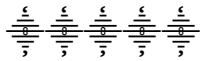
TRƯỜNG ĐẠI HỌC PHENIKAA

KHOA CÔNG NGHỆ THÔNG TIN



PHENIKAA

UNIVERSITY

Báo cáo Đồ án cơ sở

Đề tài :Tìm hiểu bài toán dự báo chuỗi thời gian trong tương lai sử dụng mô hình Neural Network và Deep Learning

GVHD: Nguyễn Văn Thiệu

SVTH: Lê Hoàng Ngọc Tú

Bùi Huy Quyền Anh

Trịnh Gia Khiêm

Mục lục

I Giới thiệu

Đề tài: Tìm hiểu bài toán dự báo chuỗi thời gian trong tương lai sử dụng mô hình Neural Netword và DeepLearing

1: Khái quát về Time Series là gì

Time series là một chuỗi các điểm dữ liệu được đo theo từng khoảng khắc thời gian liền nhau theo một tần suất thời gian thống nhất.

Time Series bao gồm một số đặc trưng:

- Trend: Thành phần này chỉ ra xu hướng tổng quan của dữ liệu theo thời gian: lên hoặc xuống , tăng hoặc giảm
- Seasonality: Thành phần chỉ ra các xu hướng theo mùa vị, chỉ ra các pattern theo tháng, theo quý
- Cycle: Thành phần chu kỳ, nó khác seasonality ở chỗ thành phần này có sự vận động trong khoảng thời gian dài
- Irregular remainder: thành phần nhiễu còn lại sau khi trích xuất hết các thành phần ở trên, chỉ ra các điểm bất thường của các điểm dữ liệu

Dự báo chuỗi thời gian bằng Deep Learning: Việc sử dụng Deep Learning để dự báo chuỗi thời gian đã khắc phục được những hạn chế của Machine Learning với nhiều cách tiếp cận khác. Mạng thần kinh hồi quy (RNN) là kiến trúc cổ điển được sử dụng nhiều nhất cho bài toán Dự báo chuỗi thời gian

2: Neural Network

2.1: Neural NetWork là gi ?

Neural Network là mạng lưới Nơ-ron nhân tạo. Đây là chuỗi thuật toán nhằm tìm kiếm quan hệ trong tập hợp dữ liệu hệ thống dựa theo cách thức hoạt động não bộ con người. Neural Networks thích ứng với mọi điều chỉnh từ đầu vào, cho ra kết quả đầu ra tốt nhất.

2.2: Đặc điểm của Neural Network

Mạng lưới nơ-ron nhân tạo hoạt động như nơ-ron trong não bộ con người.

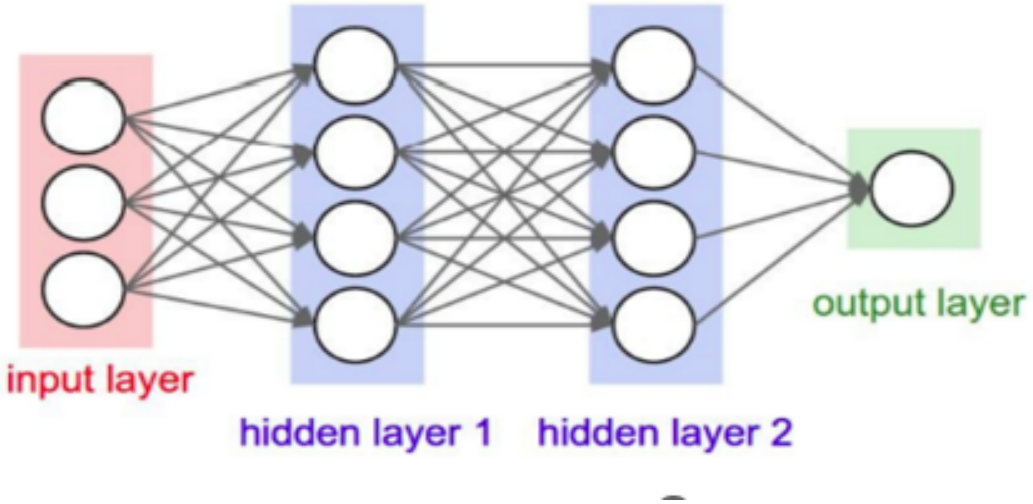
Neural Network tương đồng với những phương pháp thống kê theo đồ thị đường cong hoặc phân tích hồi quy

Mỗi nút là một tập hợp tri giác, cấu tạo tương tự hàm hồi quy đa tuyến tính, được sắp xếp liên kết với nhau.

2.3: Kiến trúc Neural NetWork

Mỗi một mạng lưới Nơ-ron nhân tạo là một Perceptron đa tầng, một Neural Network thường bao gồm 3 kiểu tầng cụ thể như sau:

- Input Layer (tầng đầu vào): Nằm bên trái của hệ thống, bao gồm dữ liệu thông tin đầu vào.
- Output Layer (tầng đầu ra): Nằm bên phải của hệ thống, bao gồm dữ liệu thông tin đầu ra.
- Hidden Layer (tầng ẩn): Nằm ở giữa tầng đầu vào và đầu ra, thể hiện quá trình suy luận và xử lý thông tin của hệ thống.



Hình 1: Cấu trúc Mạng nơ-ron

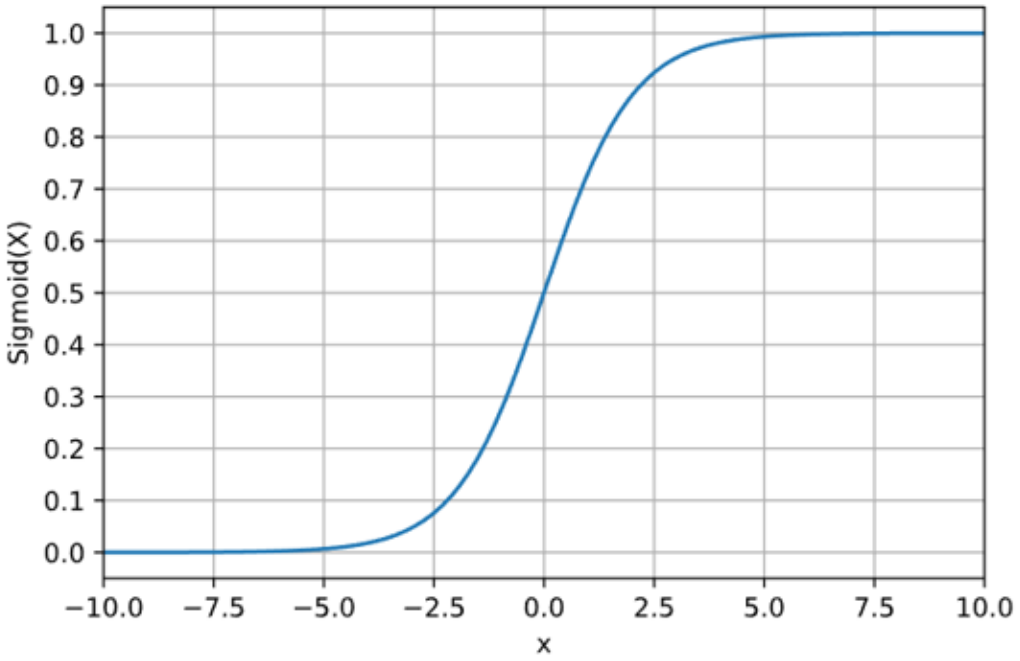
II Mô Hình Neural Network

1: Các hàm kích hoạt

1.1: Hàm Sigmoid

Công thức

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



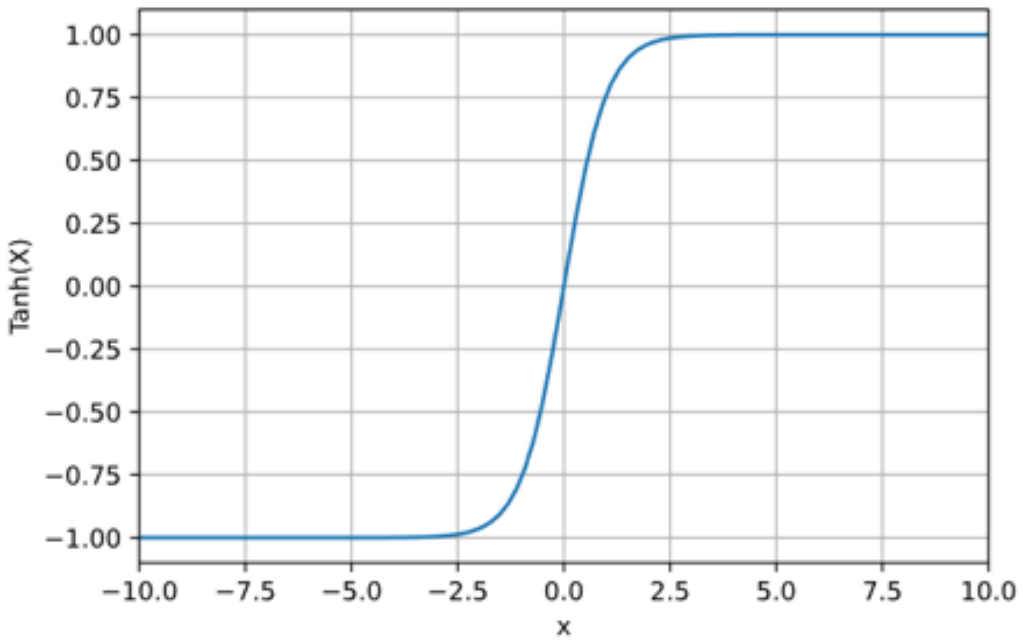
Đồ thị hàm Sigmoid

Hàm Sigmoid nhận đầu vào là một số thực và chuyển thành một giá trị trong khoảng (0;1) (xem đồ thị phía trên). Đầu vào là số thực âm rất nhỏ sẽ cho đầu ra tiệm cận với 0, ngược lại, nếu đầu vào là một số thực dương lớn sẽ cho đầu ra là một số tiệm cận với 1. Trong quá khứ hàm Sigmoid hay được dùng vì có đạo hàm rất đẹp

1.2: Hàm Tanh

Công thức

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



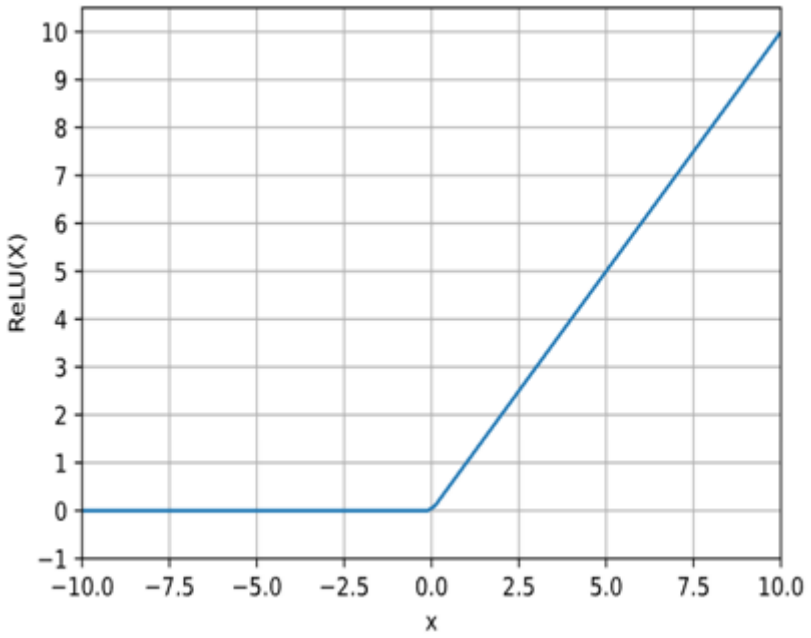
Đồ thị hàm Tanh

Hàm Tanh nhận đầu vào là một số thực và chuyển thành một giá trị trong khoảng (-1; 1). Cũng như Sigmoid, hàm Tanh bị bão hoà ở 2 đầu (gradient thay đổi rất ít ở 2 đầu). Tuy nhiên hàm Tanh lại đối xứng qua 0 nên khắc phục được một nhược điểm của Sigmoid. Hàm tanh còn có thể được biểu diễn bằng hàm sigmoid như sau: **Tanh(x) = 2σ(2x) - 1**

1.3: Hàm ReLu

Công thức

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



Đồ thị hàm ReLu

Hàm ReLU đang được sử dụng khá nhiều trong những năm gần đây khi huấn luyện các mạng neuron. ReLU đơn giản lọc các giá trị < 0. Nhìn vào công thức chúng ta dễ dàng hiểu được cách hoạt động của nó. Một số ưu điểm khá vượt trội của nó so với Sigmoid và Tanh:

(+) Tốc độ hội tụ nhanh hơn hẳn. ReLU có tốc độ hội tụ nhanh gấp 6 lần Tanh. Điều này có thể do ReLU không bị bão hoà ở 2 đầu như Sigmoid và Tanh.

(+) Tính toán nhanh hơn. Tanh và Sigmoid sử dụng hàm exp và công thức phức tạp hơn ReLU rất nhiều do vậy sẽ tốn nhiều chi phí hơn để tính toán.

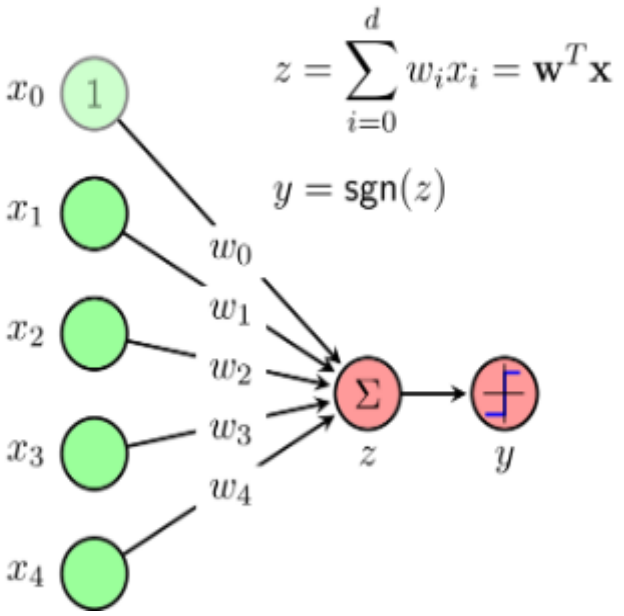
(-) Tuy nhiên ReLU cũng có một nhược điểm: Với các node có giá trị nhỏ hơn 0, qua ReLU activation sẽ thành 0, hiện tượng đấy gọi là “Dying ReLU“. Nếu các node bị chuyển thành 0 thì sẽ không có ý nghĩa với bước linear activation ở lớp tiếp theo và các hệ số tương ứng từ node đấy cũng không được cập nhật với gradient descent. => Leaky ReLU ra đời.

(-) Khi learning rate lớn, các trọng số (weights) có thể thay đổi theo cách làm tất cả neuron dừng việc cập nhật

2: Mô hình MLP (Multi-layer Perceptron)

2.1: Perceptron là gì ?

Perceptron là một thuật toán học máy tuyến tính được sử dụng để học có giám sát (supervised learning) cho các bộ phân loại nhị phân khác nhau. Một perceptron có thể được định nghĩa là một mạng nơ-ron với một lớp duy nhất phân loại dữ liệu tuyến tính.

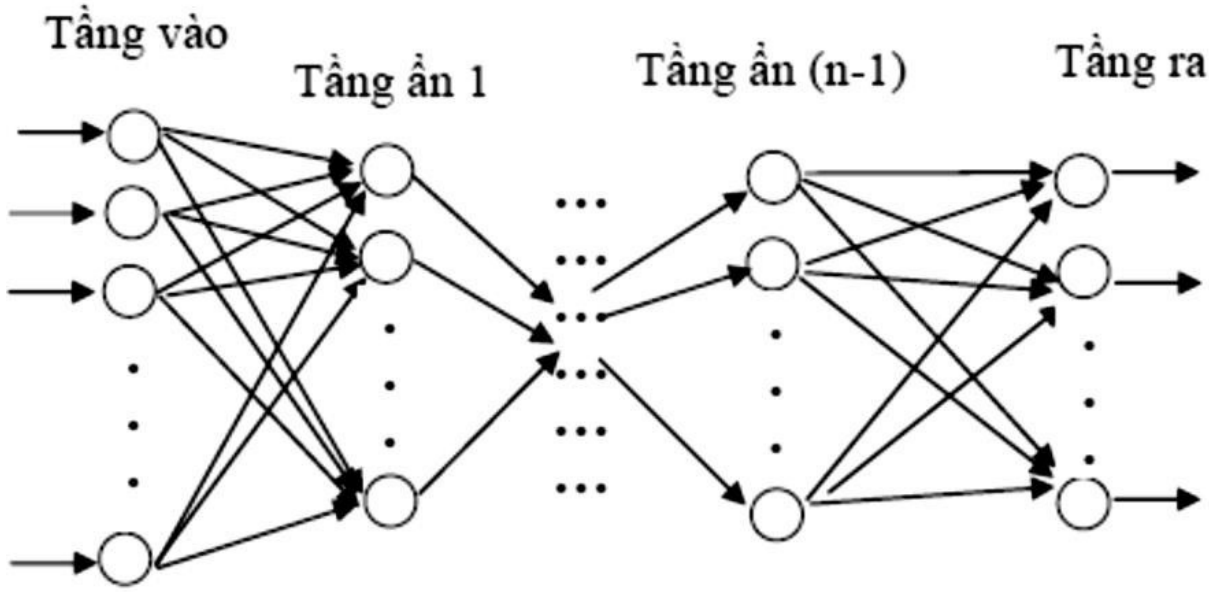


Mô hình đơn Perceptron

2.2: Giới thiệu về MLP (Multi-layer Perceptron)

Là sự kết hợp của nhiều perceptron hay còn gọi là perceptron đa tầng. Là mạng được sử dụng rộng rãi với nhiều tầng truyền thẳng. Một mạng MLP tổng quát là mạng có n (n≥2) tầng (thông thường tầng đầu vào không được tính đến): trong đó gồm một tầng đầu ra (tầng thứ n) và (n-1) tầng ẩn.

2.3 :Cách thức hoạt động của mạng mạng Multi-layer



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

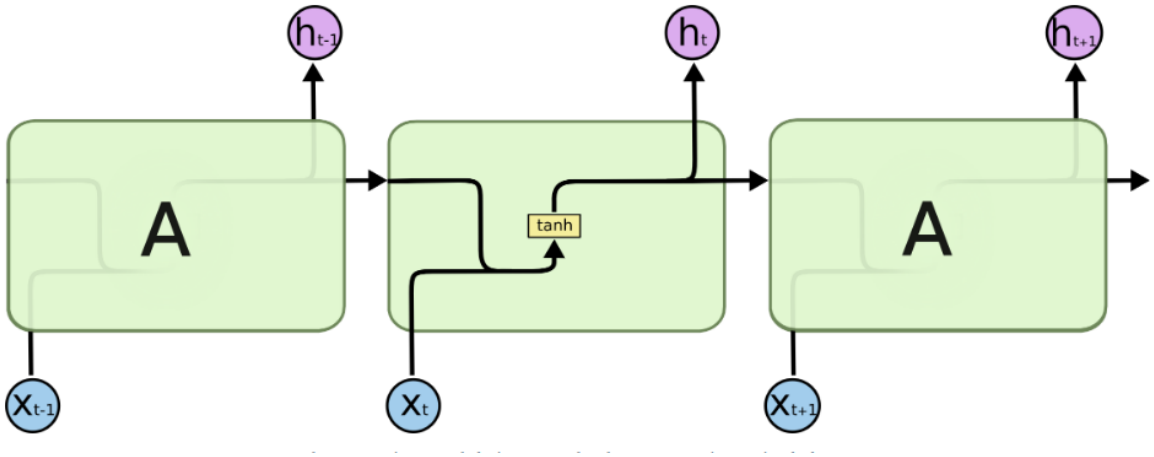
3: Mô hình LSTM (Long Short Term Memory network)

3.1: Giới thiệu về LSTM

Mạng LSTM (Long Short Term Memory network) là một dạng đặc biệt của RNN, có khả năng học được các phụ thuộc xa.

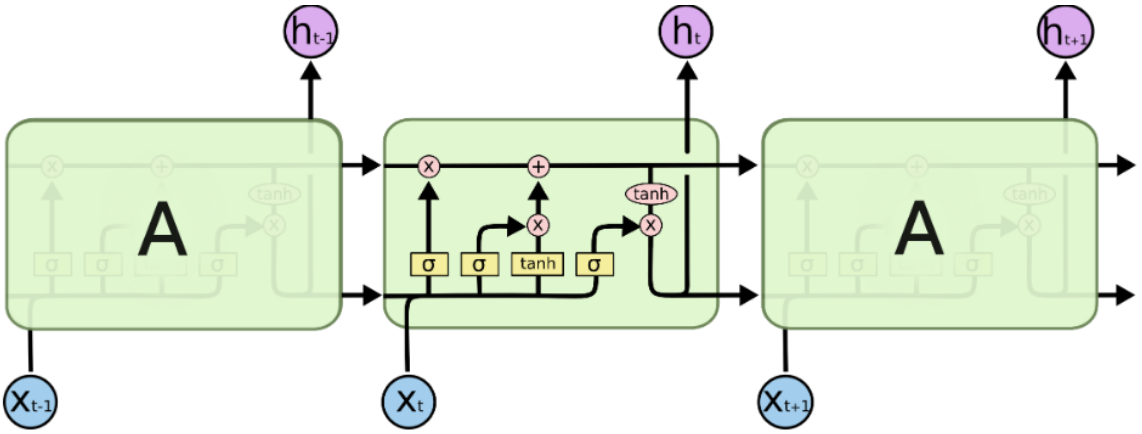
LSTM được thiết kế tránh được vấn đề phụ thuộc xa (long-term dependency). Việc nhớ thông tin trong suốt thời gian dài là đặc tính của LSTM, chứ không cần huấn luyện để có thể nhớ được nó.

Mạng hồi quy đều có dạng chuỗi các modun lặp đị lặp lại của mạng nơ-ron. Với mạng RNN chuẩn, các module có cấu trúc rất đơn giản, thường là một tầng Tanh



Hình 2: Cấu trúc mạng RNN

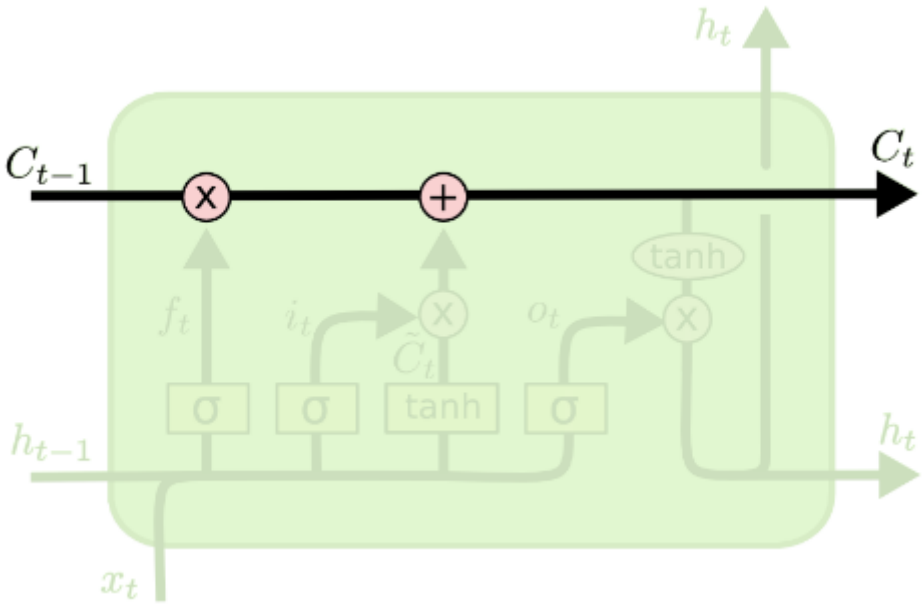
LSTM cũng có kiến trúc dạng chuỗi nhưng mô-đun của nó khác với cấu trúc mạng RNN. Thay vì có 1 tầng thì LSTM có 4 tầng tương tác với nhau 1 cách đặc biệt



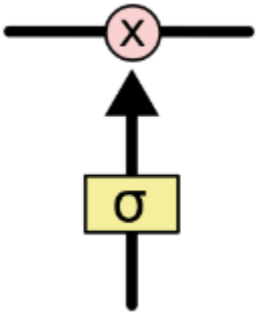
Hình 3: Cấu trúc mạng LSTM

3.2: Cách thức hoạt động mô hình LSTM

Ý tưởng của LSTM là thành phần ô trạng thái (cell state) được thể hiện qua đường chạy ngang qua đỉnh của đồ thị (Hình 4)



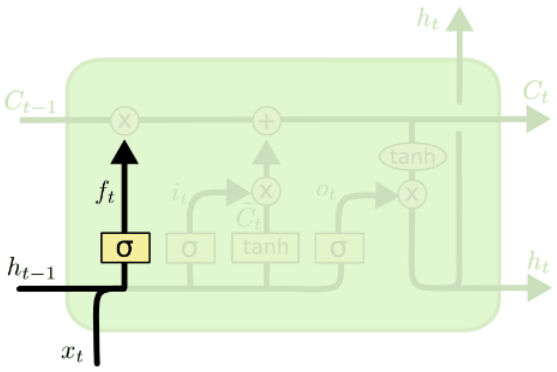
Đường đi của cell state trong LSTM



Cổng của hàm sigmoid trong LSTM

Thứ tự các bước của LSTM

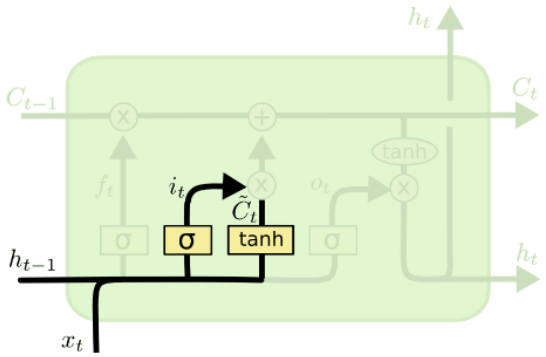
Bước 1: LSTM quyết định thông tin sẽ cho phép đi qua ô trạng thái(cell state). Kiểm soát bởi hàm sigmoid trong Forget gate layer)



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

Forget gate layer

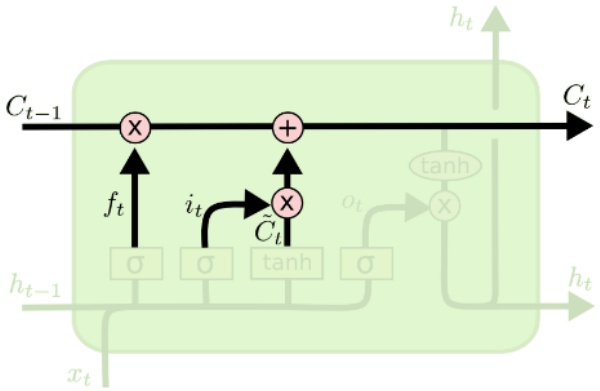
Tiếp quyết định thông tin được lưu trữ trong ô trạng thái (cell state). Gồm 2 phần, một tầng ẩn của hàm sigmoid là tầng input gate layer quyết định giá trị sẽ được cập nhật. Tầng ẩn hàm tanh tạo ra 1 vectơ của một giá mà có thể được thêm vào trạng thái mới. Kết hợp 2 tầng tạo thành 1 cập nhật cho trạng thái



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Cập nhật giá trị trạng thái kết hợp từ input gate layer và tầng ẩn hàm tanh

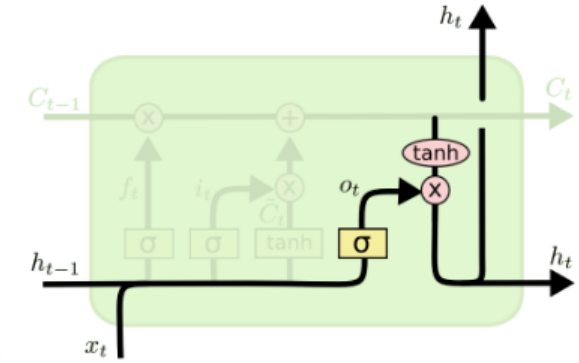
Nhân trạng thái cũ ft tương ứng. Phần tử đề cử $it \; * \; Ct$ là một giá trị đước tính toán tương ứng với bao nhiêu giá trị cập nhật mỗi trạng thái



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Ô trạng thái (cell state) mới

Cuối cùng cần quyết định đầu ra sẽ trả về như nào. Kết quả đầu ra sẽ dựa trên ô trạng thái. Đầu tiên cho chạy qua một tầng sigmoid nới quyết định của cell state sẽ ở đầu ra. Sau đó, cell state đưa qua hàm tanh(chuyển về giá trị -1 đến 1) và nhân với đầu ra của một cổng sigmoid



$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

Điều chỉnh thông tin ở đầu ra thông qua hàm tanh

3.3: Kết luận

Mô hình mạng LSTM khá giống mô hình mạng RNN,LSTM giải quyết phần nào vanishing gradient so với RNN

LSTM đã và đang được sử dụng phổ biến

III Thực nghiệm

Thử nghiệm mô hình LSTM với mô hình mạng LSTM bằng thư viện Keras

Bước 1: Khai báo thư viện và đọc dữ liệu
Tiến hành khai báo nhưng thư viện cần thiết

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

from statsmodels.tsa.seasonal import seasonal_decompose
from sklearn.preprocessing import MinMaxScaler

from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
```

Dataset:https://raw.githubusercontent.com/thieu1995/ai/master/data/preprocessing/international-airline-passengers.csv

Dataset ở đây dùng để dự báo chuỗi thời gian. Mô hình dữ báo chuỗi thời gian cho tương lai khách của hãng hàng không

Đọc 4 dòng đầu tiên của tập Dataset

	Month	International airline passengers: monthly totals in thousands. Jan 49 ? Dec 60
0	1949-01	112
1	1949-02	118
2	1949-03	132
3	1949-04	129
4	1949-05	121

Từ tập dataset trên ta sẽ dự đoán chuỗi thời gian trong tương sử dụng mô hình LSTM

Bước 2: Tiền xử lí dữ liệu

Ta sẽ dự báo cột International

Dùng hàm data.info xem kiểu dữ liệu ở các cột

```
0   Month                                                    144 non-null    object
1   International airline passengers: monthly totals in thousands. Jan 49 ? Dec 60  144 non-null    int64
```

Month và International.. đang ở dạng dữ liệu Object và số nguyên ta cần biến đổi Month về dạng datetime

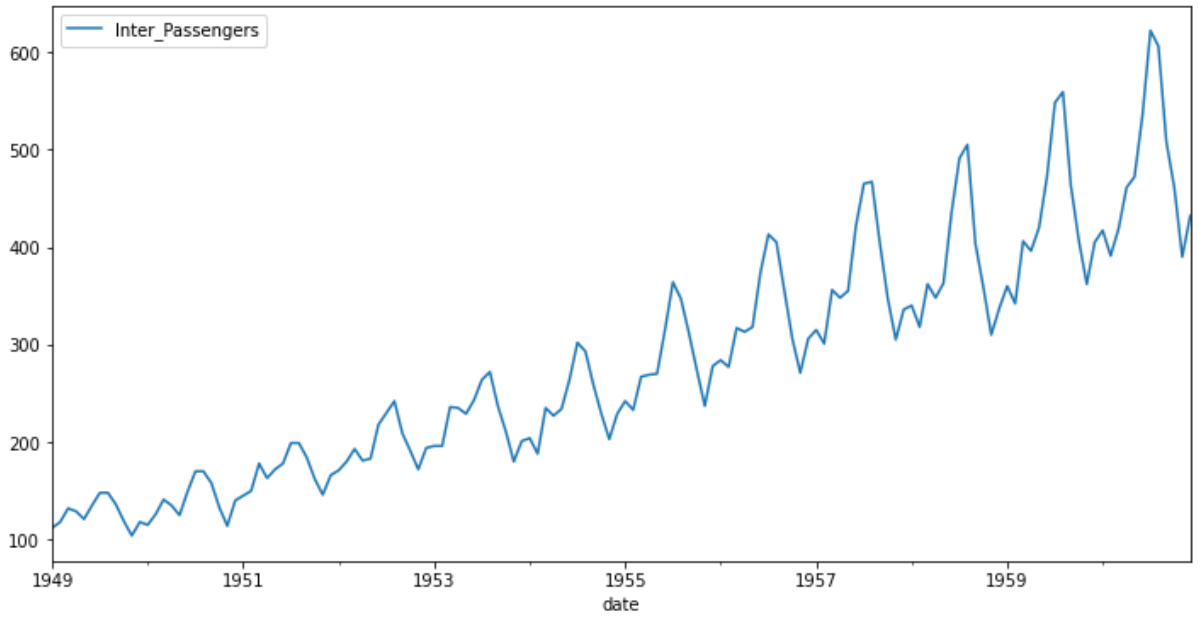
Biến đổi tên và biến đổi Month (date) về dạng dữ liệu datetime và lập chỉ mục cho Month

Đổi tên Internatinal ngắn lại thành “Inter_Passengers”

```
    0   date              144 non-null    datetime64[ns]
    1   Inter_Passengers  144 non-null    int64
```

```
df = df.set_index('date')
df.head(5)
```

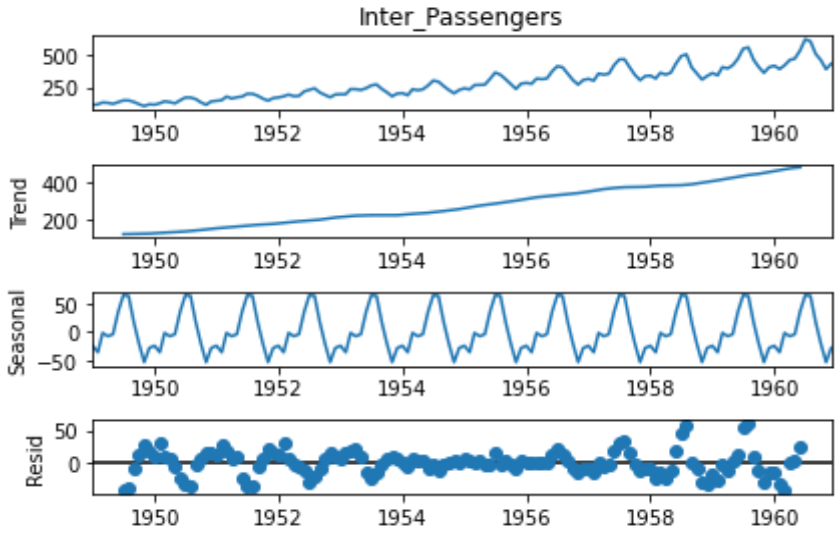
date	Inter_Passengers
1949-01-01	112
1949-02-01	118
1949-03-01	132
1949-04-01	129
1949-05-01	121



Biểu đồ dữ liệu khách hàng theo năm từ 1949 - 1960

Từ biểu đồ ta thấy lượng khách hàng lúc tăng giảm trong năm và tăng dần theo từng năm, xu hướng là tăng

Biểu đồ biểu hiển thị 3 thành phần seasonal, trend, residual của tập dữ liệu



Biểu đồ trên cho thấy Trend có xu hướng tăng còn Seasonal và Residual lúc tăng giảm trong năm

	Inter_Passengers
count	144.000000
mean	280.298611
std	119.966317
min	104.000000
25%	180.000000
50%	265.500000
75%	360.500000
max	622.000000

Trung bình mỗi Tháng sẽ có 280 hành khách và có 1 tháng cao điểm nhất là 622 hành khách và tháng ít nhất là 104 hành khách

Bước 3: Đọc dữ liệu train và test

Chia dữ liệu ra train và test

```
            Inter_Passengers
date
1949-01-01                112
1949-02-01                118
1949-03-01                132
1949-04-01                129
1949-05-01                121
...                       ...
1959-08-01                559
1959-09-01                463
1959-10-01                407
1959-11-01                362
1959-12-01                405

[132 rows x 1 columns]
            Inter_Passengers
date
1960-01-01                417
1960-02-01                391
1960-03-01                419
1960-04-01                461
1960-05-01                472
1960-06-01                535
1960-07-01                622
1960-08-01                606
1960-09-01                508
1960-10-01                461
1960-11-01                390
1960-12-01                432
```

Dùng thư viện sklearn gọi MinMaxScaler và để chuẩn hóa train và test bằng MinMaxScaler().transform()

```
scaler = MinMaxScaler()
scaler.fit(train)
scaled_train = scaler.transform(train)
scaled_test = scaler.transform(test)
```

```
print(scaled_train.shape)
print(scaled_test.shape)
```

```
(132, 1)
(12, 1)
```

```
scaled_train[:10]
```

```
array([[0.01758242],
       [0.03076923],
       [0.06153846],
       [0.05494505],
       [0.03736264],
       [0.06813187],
       [0.0967033 ],
       [0.0967033 ],
       [0.07032967],
       [0.03296703]])
```

Sử dụng Keras TimeseriesGenerator để chuẩn bị dữ liệu chuỗi thời gian cho mô hình Deep Learning

```
n_input = 3
n_features = 1
generator = TimeseriesGenerator(scaled_train, scaled_train, length=n_input, batch_size=1)
```

```
generator
```

```
<keras.preprocessing.sequence.TimeseriesGenerator at 0x7f26d00906a0>
```

```
X,y = generator[0]
print(f'Given the Array: \n{X.flatten()}')
print(f'Predict this : \n {y}')
```

```
Given the Array:
[0.01758242 0.03076923 0.06153846]
Predict this :
 [[0.05494505]]
```

Bước 4: Xây dựng mô hình

Xây dựng mô hình dự báo

```
model = Sequential()
model.add(LSTM(100, activation='relu', input_shape=(n_input, n_features)))
model.add(Dense(25))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mse')
```

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 100)               40800

 dense (Dense)               (None, 25)                2525

 dense_1 (Dense)             (None, 1)                 26

=================================================================
Total params: 43,351
Trainable params: 43,351
Non-trainable params: 0
```

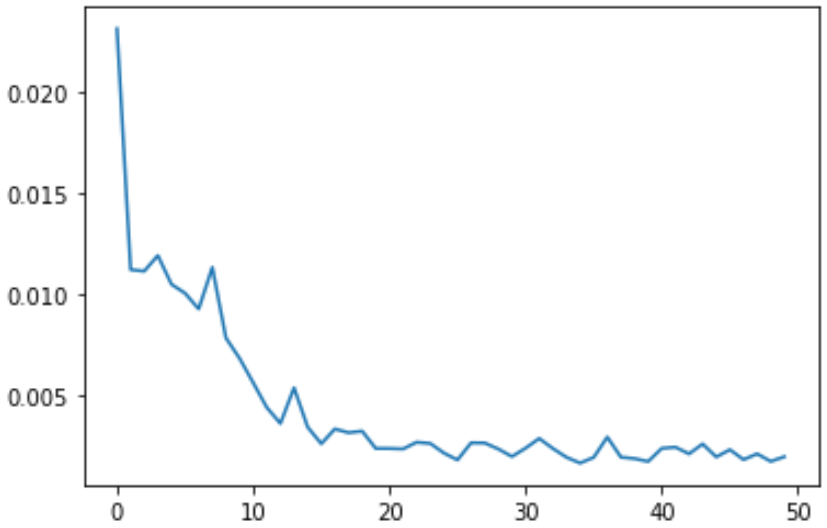
Mô hình sử dụng mạng LSTM với activation là hàm Relu với đầu vào là 3 để dự đoán 1
Thêm 2 hàm Dense layer
Compile model sử dụng optimizer là Adam và hàm loss là MSE

Bước 5: Huấn luyện mô hình

```
model.fit(generator,epochs=50)
```

```
120/120 [==============================] - 1s 7ms/step - loss: 0.0106
Epoch 7/50
120/120 [==============================] - 1s 7ms/step - loss: 0.0066
Epoch 8/50
120/120 [==============================] - 1s 7ms/step - loss: 0.0060
Epoch 9/50
120/120 [==============================] - 1s 7ms/step - loss: 0.0045
Epoch 10/50
120/120 [==============================] - 1s 7ms/step - loss: 0.0034
Epoch 11/50
120/120 [==============================] - 1s 7ms/step - loss: 0.0044
Epoch 12/50
120/120 [==============================] - 1s 7ms/step - loss: 0.0048
Epoch 13/50
120/120 [==============================] - 1s 7ms/step - loss: 0.0037
Epoch 14/50
120/120 [==============================] - 1s 7ms/step - loss: 0.0041
Epoch 15/50
120/120 [==============================] - 1s 7ms/step - loss: 0.0043
Epoch 16/50
120/120 [==============================] - 1s 7ms/step - loss: 0.0027
Epoch 17/50
120/120 [==============================] - 1s 7ms/step - loss: 0.0025
```

Huấn luyện mô hình với dữ liệu vừa chuẩn hóa và 50 epochs ta thấy hàm Loss giảm chứng tỏ dự đoán được tăng lên



Biểu đồ cho thấy hàm Loss đang giảm chứng tỏ sẽ ra kết quả tốt

Bước 6: Dữ báo mô hình

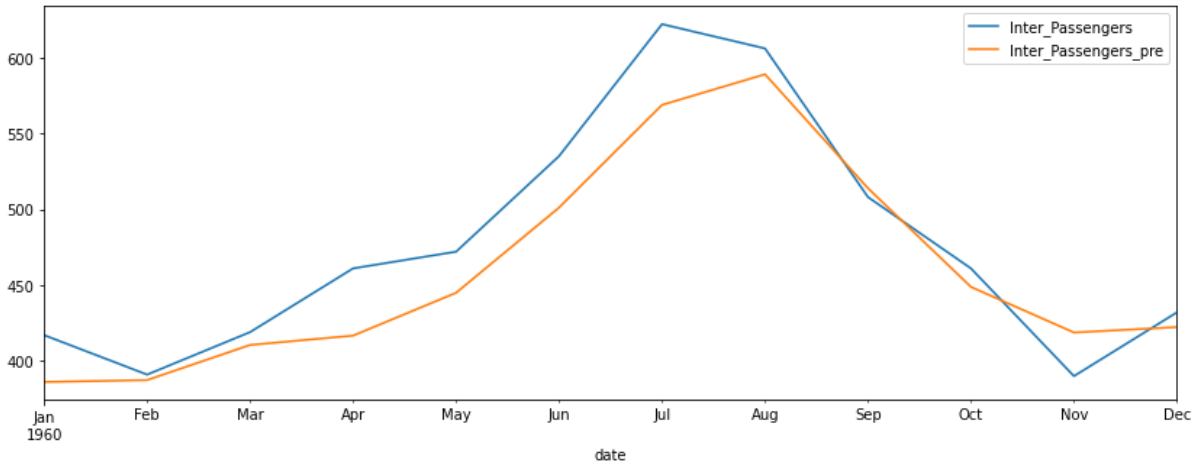
```
model.predict(last_train_batch)
```

```
1/1 [==============================] - 0s 214ms/step
array([[0.62006074]], dtype=float32)
```

Dự đoán mô hình

date	Inter_Passengers	Inter_Passengers_pre
1960-01-01	417	386.127638
1960-02-01	391	387.388045
1960-03-01	419	410.531129
1960-04-01	461	416.680913
1960-05-01	472	444.896327
1960-06-01	535	501.081174
1960-07-01	622	568.762427
1960-08-01	606	588.949229
1960-09-01	508	513.838936
1960-10-01	461	448.787222
1960-11-01	390	418.755331
1960-12-01	432	422.290383

Dự báo chuỗi thời gian 12 tháng của năm 1960
Inter_Passengers là dữ liệu ban đầu và Inter_Passengers_pre là dự báo của mô hình



Biểu đồ dữ liệu ban đầu và dự báo của năm 1960

Kết luận: Từ biểu đồ cho thấy dự báo của model cũng không chênh lệch quá nhiều, 2 đường đi cũng rất gần nhau, từ tháng 5 - 7 cách xa nhau cũng không quá lớn

IV: Tài Liệu Tham Khảo

Time Series Forecasting With RNN(LSTM)| Complete Python Tutorial|
Time Series Prediction with LSTMs using TensorFlow 2 and Keras in Python
https://keras.io/examples/timeseries/timeseries_weather_forecasting/
https://www.cienciadedatos.net/documentos/py27-time-series-forecasting-python-scikitlearn.html
https://www.tensorflow.org/tutorials/structured_data/time_series
https://viblo.asia/s/su-dung-mang-lstm-long-short-term-memory-de-du-doan-co-phieu-24lJDz06KPM