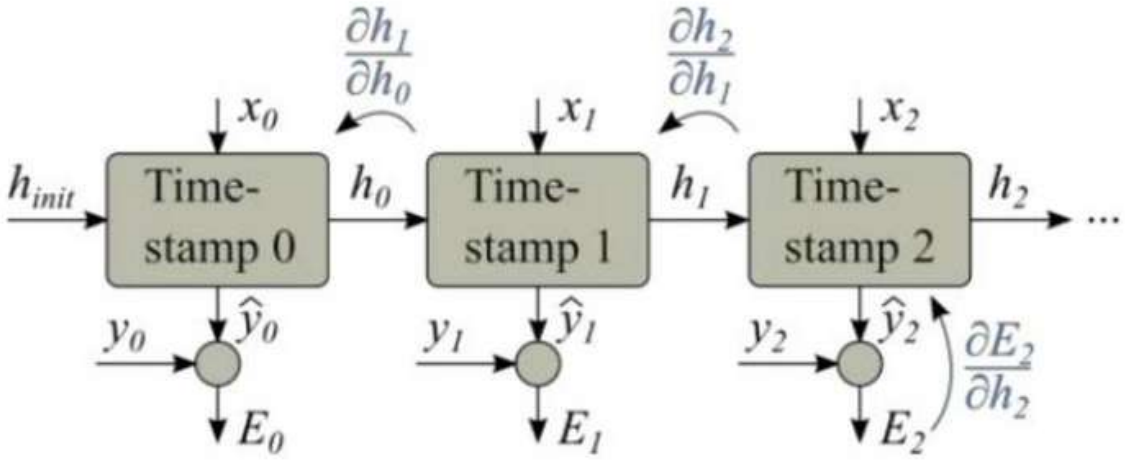
Ta cần chú ý đến 3 giá trị h, Wx và Wy để tính độ lỗi và cập nhật trọng số theo Gradient Descent:

- 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.

Hạn chế

Loss function & back propagation

Ta xét sự lan truyền ngược của từng trạng thái h



Sơ đồ lan truyền ngược của RNN

Hạn chế

Loss function & back propagation

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

Đạo hàm độ lỗi ở thời điểm bất kỳ

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

Sử dụng chain rule: $$\frac{\partial E_i}{\partial W} = \sum_{k=1}^{i} \frac{\partial E_i}{\partial y_i} \frac{\partial y_i}{\partial h_i} \frac{\partial h_i}{\partial h_k} \frac{\partial h_k}{\partial W}$$

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

Hạn chế

Vanishing Gradients

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.

Hạn chế

Vanishing Gradients

- 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 đệ quy đến khi về trạng thái đầu tiên.

- Nếu mạng RNN càng mở rộng và những yếu tố này có giá trị nhỏ hơn 1, nó sẽ gây ra tình trạng mất mát đạo hàm.

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

Hạn chế

Vanishing Gradients

- Vanishing gradient sẽ xảy ra nếu như bộ dữ liệu có đủ nhiều trạng thái, dẫn đến việc đạo hàm độ lỗi khi lan truyền ngược sẽ gần như bằng 0.

- Khi đó mô hình sẽ không còn cập nhật trọng số cho Wh.

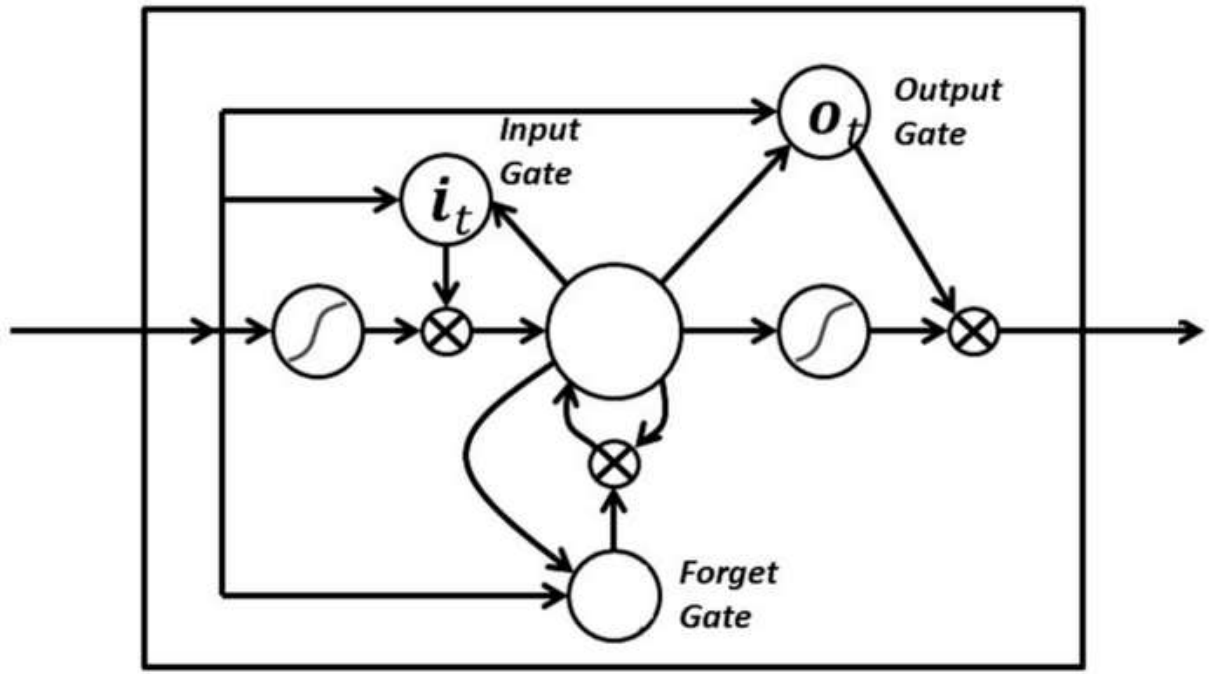
- Vì vậy, RNN thuần túy không phù hợp cho các bộ dữ liệu TimeSeries có tính phụ thuộc xa.

Long Short-Term Memory (LSTM)

- Phiên bản nâng cấp 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, 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.

Long Short-Term Memory (LSTM)

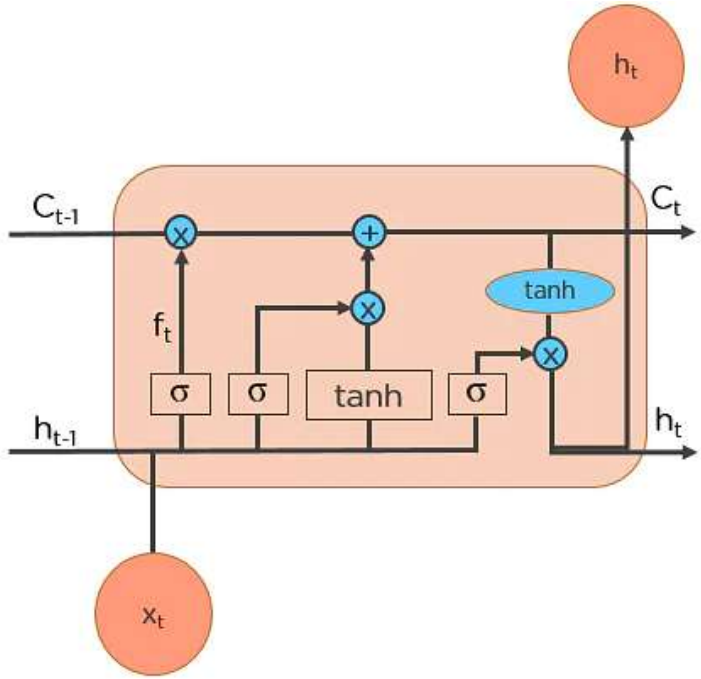


Nguyên tắc hoạt động của LSTM

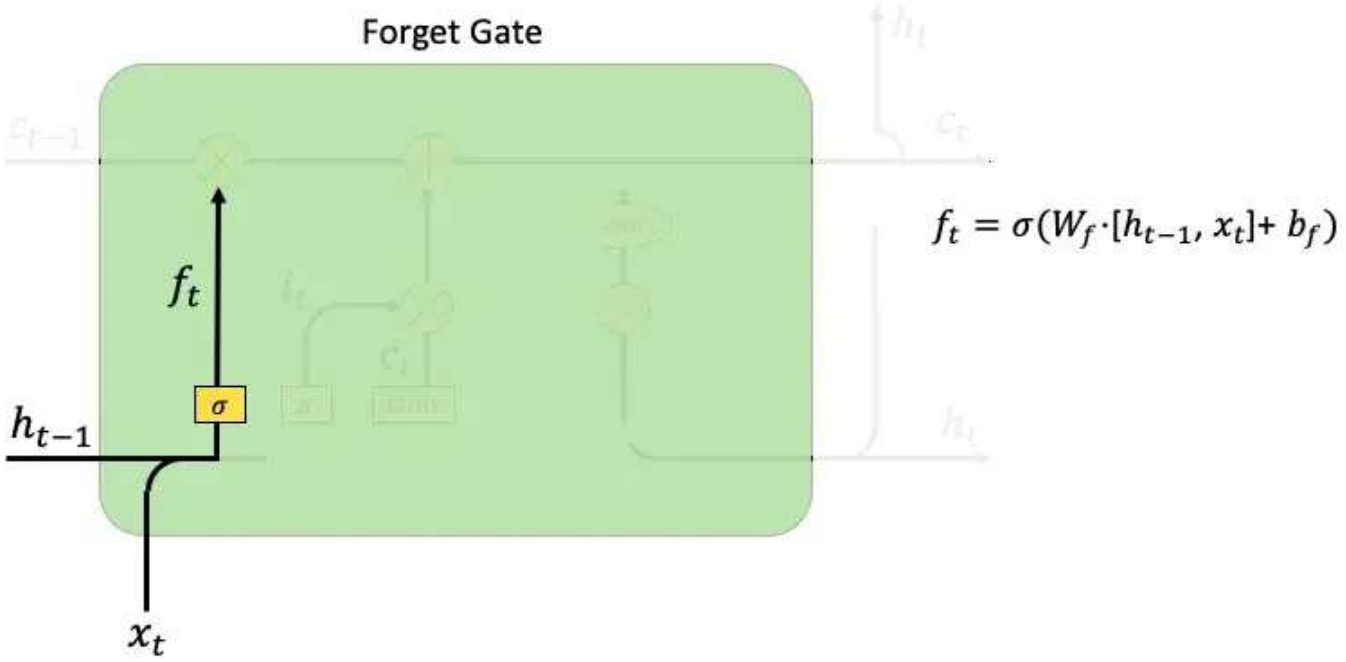
Long Short-Term Memory (LSTM)

LSTM có ba cổng: cổng input, cổng output và cổng quên.

- Cổng input: xác định có cho phép đầu vào mới vào hay không
- Cổng quên: xóa thông tin nếu nó không quan trọng
- Cổng output: tác động đến đầu ra ở mốc thời gian hiện tại

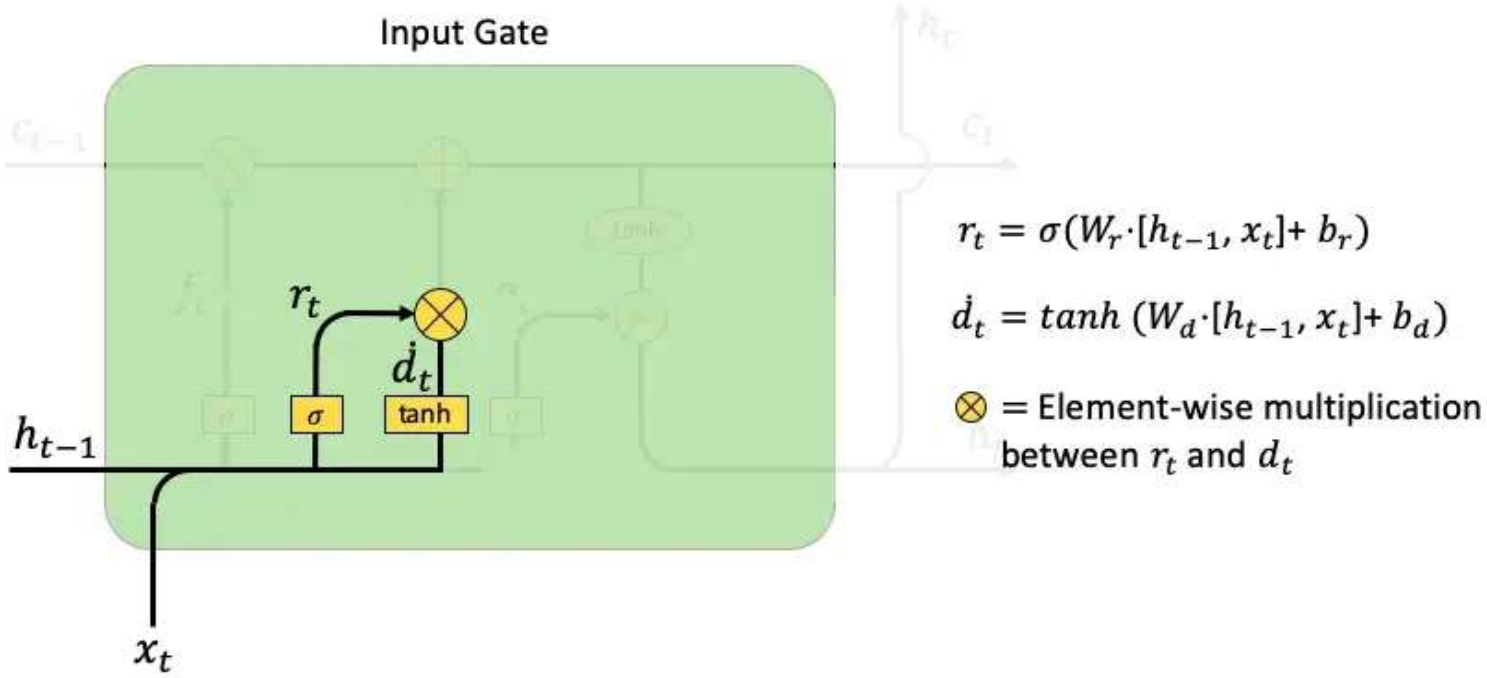


Long Short-Term Memory (LSTM)



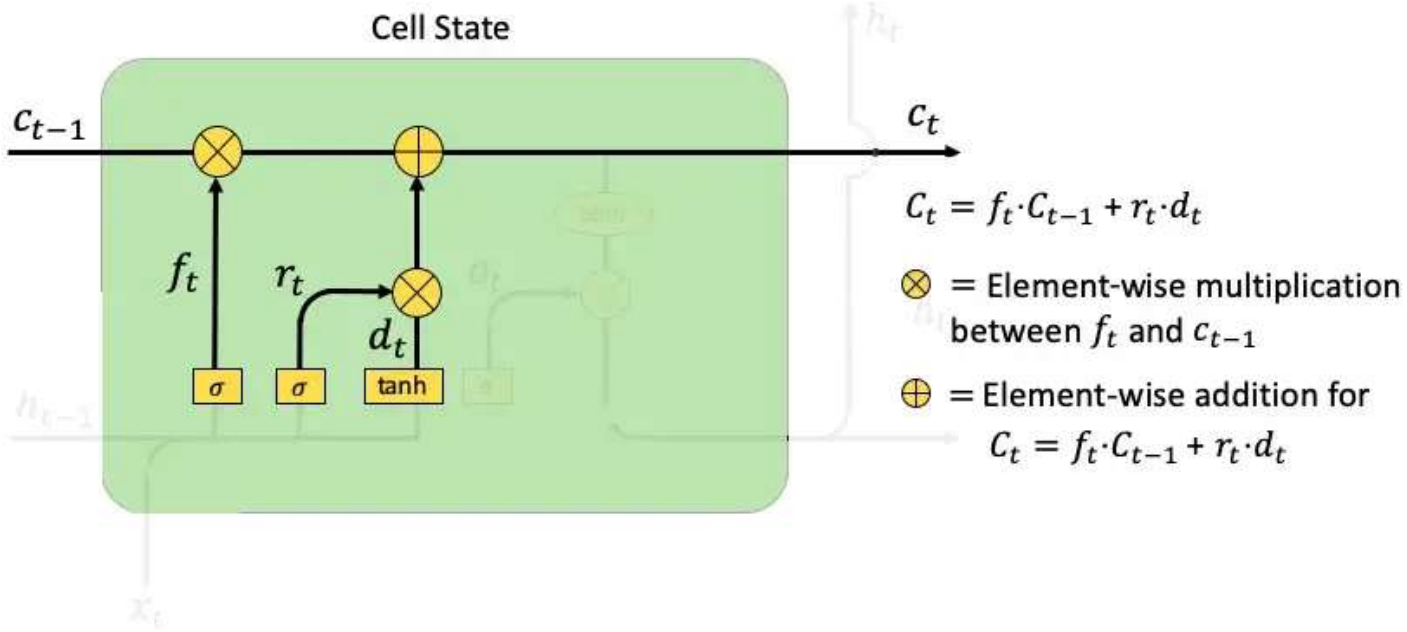
Sơ đồ cổng quên

Long Short-Term Memory (LSTM)



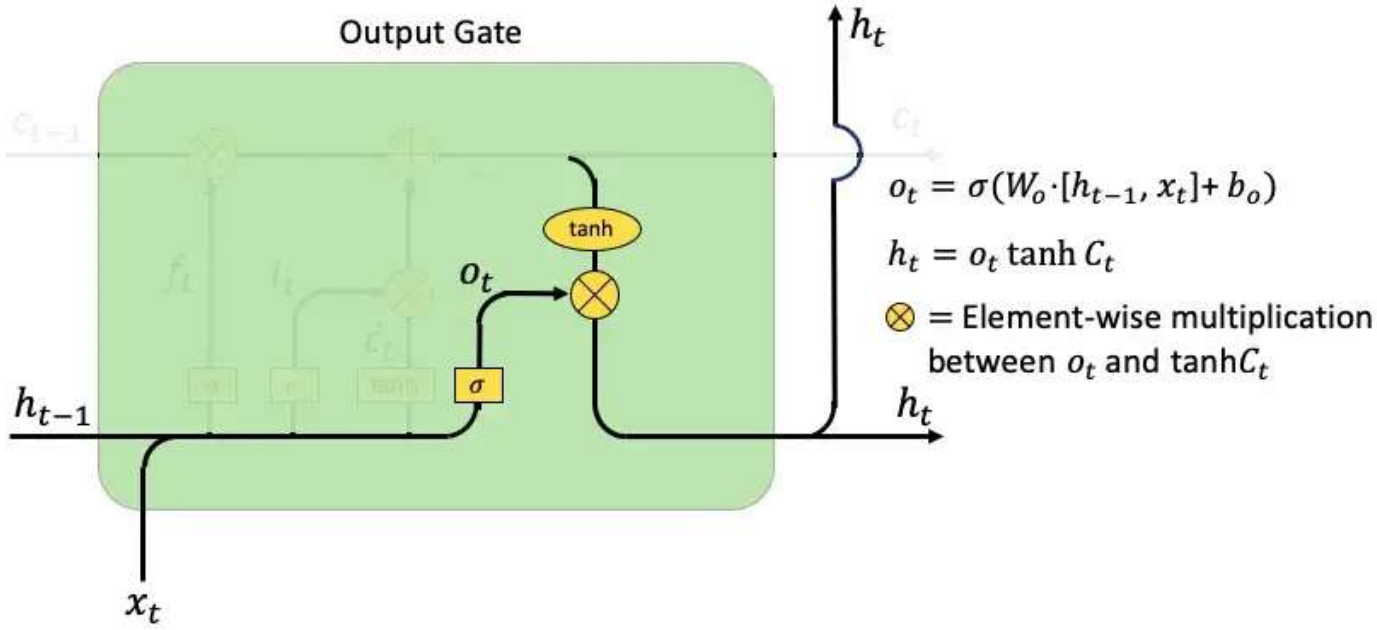
Sơ đồ cổng input

Long Short-Term Memory (LSTM)



Sơ đồ trạng thái bộ nhớ trung tâm (cell)

Long Short-Term Memory (LSTM)



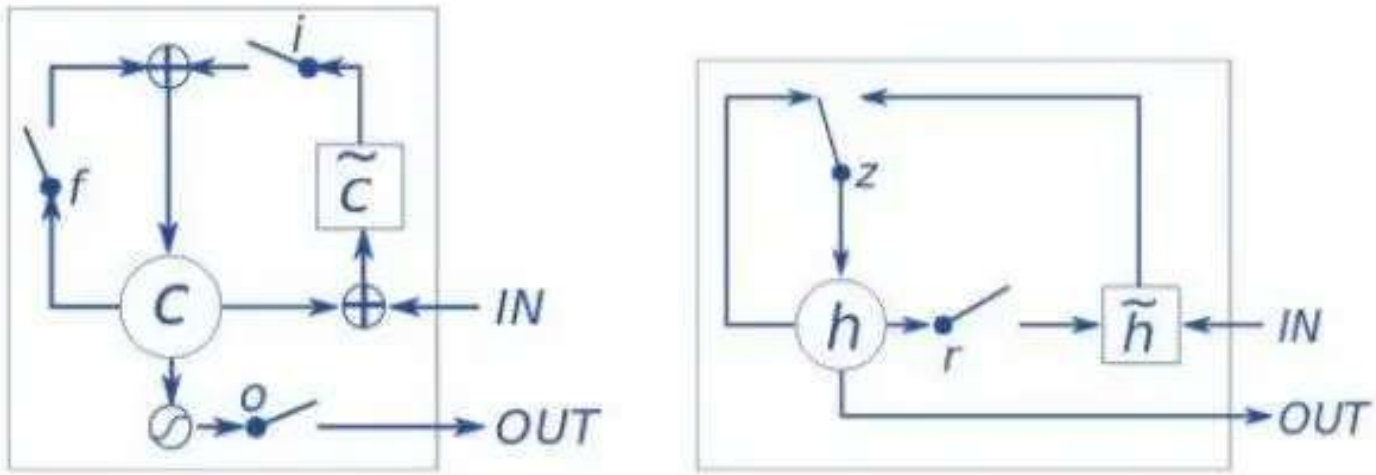
Sơ đồ cổng output

Gated Recurrent Units (GRU)

GRU là 1 phiên bản khác nâng cấp từ RNN, cũng gần tương tự với LSTM:

- LSTM: Kiến trúc xây dựng gồm 1 cell và 3 cổng input gate, output gate và forget gate
- GRU: Kiến trúc xây dựng gồm 1 cell và 2 cổng reset gate và update gate

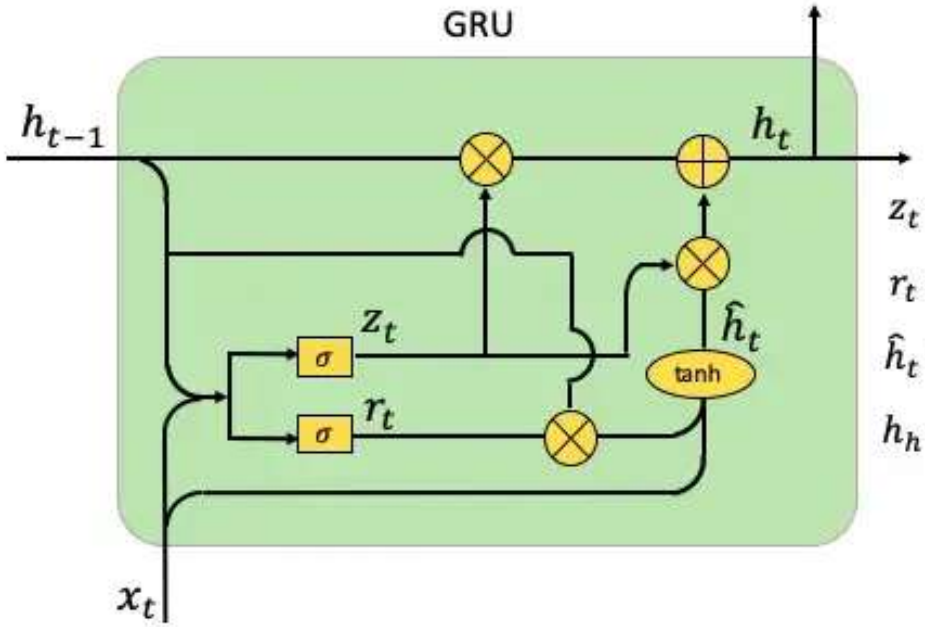
Gated Recurrent Units (GRU)



(a) Long Short-Term Memory (b) Gated Recurrent Unit

Sự khác nhau giữa LSTM và GRU

Gated Recurrent Units (GRU)



$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t] + b_z)$$

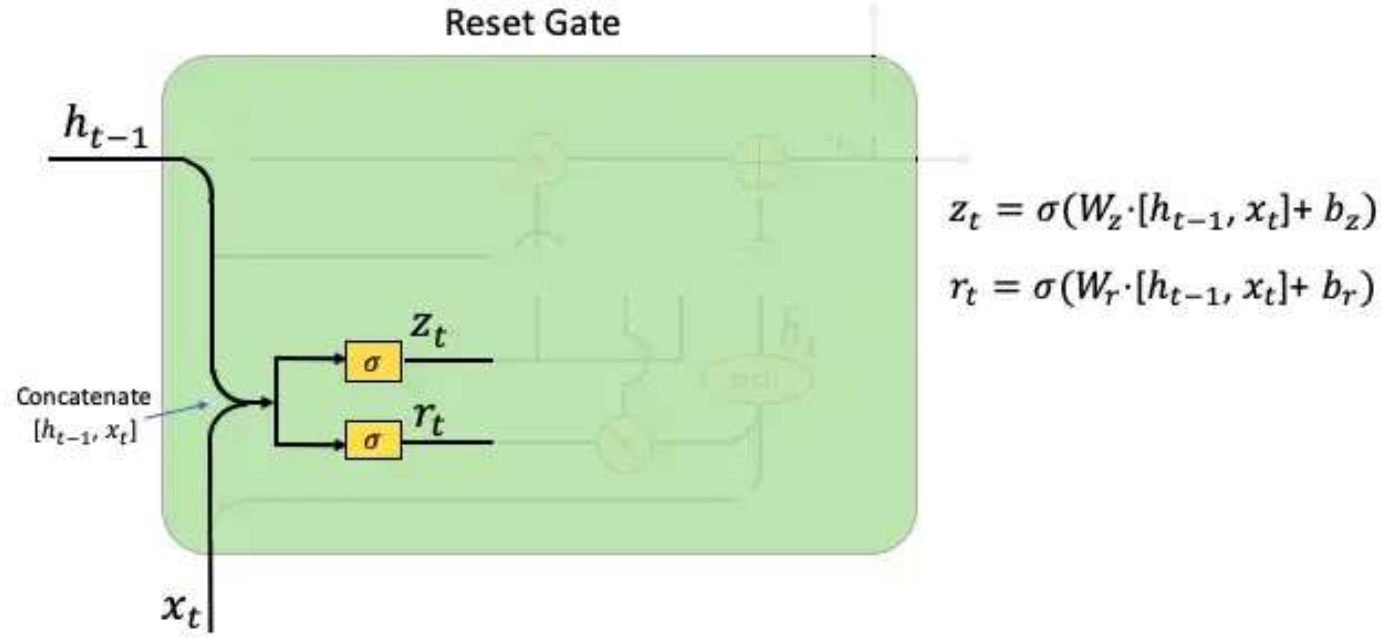
$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t] + b_r)$$

$$\hat{h}_t = tanh\,(W_h \cdot [r_t \cdot h_{t-1}, x_t] + b_h)$$

$$h_h = (1 - z_t) \cdot h_{t-1} + z_t \cdot \hat{h}_t$$

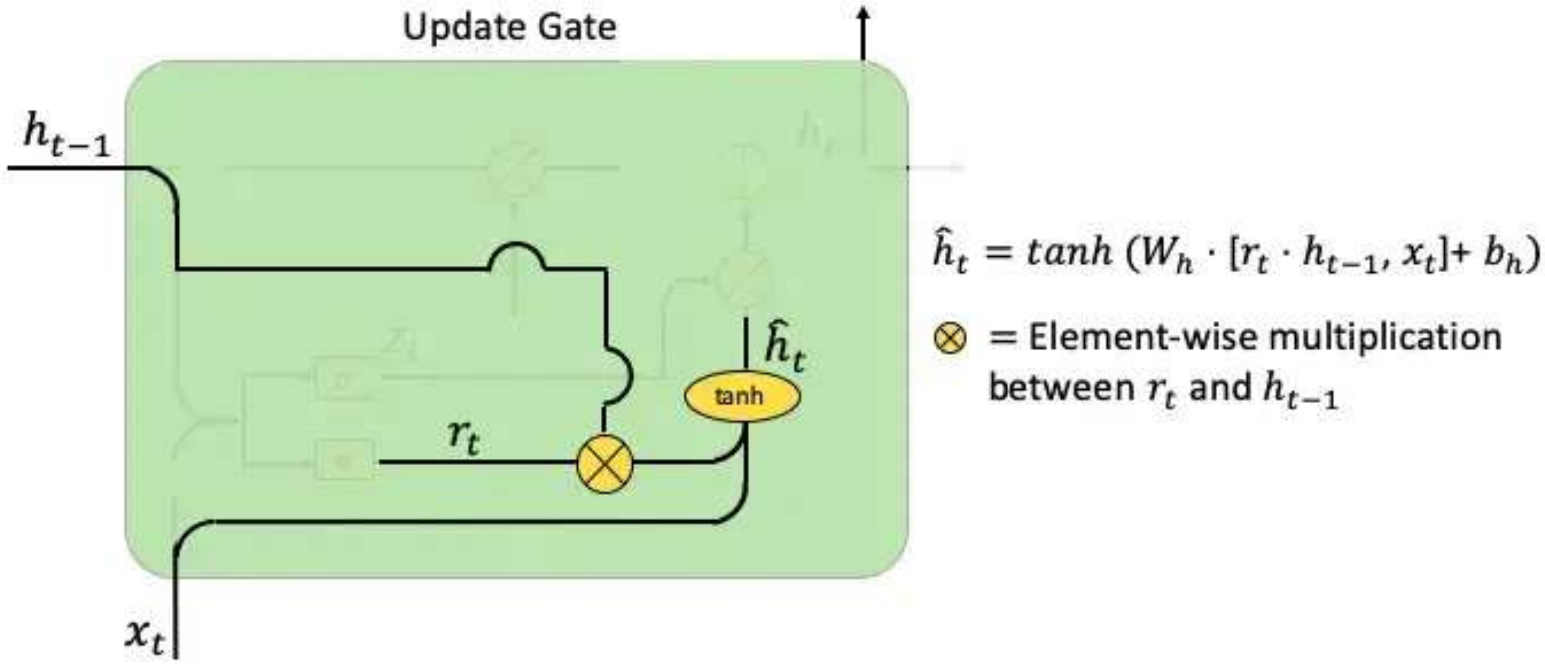
Cấu trúc của GRU

Gated Recurrent Units (GRU)



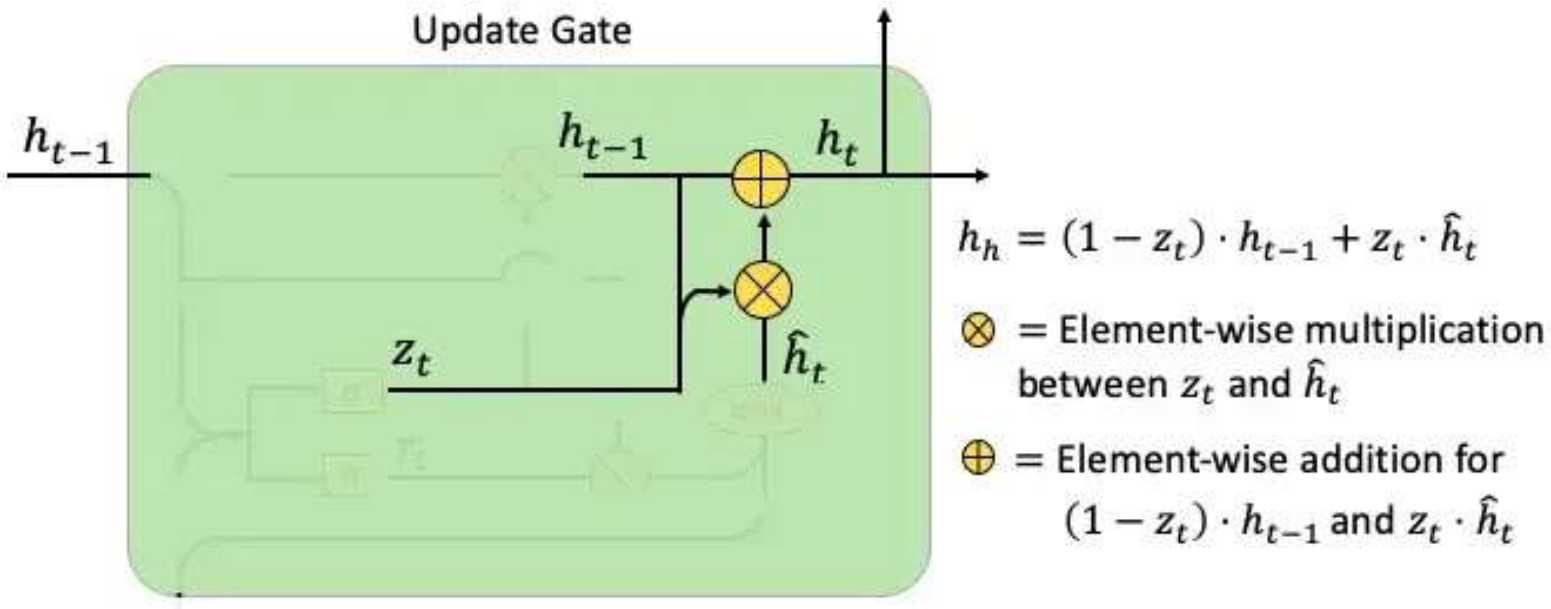
Sơ đồ cổng Reset

Gated Recurrent Units (GRU)



Sơ đồ cổng Update I

Gated Recurrent Units (GRU)



Sơ đồ cổng Update II

IV.

Time Series forecasting with RNN

Phương pháp luận

Cho dữ liệu chuỗi thời gian:

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

xt là vector của *m* đặc trưng đầu vào tại thời điểm *t*.

Phương pháp luận

Mục tiêu dự đoán giá trị đầu ra **y(t+k)** tại thời điểm **t + k**, sử dụng dữ liệu trong lịch sử, đồng nghĩa với việc tập dữ liệu chuỗi thời gian kết thúc tại **t – 1**

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

Phương pháp luận

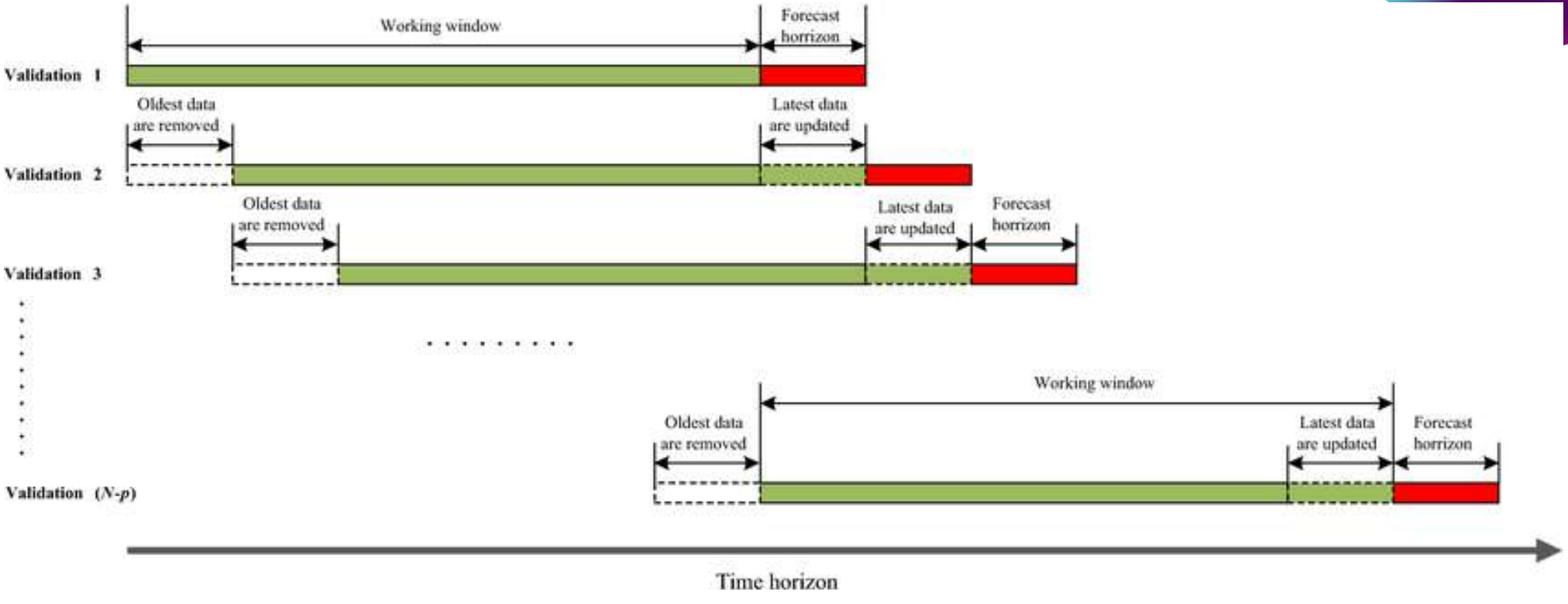
Giá trị đầu ra tại thời điểm t + k:

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

với:

- 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.
- y(t - w) , ..., y(t - 1) là các giá trị đầu ra từ thời điểm t – w tới t – 1
- x(t - w) , ..., x(t - 1) là các vector của m đặc trưng đầu vào từ thời điểm t – w tới t – 1.
- fk 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.

Phương pháp luận



Sử dụng cửa sổ thời gian trượt (sliding window) có chiều dài cố định w để làm các ma trận đầu vào của mô hình có kích thước đồng đều

Validation 1

$$\left[\begin{array}{cccc|c} x_1 & x_2 & \cdots & x_m & x_{m+1} \\ x_2 & x_3 & \cdots & x_{m+1} & x_{m+2} \\ \vdots & \vdots & \ddots & \cdots & \vdots \\ \vdots & \vdots & \ddots & \cdots & \vdots \\ x_{p-m-1} & x_{p-m} & \cdots & x_{p-2} & x_{p-1} \\ x_{p-m} & x_{p-m+1} & \cdots & x_{p-1} & x_p \end{array}\right] \Rightarrow x_{p+1}$$

Validation 2

$$\left[\begin{array}{cccc|c} x_2 & x_3 & \cdots & x_{m+1} & x_{m+2} \\ x_3 & x_4 & \cdots & x_{m+2} & x_{m+3} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{p-m} & x_{p-m+1} & \cdots & x_{p-1} & x_p \\ x_{p-m+1} & x_{p-m+2} & \cdots & x_p & x_{p+1} \end{array}\right] \Rightarrow x_{p+2}$$

Validation 3

$$\left[\begin{array}{cccc|c} x_3 & x_4 & \cdots & x_{m+2} & x_{m+3} \\ x_4 & x_5 & \cdots & x_{m+3} & x_{m+4} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{p-m+1} & x_{p-m+2} & \cdots & x_p & x_{p+1} \\ x_{p-m+2} & x_{p-m+3} & \cdots & x_{p+1} & x_{p+2} \end{array}\right] \Rightarrow x_{p+3}$$

$\vdots$

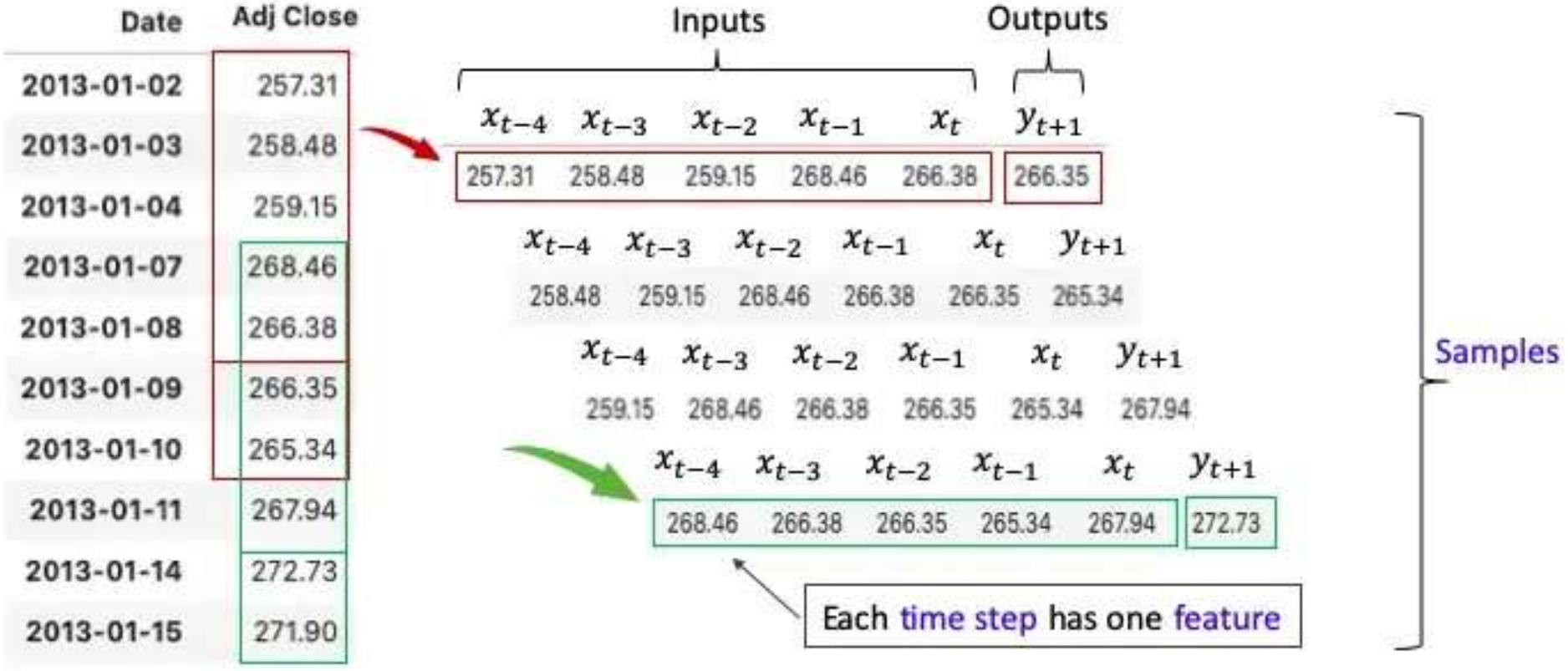
Validation (N-p)

$$\left[\begin{array}{cccc|c} x_{N-p} & x_{N-p+1} & \cdots & x_{N-p+m-1} & x_{N-p+m} \\ x_{N-p+1} & x_{N-p+2} & \cdots & x_{N-p+m} & x_{N-p+m+1} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{N-m-2} & x_{N-m-1} & \cdots & x_{N-3} & x_{N-2} \\ x_{N-m-1} & x_{N-m} & \cdots & x_{N-2} & x_{N-1} \end{array}\right] \Rightarrow x_N$$

Phương pháp luận

Ví dụ: Bài toán dự đoán Stock price

Cấu trúc Many-to-one

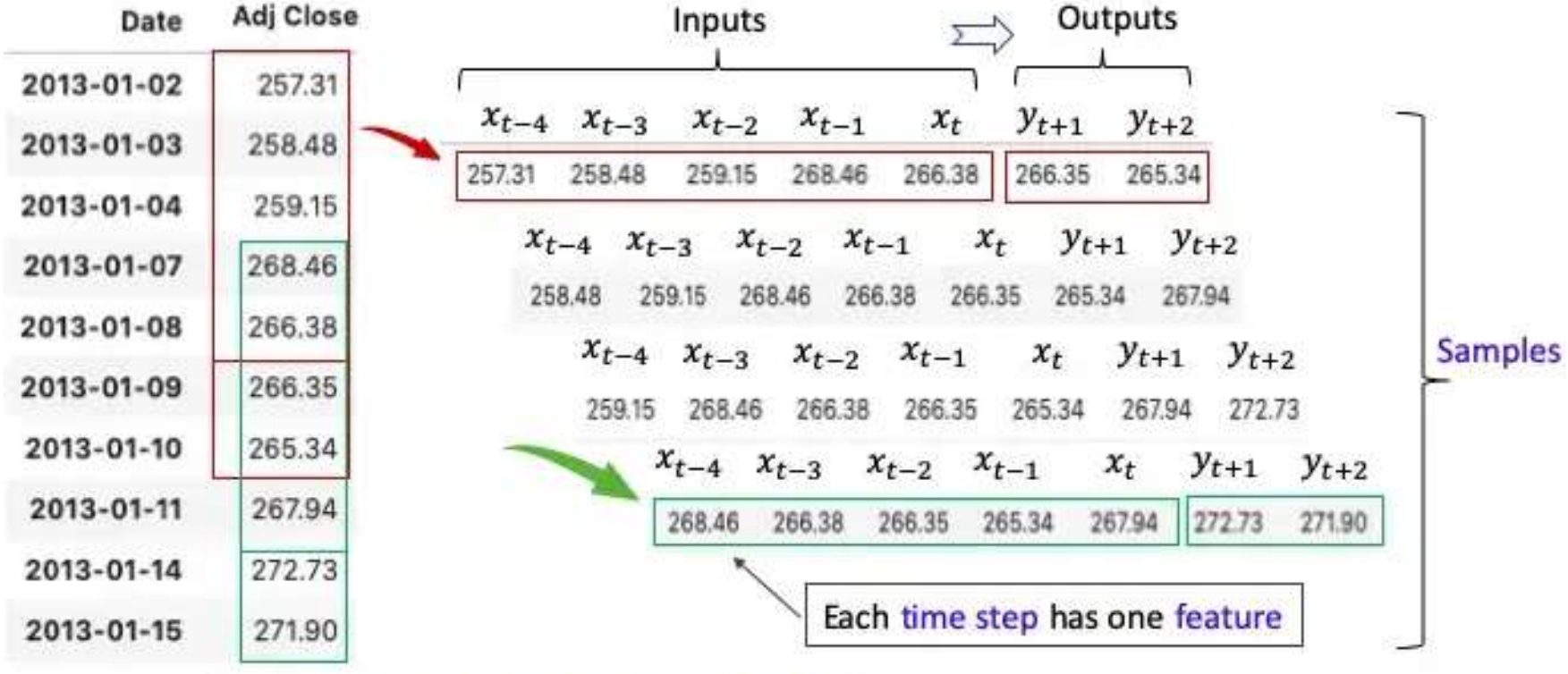


3-D array = [# of samples, # of time steps, # of features]

Phương pháp luận

Ví dụ: Bài toán dự đoán Stock price

Cấu trúc Many-to-many

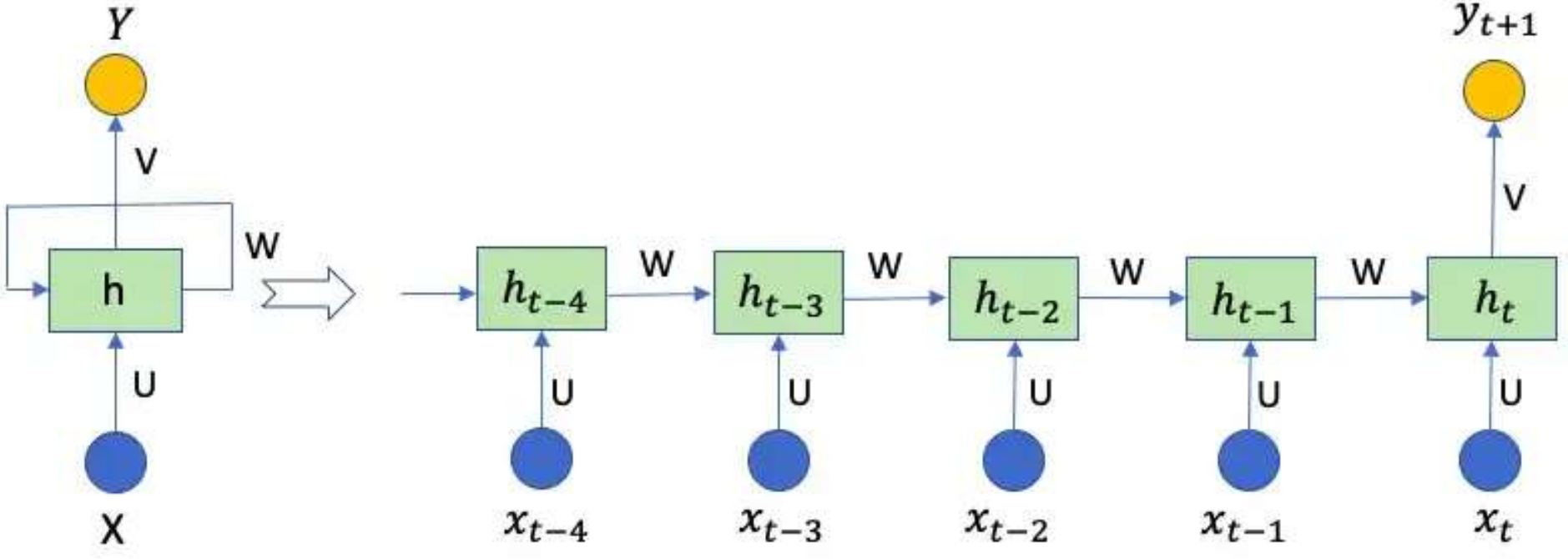


3-D array = [# of samples, # of time steps, # of features]

Phương pháp luận

Giải: Bài toán dự đoán Stock price

Sơ đồ mô hình với cấu trúc Many-to-one:



Xây dựng mô hình

1. Chuẩn hoá 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 hoá ngược đầu ra
5. Tính lỗi hay tính sự khác biệt giữa giá trị dự báo 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

Xây dựng mô hình

1. Chuẩn hoá dữ liệu

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

Xây dựng mô hình

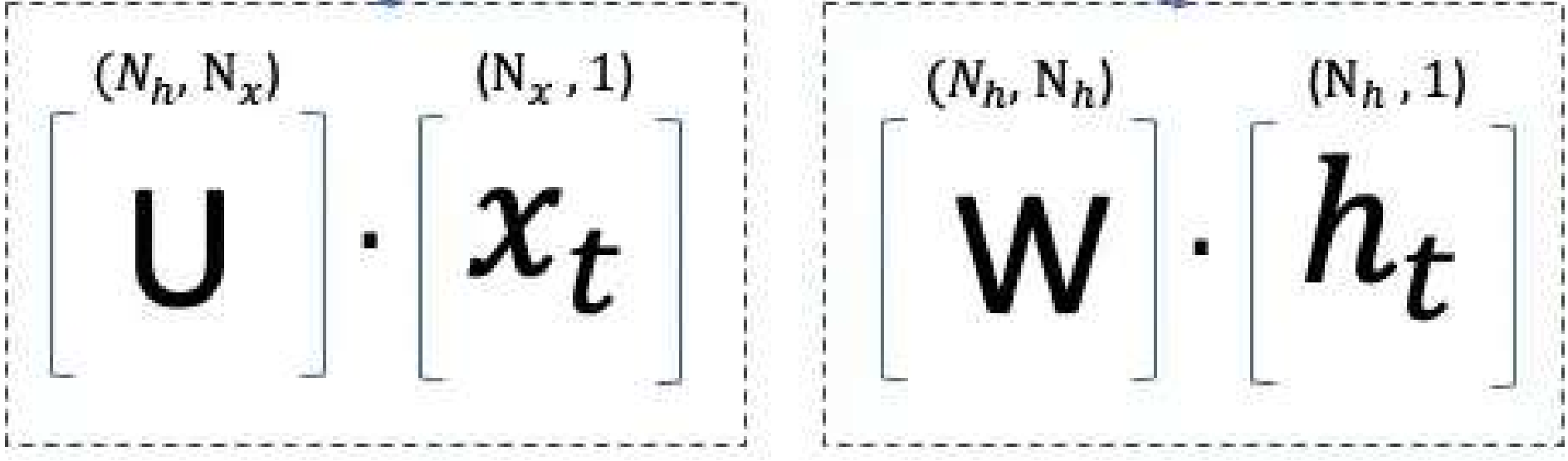
1. Chuẩn hoá dữ liệu

Date	Adj Close	Normalized
2013-01-02	257.31	-1
2013-01-03	258.48	-0.7901
2013-01-04	259.15	-0.67
2013-01-07	268.46	1
2013-01-08	266.38	0.6269
2013-01-09	266.35	

Xây dựng mô hình

2. Khởi tạo trọng số

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



Xây dựng mô hình

2. Khởi tạo trọng số

Bộ trọng số nối giữa hidden và hidden có chiều tuỳ ý => chiều của U, W là (2x1), (2x2) và chiều của V là (1x2). Các giá trị của các ma trận trọng số được khởi tạo ngẫu nhiên là các giá trị thuộc bảng phân phối chuẩn.

$$U = \begin{bmatrix} 0.9804 \\ -0.6080 \end{bmatrix}$$

$$W = \begin{bmatrix} -0.5620 & 0.3111 \\ 0.4655 & -0.2602 \end{bmatrix}$$

$$V = \begin{bmatrix} 0.8718 & -0.5877 \end{bmatrix}$$

Xây dựng mô hình

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:

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

Xây dựng mô hình

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

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:

$$h_5 = f(U.x_5 + W.h_4)$$

$$h_5 = [-0.4986 \quad 0.5255]$$

$$(x_5 = [0.6269])$$

Xây dựng mô hình

3. Tính output

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

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

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

Xây dựng mô hình

4. Chuẩn hóa ngược output:

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

=> *y_hat = 264.5893*

Xây dựng mô hình

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

V.

Conclusion

Kết quả thực nghiệm của bài báo

- **RNN:** luôn cho ra kết quả kém nhất so với các mô hình khác. Đặc biệt ở các giá trị kích thước cửa sổ thời gian trượt w càng lớn, điều này chứng tỏ hiện tượng Vanishing Gradient đã xảy ra khi quá trình trải qua một số lượng hidden state đủ lớn.
- **LSTM:** cho ra kết quả tốt hơn mô hình RNN truyền thống, đặc biệt là ở các giá trị w càng lớn
- **GRU:** tương tự như LSTM, nhưng GRU có xu hướng cho ra kết quả tốt hơn ở các giá trị w không quá lớn.

Về mô hình

- Recurrent Neural Network:

RNN được gọi là recurrent vì mô hình thực hiện cùng một tác vụ cho mọi mẫu dữ liệu, cho ra kết quả từ các tính toán trước đó. Về mặt lý thuyết, RNN có thể xử lí với một mẫu dài gồm nhiều bước thời gian liên tiếp.

Tuy nhiên việc mẫu quá dài có thể dẫn đến "memory" của RNN dễ dàng quên đi dữ liệu của các timesteps xa hơn trong quá khứ trong quá trình truyền dữ liệu giữa các trạng thái.

Về mô hình

- LSTM: LSTM có cấu trúc có thể giữ bộ nhớ trong thời gian dài hơn RNN. Nó giải quyết vấn đề mất mát đạo hàm bằng cách khởi tạo thêm 3 cổng cổng ra, cổng vào và cổng quên. Các cổng bổ sung này có thể kiểm soát tốt hơn độ dốc (gradient), cho phép thông tin được quyết định lưu giữ lại hay quên đi.

- GRU: GRU cũng như LSTM, cấu trúc được thiết kế để giải quyết vấn đề Vanishing Gradient. Tuy nhiên GRU có ít thành phần hơn LSTM, chỉ có 2 cổng là update và reset.

Về bài báo

Những ý nghĩa bài báo đem lại:

1. 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
2. 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.
3. 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.

Thank you for watching

Any questions?