ĐẠI HỌC BÁCH KHOA HÀ NỘI

Viện Công nghệ thông tin và truyền thông

Dự đoán xu thế chỉ số chứng khoán với mô hình RNN

Môn: Kĩ thuật lập trình.

Giảng viên hướng dẫn : PGS.TS Huỳnh Quyết Thắng

Sinh viên thực hiện:

Họ và tên	MSSV	Lớp
Nguyễn Hùng Cường	20164820	KSTN_CNTT_K61
Nguyễn Tuấn Hưng	20164837	KSTN_CNTT_K61
Nguyễn Tiến Tài	20162031	KSTN_CNTT_K61

Mục lục

Chương 1 : Các khái niệm cơ bản

1.1 Chỉ số chứng khoán VN-Index

Ở Việt Nam có ba sàn giao dịch đó là sàn HOSE ở TP. Hồ Chí Minh và hai sàn HNX, Upcom ở Hà Nội. Chỉ số chứng khoán Việt Nam VN-Index là chỉ số thể hiện xu hướng biến động giá của tất cả cổ phiếu niêm yết và giao dịch tại sàn HOSE. Ở đây, giá chỉ số VN-Index được hiểu là giá đóng cửa chỉ số VN-Index sau mỗi ngày giao dịch trên TTCK Việt Nam. Đây cũng chính là giá tham chiếu (hay giá mở cửa) cho chỉ số VN-Index trong ngày giao dịch kế tiếp. Dưới đây là công thức tính chỉ số VN-Index:

$$\text{VN} - \text{Index} = \frac{\text{CMV}}{\text{BMV}} * 100$$

Trong đó CMV là tổng giá trị thị trường của các cổ phiếu niêm yết hiện tại, BMV là tổng giá trị của các cổ phiếu niêm yết cơ sở tính ở thời điểm gốc ngày 28/07/2000, ngày đầu tiên thị trường chứng khoán chính thức đi vào hoạt động. Giá trị vốn hóa thị trường cơ sở tính trong công thức chỉ số được điều chỉnh trong các trường hợp như niêm yết mới, hủy niêm yết và các trường hợp có thay đổi về vốn niêm yết. Gọi M là số lượng công ty niêm yết trên sàn HOSE; p_{it}, q_{it} tương ứng là giá và số lượng cổ phiếu niêm yết của công ty thứ i trên sàn HOSE tại thời điểm hiện tại; p_{io}, q_{io} tương ứng là giá và số lượng cổ phiếu niêm yết của công ty thứ i trên sàn HOSE tại thời điểm cơ sở. Công thức tính CMV và BMV như sau:

$$\text{CMV} = \sum_{i=1}^{M} p_{it} * q_{it}$$

$$\text{BMV} = \sum_{i=1}^{M} p_{io} * q_{io}$$

1.2 Mô hình chuỗi thời gian

Chuỗi thời gian là một chuỗi các điểm dữ liệu, được đo theo từng khoảnh khắc thời gian liền nhau theo một tần suất thời gian thống nhất. Một ví dụ cho chuỗi thời gian là giá đóng cửa của chỉ số VN-Index qua các ngày giao dịch. Gọi Y_t là chuỗi thời gian đầu vào. Chuỗi thời gian này có thể phân tích thành bốn thành phần:

Thành phần xu thế (gọi là T_t): Xu thế mang tính chất dài hạn, thể hiện mẫu hình tăng hay giảm của các giá trị trong chuỗi thời gian. Kỹ thuật để xác định giá trị thành phần xu thế tại vị trí k trong chuỗi thời gian là lấy trung bình giá trị tại các điểm liên tiếp trong chuỗi thời gian xung quanh lân cận của điểm đó sao cho độ dài bằng khoảng chu kỳ, và xem đó là giá trị xu thế của chuỗi thời gian tại điểm đó. Trong trường hợp chu kỳ lẻ có dạng (2C+1), công thức tính giá trị thành phần xu thế như sau:

$$T_k = \frac{1}{2C+1} * \sum_{i=-C}^{C} Y_{i+k}$$

Trong công thức trên k là vị trí của điểm đang xét trong chuỗi thời gian. T_k là giá trị thành phần xu thế tại điểm k. Y_{i+k} là giá trị chuỗi thời gian đầu vào tại điểm (i+k). Ví dụ với chu kỳ năm ngày, giá trị thành phần xu thế tại điểm k = 10 được tính theo công thức triển khai sau:

$$T_{10} = \frac{1}{5} * \sum_{i=-2}^{2} Y_{i+k} = \frac{Y_8 + Y_9 + Y_{10} + Y_{11} + Y_{12}}{5}$$

Thành phần thời vụ (gọi là S_t): Tính thời vụ thể hiện sự tuần hoàn của dữ liệu chuỗi thời gian trong một khoảng thời gian xác định. Thành phần này thể hiện ảnh hưởng của mùa vụ như tuần, tháng, quý, năm lên giá trị chuỗi dữ liệu.

Thành phần chu kỳ (gọi là C_t): Thành phần này được đặc trưng bởi hệ số biến đổi mùa, thể hiện sự tăng giảm lặp lại của các giá trị trong chuỗi thời gian theo một giai đoạn không cố định. Khoảng thời gian chu kỳ thường lớn hơn nhiều so với khoảng thời gian mùa vụ trong chuỗi thời gian.

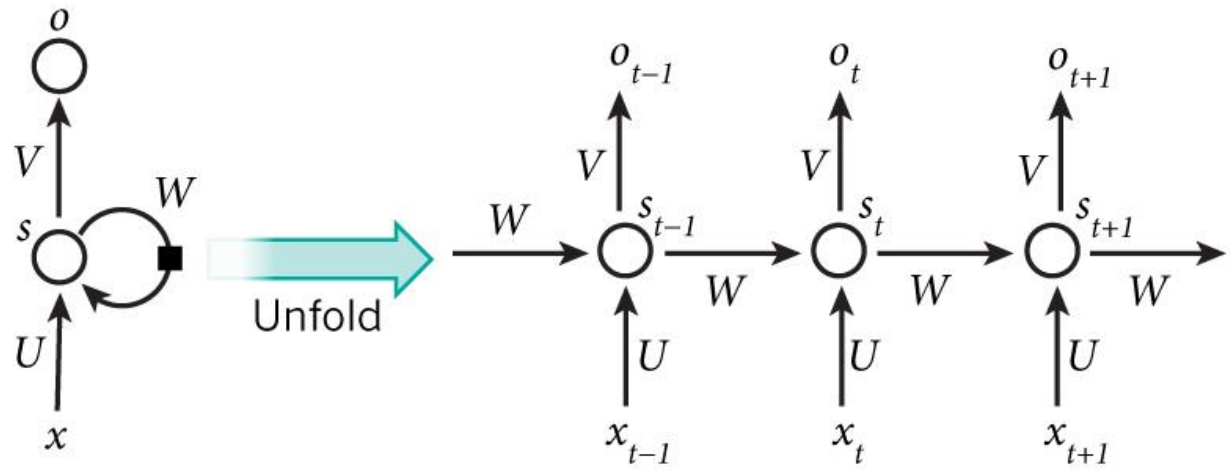
Thành phần ngẫu nhiên (gọi là R_t): Là thành phần còn lại sau khi loại bỏ đi ba thành phần ở trên từ chuỗi thời gian ban đầu. Thành phần ngẫu nhiên có tính dừng. Tính dừng thể hiện ở hàm tự tương quan (ACF) giữa một điểm với các điểm trong quá khứ là nhỏ, thường nằm trong

khoảng $\pm\frac{1.96}{\sqrt{N}}$ với N là kích thước chuỗi thời gian. Giá trị 1.96 thể hiện khoảng tin cậy 95% của phân phối Gauss chuẩn kỳ vọng 0 và phương sai 1.

Có hai cách cơ bản để tổng hợp các thành phần của chuỗi thời gian để thu được chuỗi ban đầu. Một là phương pháp cộng, thực hiện bằng cách lấy tổng các thành phần ($Y_t = T_t + S_t + C_t + R_t$). Hai là phương pháp nhân, thực hiện bằng cách lấy tích các thành phần ($Y_t = T_t * S_t * C_t * R_t$). Trên thực tế các thư viện hỗ trợ phân tách chuỗi thời gian thường không phân biệt thành phần T_t và C_t, và gọi chung đó là thành phần xu thế. Trong luận văn này tác giả lựa chọn phương pháp cộng khi thực hiện phân tách chuỗi thời gian đầu vào.

1.3 Mô hình cơ bản RNN

Ý tưởng chính của RNN (Recurrent Neural Network) là sử dụng chuỗi các thông tin. Trong các mạng nơ-ron truyền thống tất cả các đầu vào và cả đầu ra là độc lập với nhau. Tức là chúng không liên kết thành chuỗi với nhau. Nhưng các mô hình này không phù hợp trong rất nhiều bài toán. Ví dụ, nếu muốn đoán từ tiếp theo có thể xuất hiện trong một câu thì ta cũng cần biết các từ trước đó xuất hiện lần lượt thế nào chứ nhỉ? RNN được gọi là hồi quy (Recurrent) bởi lẽ chúng thực hiện cùng một tác vụ cho tất cả các phần tử của một chuỗi với đầu ra phụ thuộc vào cả các phép tính trước đó. Nói cách khác, RNN có khả năng nhớ các thông tin được tính toán trước đó. Trên lý thuyết, RNN có thể sử dụng được thông tin của một văn bản rất dài, tuy nhiên thực tế thì nó chỉ có thể nhớ được một vài bước trước đó (ta cùng bàn cụ thể vấn đề này sau) mà thôi. Về cơ bản một mạng RNN có dạng như sau:



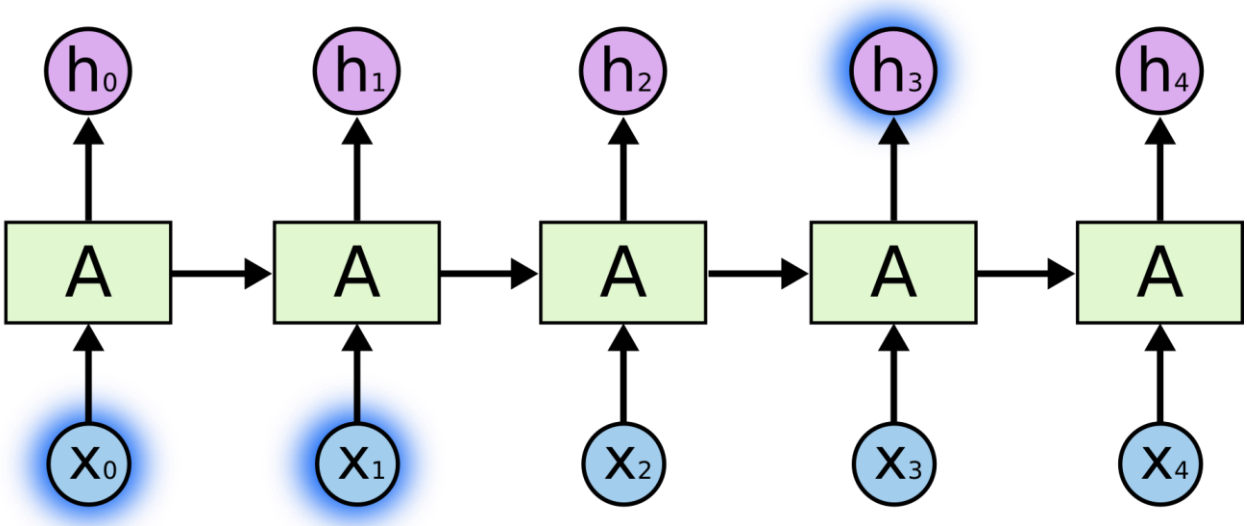
A recurrent neural network and the unfolding in time of the computation involved in its forward computation. Source: Nature

Mô hình trên mô tả phép triển khai nội dung của một RNN. Triển khai ở đây có thể hiểu đơn giản là ta vẽ ra một mạng nơ-ron chuỗi tuần tự. Ví dụ ta có một câu gồm 5 chữ "*Đẹp trai lắm gái theo*", thì mạng nơ-ron được triển khai sẽ gồm 5 tầng nơ-ron tương ứng với mỗi chữ một tầng. Lúc đó việc tính toán bên trong RNN được thực hiện như sau:

- x_t là đầu vào tại bước t. Ví dụ x_1 là một vec-tơ one-hot tương ứng với từ thứ 2 của câu (*trai*).
- s_t là trạng thái ẩn tại bước t. Nó chính là ***bộ nhớ*** của mạng. s_t được tính toán dựa trên cả các trạng thái ẩn phía trước và đầu vào tại bước đó: $s_t=f(Ux_t+Ws_{t-1})$. Hàm f thường là một hàm phi tuyến tính như hàm tanh, ReLU. Để làm phép toán cho phần tử ẩn đầu tiên ta cần khởi tạo thêm s_{-1}, thường giá trị khởi tạo được gắn bằng 0.
- o_t là đầu ra tại bước t. Ví dụ, ta muốn dự đoán từ tiếp theo có thể xuất hiện trong câu thì o_t chính là một vec-tơ xác xuất các từ trong danh sách từ vựng của ta : $o_t=\text{softmax}(Vs_t)$

- Một điểm nổi bật của RNN chính là ý tưởng kết nối các thông tin phía trước để dự đoán cho hiện tại. Việc này tương tự như ta sử dụng các cảnh trước của bộ phim để hiểu được cảnh hiện thời. Nếu mà RNN có thể làm được việc đó thì chúng sẽ cực kì hữu dụng, tuy nhiên liệu chúng có thể làm được không? Câu trả lời là *còn tùy*.

    - Đôi lúc ta chỉ cần xem lại thông tin vừa có thôi là đủ để biết được tình huống hiện tại. Ví dụ, ta có câu: "*các đám may trên bầu trời*" thì ta chỉ cần đọc tới "*các đám may trên bầu*" là đủ biết được chữ tiếp theo là "*trời*" rồi. Trong tình huống này, khoảng cách tới thông tin có được cần để dự đoán là nhỏ, nên RNN hoàn toàn có thể học được.



Nhưng trong nhiều tình huống ta buộc phải sử dụng nhiều ngữ cảnh hơn để suy luận. Ví dụ, dự đoán chữ cuối cùng trong đoạn: "*I grew up in France… I speak fluent French.*". Rõ ràng là các thông tin gần ("*I speak fluent*") chỉ có phép ta biết được đằng sau nó sẽ là tên của một ngôn ngữ nào đó, còn không thể nào biết được đó là tiếng gì. Muốn biết là tiếng gì, thì ta cần phải có thêm ngữ cảnh "*I grew up in France*" nữa mới có thể suy luận được. Rõ ràng là khoảng cách thông tin lúc này có thể đã khá xa rồi.

Thật không may là với khoảng cách càng lớn dần thì RNN bắt đầu không thể nhớ và học được nữa. Tuy nhiên, rất cám ơn là LSTM không vấp phải vấn đề đó. Chúng em chọn mô hình này cho bài tập của mình.

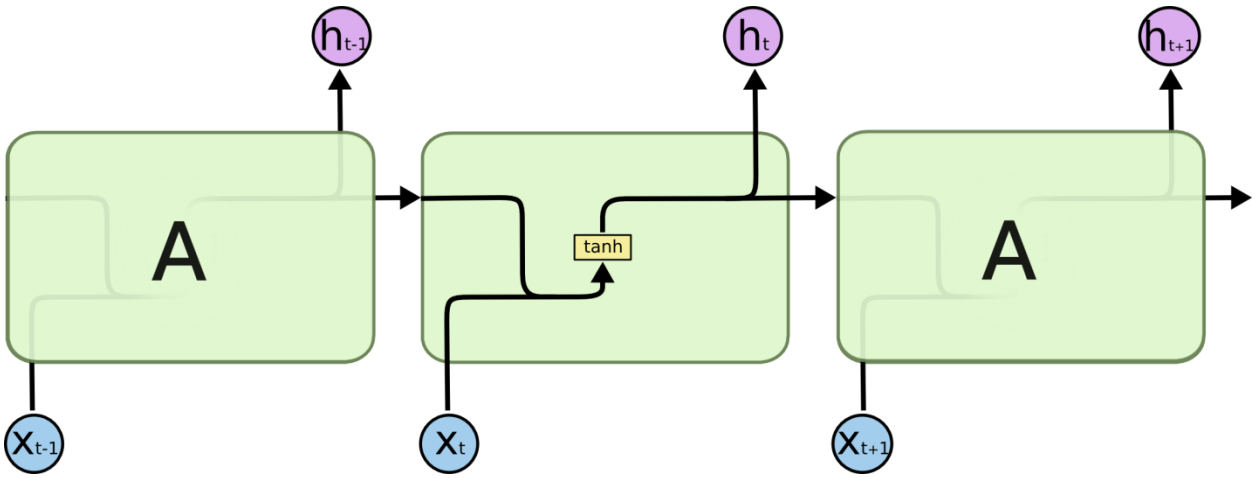
1.4 Mô hình LSTM trong RNN

Mạng bộ nhớ dài-ngắn (Long Short Term Memory networks), thường được gọi là LSTM - là một dạng đặc biệt của RNN, nó có khả năng học được các phụ thuộc xa. LSTM được giới thiệu bởi Hochreiter & Schmidhuber (1997), và sau đó đã được cải tiến và phổ biến bởi rất nhiều người trong ngành. Chúng hoạt động cực kì hiệu quả trên nhiều bài toán khác nhau nên dần đã trở nên phổ biến như hiện nay.

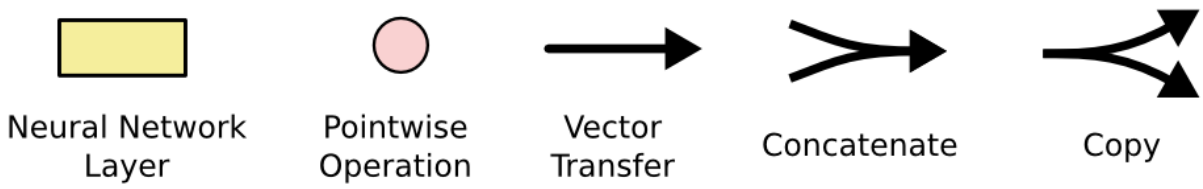
LSTM được thiết kế để tránh được vấn đề phụ thuộc xa (long-term dependency). Việc nhớ thông tin trong suốt thời gian dài là đặc tính mặc định của chúng, chứ ta không cần phải huấn luyện nó để có thể nhớ được. Tức là ngay nội tại của nó đã có thể ghi nhớ được mà không cần bất kì can thiệp nào.

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

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



Làm quen với các kí tự :

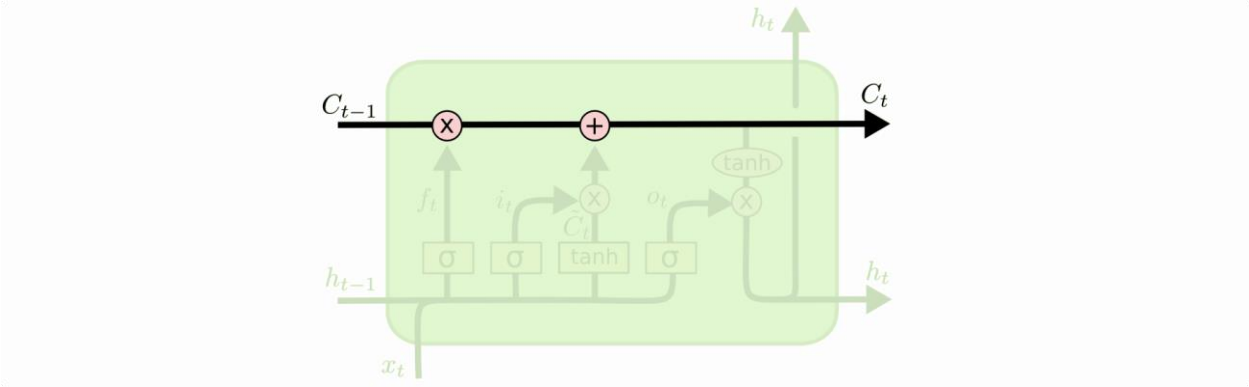


Ở sơ đồ trên, mỗi một đường mang một véc-tơ từ đầu ra của một nút tới đầu vào của một nút khác. Các hình trong màu hồng biểu diễn các phép toán như phép cộng véc-tơ chẳng hạn, còn các ô màu vàng được sử dụng để học trong các từng mạng nơ-ron. Các đường hợp nhau kí hiệu việc kết hợp, còn các đường rẽ nhánh ám chỉ nội dung của nó được sao chép và chuyển tới các nơi khác nhau.

1.4.1. Ý tưởng cốt lõi của LSTM

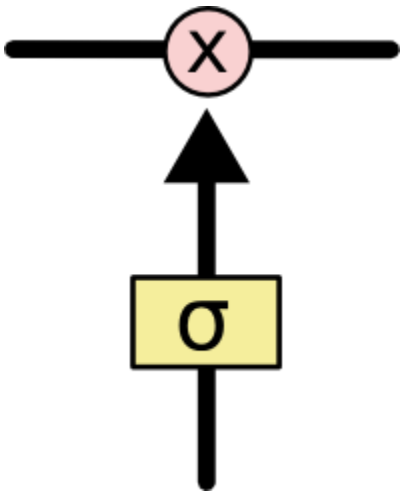
Chìa khóa của LSTM là trạng thái tế bào (cell state) - chính đường chạy thông ngang phía trên của sơ đồ hình vẽ.

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



LSTM có khả năng bỏ đi hoặc thêm vào các thông tin cần thiết cho trạng thái tế báo, chúng được điều chỉnh cẩn thận bởi các nhóm được gọi là cổng (gate).

Các cổng là nơi sàng lọc thông tin đi qua nó, chúng được kết hợp bởi một tầng mạng sigmoid và một phép nhân.



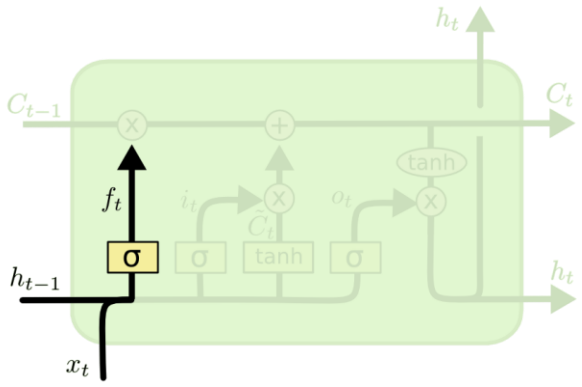
Tầng sigmoid sẽ cho đầu ra là một số trong khoản [0, 1][0,1], mô tả có bao nhiêu thông tin có thể được thông qua. Khi đầu ra là 00 thì có nghĩa là không cho thông tin nào qua cả, còn khi là 11 thì có nghĩa là cho tất cả các thông tin đi qua nó.

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

1.4.2. Bên trong LSTM

Bước đầu tiên của LSTM là quyết định xem thông tin nào cần bỏ đi từ trạng thái tế bào. Quyết định này được đưa ra bởi tầng sigmoid - gọi là "tầng cổng quên" (forget gate layer). Nó sẽ lấy đầu vào là *ht*−1 và *xt* rồi đưa ra kết quả là một số trong khoảng [0, 1][0,1] cho mỗi số trong trạng thái tế bào *Ct*−1. Đầu ra là 11thể hiện rằng nó giữ toàn bộ thông tin lại, còn 00 chỉ rằng taonf bộ thông tin sẽ bị bỏ đi.

Quay trở lại với ví dụ mô hình ngôn ngữ dự đoán từ tiếp theo dựa trên tất cả các từ trước đó, với những bài toán như vậy, thì trạng thái tế bào có thể sẽ mang thông tin về giới tính của một nhân vật nào đó giúp ta sử dụng được đại từ nhân xưng chuẩn xác. Tuy nhiên, khi đề cập tới một người khác thì ta sẽ không muốn nhớ tới giới tính của nhân vật nữa, vì nó không còn tác dụng gì với chủ thế mới này.

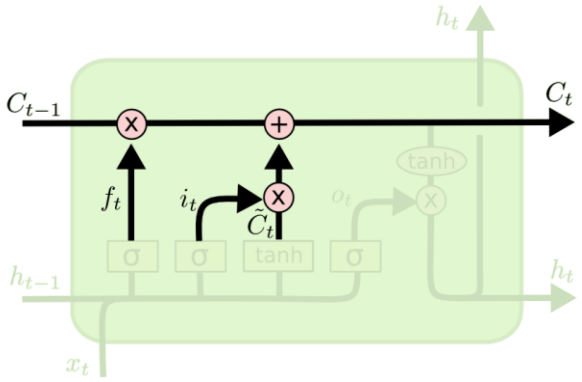


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Giờ là lúc cập nhập trạng thái tế bào cũ *Ct*−1 thành trạng thái mới *Ct*. Ở các bước trước đó đã quyết định những việc cần làm, nên giờ ta chỉ cần thực hiện là xong.

Ta sẽ nhân trạng thái cũ với *ft* để bỏ đi những thông tin ta quyết định quên lúc trước. Sau đó cộng thêm *it*∗*Ct*~. Trạng thái mơi thu được này phụ thuộc vào việc ta quyết định cập nhập mỗi giá trị trạng thái ra sao.

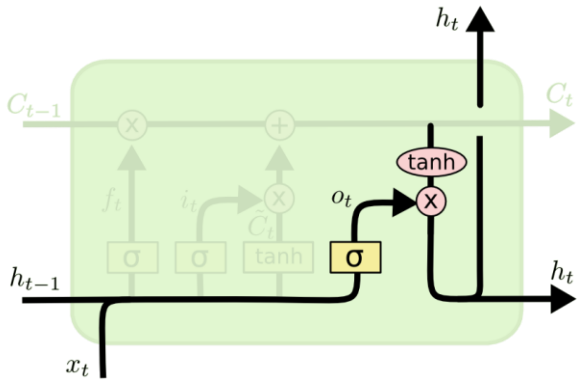
Với bài toàn mô hình ngôn ngữ, chính là việc ta bỏ đi thông tin về giới tính của nhân vật cũ, và thêm thông tin về giới tính của nhân vật mới như ta đã quyết định ở các bước trước đó.



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

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

Với ví dụ về mô hình ngôn ngữ, chỉ cần xem chủ thể mà ta có thể đưa ra thông tin về một trạng từ đi sau đó. Ví dụ, nếu đầu ra của chủ thể là số ít hoặc số nhiều thì ta có thể biết được dạng của trạng từ đi theo sau nó phải như thế nào.



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

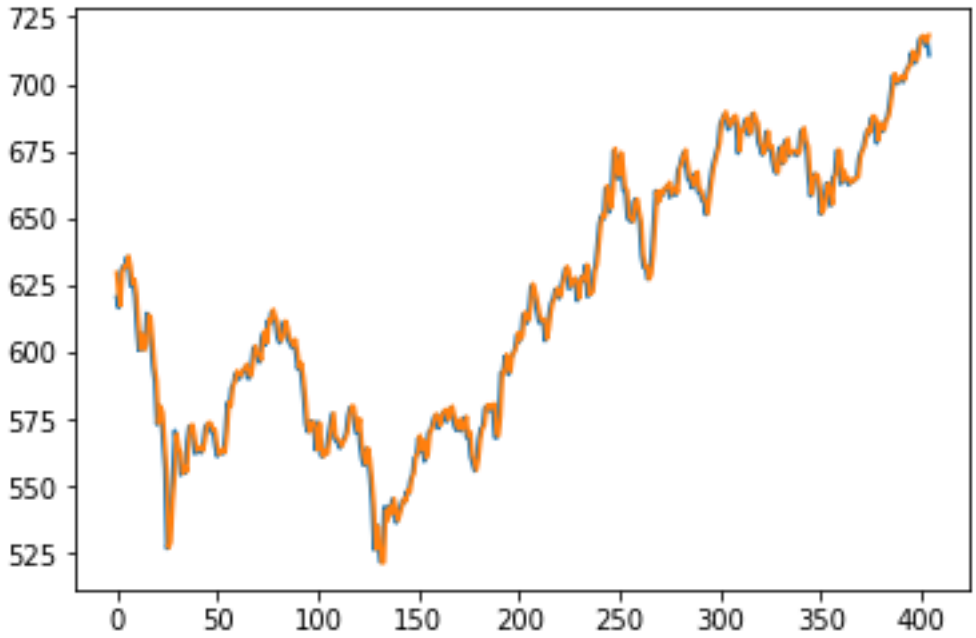
$$h_t = o_t * \tanh\left(C_t\right)$$

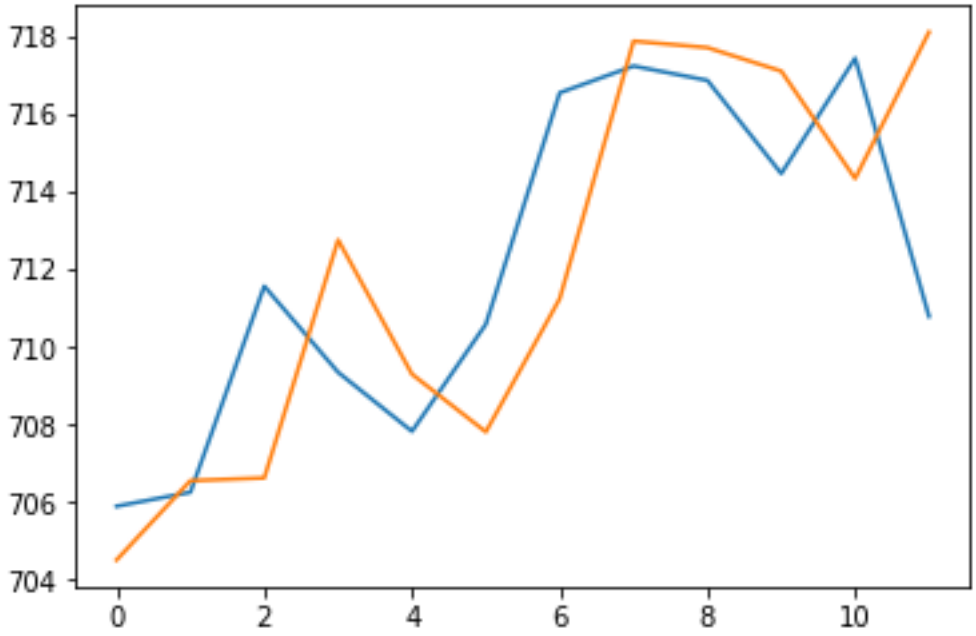
CHƯƠNG 2: Chạy và chỉnh sửa chương trình

Trong tài liệu chúng em nhận được gồm có hai chương trình dự đoán chỉ số VN-Index. Trong đó thì chúng em tập chung nghiên cứu về mô hình LSTM với sự hỗ trợ của anh Việt Anh

2.1 Kết quả chạy thử nghiệm chương trình

Chương trình chạy xử dụng mô hình phân tích chuỗi thời gian time-series với LSTM





Hình 2.1 epoch=50, rmse = 4.5

2.2 Chạy và chỉnh sửa code chương trình

2.2.1 Mô tả chương trình

Trong phần bài tập lớn thì chúng em chủ yếu tập trung nghiên cứu mô hình cải tiến này.Trong chương trình này chúng em đã làm mới bộ dữ liệu đầu vào đến

Ngày 20-5-2018,cùng với đó là chúng em đã chỉnh sửa một vài dữ liệu trong chương trình để chương trình dự đoán trong vòng 20 ngày của tháng 7-2017 thay vì dự đoán vào ngày cuối năm 2016 như chương trình cũ. Đồng thời chúng em sửa lại cách tính sai số để tính chính xác sai số các phương pháp dự đoán.

Trong chương trình này có sử dụng các gói thư viện:

Mô Tả	**Tên Gói Thư Viện**
Các gói thư viện xử lý toán học và hỗ trợ mô hình LSTM	- Numpy - Scipy - Pandas - Kereas
Gói thư viện xử lý đồ thị	- Matplotlib

2.2.2 Dữ liệu đầu vào của chương trình

Dữ liệu đầu vào của chương trình được chứa trong file raw_data.csv

Dữ liệu đầu vào này được chúng em làm mới phù hợp với định dạng yêu cầu

Nguồn dữ liệu này được bọn em lấy từ chỗ anh Việt Anh. Chúng em tách bộ dữ liệu thành 2:

0.9 bộ dữ liệu để train, còn lại dung để test và phát triển

2.2.3 Chương trình và một số thành phần cơ bản

1.Chuẩn hóa dữ liệu đầu vào:

```
def
scale(train,test):
                        # fit scaler
                         scaler = MinMaxScaler(feature_range=(-1, 1))
                         scaler = scaler.fit(train)
                         # transform train
                         train = train.reshape(train.shape[0],
                                               train.shape[1])
                         train_scaled = scaler.transform(train)
                         # transform test
                         test = test.reshape(test.shape[0],
                                             test.shape[1])
                         test_scaled = scaler.transform(test)
                         return scaler, train_scaled, test_scaled
```

Hàm này trả về các giá trị của các chỉ số đóng cửa của các ngày được khảo sát và train, tuy nhiên được scale lại trong khoảng -1; 1

2. Hàm dự báo theo phương pháp LSTM

```
def
forecast_lstm(model,batch_size,X):
                                        #reshape X to fit keras
                                    X = X.reshape(1, 1, len(X))
                                    yhat = model.predict(X,
                                    batch_size=batch_size)
                                    return yhat[0,0]
```

Hàm này trả về giá trị dự đoán xu hướng của chỉ số ngày hôm sau dựa vào những dữ liệu đã học được từ mô hình LSTM

3. Hàm xây dựng mô hình LSTM

```
def
fit_lstm(train,batch_size,nb_epoch,
neurons):
                                        X,y = train[:,0:-1], train[:,-1]
                                        X =
                                        X.reshape(X.shape[0],1,X.shape[1
                                        ])

                                            #X=[ [[1,2,3,...]] ,
                                        [[2,3,4,....]] , [[3,4,5,....]]
                                        , ....]
```

```
    #init
model = Sequential()
    #choose neurons, batch (3-
dimensions) , stateful
model.add(LSTM(neurons,
batch_input_shape = (batch_size,
X.shape[1],
X.shape[2]),stateful= True))
model.add(Dense(1))



    ####### set to test
#########
model.compile(loss='mean_squared
_error', optimizer='adam')
    ####### set to test
#########


for i in range(nb_epoch):
        #verbose
model.fit(X, y, epochs=1,
batch_size=batch_size,
verbose=0, shuffle=False)
model.reset_states()

return model
```

Trong hàm này chúng em sử dụng thư viện kereas để build LSTM, do đó không cần setup các bước activation như phần trước chúng em trình bày. Với các tham số:

Train : là bộ dữ liệu train,

Batch_size : là độ lớn của mỗi tập dữ liệu chi bị chia nhỏ,

nb_epoch : số epoch trong mỗi lần train

neurons : số lượng hidden_layers

Hàm này sẽ trả về 1 model được optimized theo phương pháp adam, và model này kết hợp với đầu vào để đưa ra những giá trị predict.

4. Đưa ra kết quả

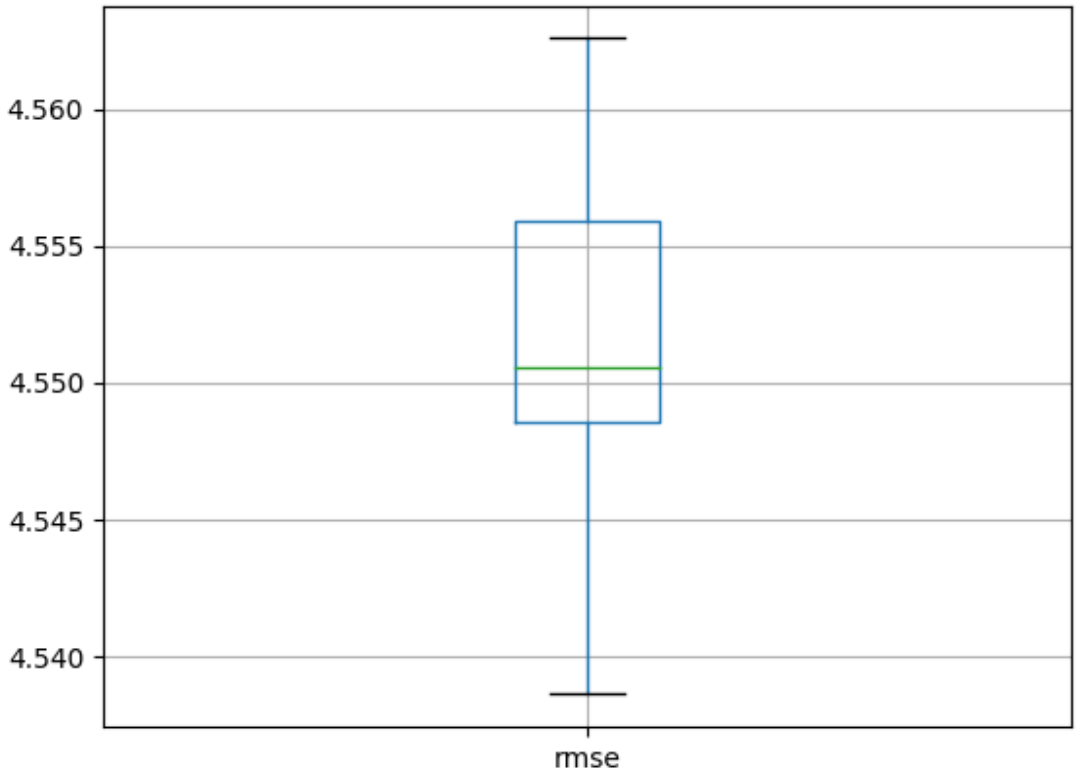
Với mô hình như ở phần 3 thì chúng em cho mô hình này học đi, học lại trong “repeats” lần

```
batch_size=1
              nb_epoch = 3
              neurons = 2
              repeats = 20
              error_scores = list()
              for r in range(repeats):

                      ####### set to test #########
                  lstm_model = fit_lstm(train_scaled,
              batch_size,nb_epoch, neurons)
                  ####### set to test #########




                  # forecast the training dataset to build up state
              for forecasting
                  train_reshaped = train_scaled[:,
              0].reshape(len(train_scaled), 1, 1)



                  ####### set to test #########
                  lstm_model.predict(train_reshaped, batch_size=1)
                  ####### set to test #########









                  #predict for test data
                  predictions = list()
                  for i in range(len(test_scaled)):
                      # make one-step forecast
                      X, y = test_scaled[i, 0:-1], test_scaled[i, -1]

                      yhat = forecast_lstm(lstm_model, 1, X)
```

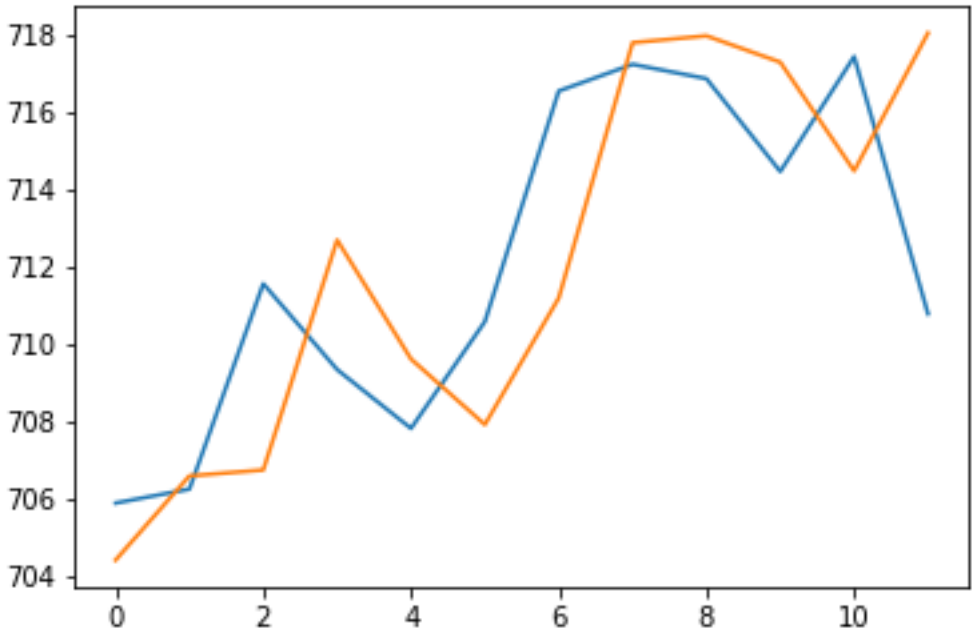
```
        #    yhat= y

            # invert scaling
            yhat = invert_scale(scaler, X, yhat)
            # invert differencing
            yhat = inverse_difference(datas, yhat,
len(test_scaled)+1-i)
            # store forecast
            predictions.append(yhat)
            expected = datas[len(train) + i + 1]
#           print('Month=%d, Predicted=%f, Expected=%f' %
(i+1, yhat, expected))



        rmse = math.sqrt(mean_squared_error(datas[-
test_size:], predictions))
        print('%d) Test RMSE: %.3f' % (r+1, rmse))
        error_scores.append(rmse)
```

Trong hàm này chúng em in ra độ lệch trung bình của giá trị predict và giá trị trong bộ test

2.2.4 Kết quả chạy chương trình




```
1) Test RMSE: 4.548
2) Test RMSE: 4.549
3) Test RMSE: 4.551
4) Test RMSE: 4.556
5) Test RMSE: 4.548
6) Test RMSE: 4.563
7) Test RMSE: 4.553
8) Test RMSE: 4.550
9) Test RMSE: 4.552
10) Test RMSE: 4.549
11) Test RMSE: 4.548
12) Test RMSE: 4.545
13) Test RMSE: 4.557
14) Test RMSE: 4.561
15) Test RMSE: 4.550
16) Test RMSE: 4.549
17) Test RMSE: 4.539
18) Test RMSE: 4.552
19) Test RMSE: 4.560
20) Test RMSE: 4.562
             rmse
count  20.000000
mean    4.552035
std     0.006157
min     4.538628
25%     4.548535
50%     4.550554
75%     4.555936
max     4.562591
```



Độ lệch trung bình là 4.5

2.3.Đánh giá chương trình và so sánh với chương trình nhóm 3 làm

Chương trình mô hình cải tiến như trên được đánh giá chạy thử trong 90% số ngày dữ liệu .Tỉ lệ đúng của chương trình là đạt khoảng 70%.giá trị sai số bằng khoảng 0.8% nghĩa là khoảng 4.5 điểm.

So sánh với nhóm 3 khi cả hai đều chạy trên dữ liệu cập nhập của nhóm em làm thì sai số của nhóm em thấp hơn so với nhóm 3. Sai số trung bình của nhóm em là 4.5 điểm trong khi đó của nhóm 3 là 5.5 điểm ít hơn họ 1 điểm.Về thời gian tính toán thì chương trình của nhóm em chạy nhanh hơn chương trình của nhóm 3 khi được chạy trên cùng một máy tính. Và hiện nay mô hình này cũng đang được sử dụng rộng rãi trên các sàn chứng khoán trong nước và quốc tế.