TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN TOÁN ỨNG DỤNG VÀ TIN HỌC

o0o

ĐỒ ÁN CHUỖI THỜI GIAN

So sánh mô hình LSTM và mô hình ARIMA trong dự báo ô nhiễm môi trường

Phan Tiến Đạt - 20195854

Dương Đình Văn - 20206185

Nguyễn Ngọc Quang - 20185395

Ngành: Toán Tin

Giảng viên hướng dẫn: TS. Nguyễn Thị Ngọc Anh

Chữ kí của GVHD

Viện: Toán ứng dụng và Tin học

HÀ NỘI, 12/2021

Lời cảm ơn

Nhóm chúng em xin gửi lời cảm ơn sâu sắc tới **TS.Nguyễn Thị Ngọc Anh**, người giảng viên đã tận tình chỉ bảo, luôn theo dõi sát sao và giúp đỡ chúng em trong quá trình nghiên cứu. Không có những lời động viên và hướng dẫn của cô, đồ án sẽ không thể hoàn thiện.

Chúng em cũng xin gửi lời cảm ơn đến Viện Toán ứng dụng và Tin học, Trường Đại học Bách Khoa Hà Nội đã cung cấp những kiến thức để tạo điều kiện thuận lợi cho chúng em hoàn thành đồ án này.

Nhóm chúng em đặc biệt cảm ơn gia đình, người thân, bạn bè luôn ở bên và hỗ trợ trong quá trình thực hiện đồ án.

Chúng em xin chân thành cảm ơn!

Tóm tắt nội dung đồ án

Dự báo chuỗi thời gian là một lĩnh vực đã được nghiên cứu từ lâu và được nhiều người quan tâm. Tuy nhiên, trong những năm gần đây, Machine Learning và cụ thể hơn là Deep Learning đang phát triển mạnh mẽ, đạt được nhiều thành tựu vượt trội và thể hiện được nhiều tiềm năng ứng dụng trong chuỗi thời gian. Vì vậy, trong đồ án này, chúng em sẽ sử dụng một mô hình mô hình Deep Learning hiện đại là LSTM trong bài toán dự đoán ô nhiễm môi trường hiện nay.

Từ khóa: *Deep Learning, Long Short-Term Memory (LSTM), Autoregressive Integrated Moving Average (ARIMA), Forecasting, Time Series Data..*

Hà Nội, ngày 7 tháng 3 năm 2023

Học viên thực hiện
Phan Tiến Đạt
Nguyễn Ngọc Quang
Dương Đình Văn

MỤC LỤC

DANH MỤC HÌNH VẼ

DANH MỤC BẢNG BIỂU

DANH MỤC KÍ HIỆU VÀ CHỮ VIẾT TẮT

Từ viết tắt	**Ý nghĩa**
LSTM	Long short-term memory
RNN	Recurrent Neural Network
ML	Machine Learning
DF	Deep Learning

CHƯƠNG 1. MỞ ĐẦU

1.1 Lý do chọn đề tài

Sự biến đổi khí hậu đang trở thành một trong những vấn đề quan trọng nhất đang được quan tâm trên toàn thế giới. Việc dự báo thời tiết chính xác và đáng tin cậy là rất cần thiết để giúp cho các nhà quản lý đô thị, các nhà nghiên cứu và các cá nhân đưa ra các quyết định hợp lý về kinh tế, môi trường và an toàn.

Nhằm ứng phó với những vấn đề về thời tiết, con người từ xa xưa đến nay đã luôn tìm tòi và nghiên cứu về một số phương pháp dự báo. Những phương pháp này giúp những nhà quản lý có thể đưa ra các quyết định hợp lý về kinh tế, môi trường và an toàn.

Trong vấn đề dự đoán thời tiết sử dụng chuỗi thời gian có khá nhiều phương pháp được áp dụng. Nhưng có 2 hướng tiếp cận chính là phương pháp thống kê và phương pháp học máy. Trong phương pháp thống kê, các mô hình chuỗi thời gian như ARMA, ARIMA, SARIMA đã được áp dụng một cách rộng rãi. Trong phương pháp học máy có rất nhiều mô hình được áp dụng cho dự báo mực nước như hồi quy tuyến tính (LRM-Linear Regreesion Model), rừng ngẫu nhiên hồi quy(Random Forest Regression),hồi quy vec-tơ hỗ trợ(SVR-Support Vector Regression), hồi quy LASSO, mạng nơ-ron nhân tạo,. . . Trong đó, mạng nơ-ron nhân tạo là một mô hình mới có tính ứng dụng cao và đặc trưng là mạng nơ-ron hồi quy được sử dụng cho các bài toán dự bao chuỗi thời gian.

Sau đây chúng em sẽ đi tìm hiểu về lý thuyết của hai phương pháp tiếp cận, sử dụng chúng để dự báo chứng khoán với một độ chính xác chấp nhận được và so sánh kết quả thu được từ hai mô hình.

1.2 Giới thiệu bài toán và đối tượng nghiên cứu

Bài toán: Nhóm chúng em sẽ sử dụng hai mô hình ARIMA và LSTM để dự báo chỉ số PM-10 đo được trong không khí, sau đó so sánh kết quả hai mô hình với nhau.

1.3 Mục tiêu nghiên cứu

Đồ án này sẽ dự báo chỉ số PM-10 trong 48 giờ tiếp theo bằng mô hình ARMA và mô hình LSTM. Sau đó so sánh kết quả sai số với nhau và đánh giá.

CHƯƠNG 2. CƠ SỞ LÝ THUYẾT

2.1 Tổng quan về chuỗi thời gian

2.1.1 Chuỗi thời gian và các thành phần của chuỗi thời gian

Chuỗi thời gian là một chuỗi các phép đo được thực hiện theo thời gian, thường thu được ở các khoảng cách đều nhau, có thể là hàng ngày, hàng tháng, hàng quý hoặc hàng năm.

Phân tích chuỗi thời gian bao gồm các phương pháp để phân tích dữ liệu chuỗi thời gian để trích xuất số liệu thống kê có ý nghĩa và các đặc điểm khác của dữ liệu. Dự báo chuỗi thời gian là việc sử dụng một mô hình để dự đoán các giá trị trong tương lai dựa trên các giá trị được quan sát trước đó.

Các thành phần chính trong chuỗi thời gian là:

- Tính xu hướng (Trend)
  Tính xu hướng là yếu tố thể hiện xu hướng thay đổi của dữ liệu theo thời gian. Đây là đặc trưng thường thấy của rất nhiều dữ liệu chuỗi thời gian. Tính xu hướng cũng ảnh hưởng không nhỏ tới việc đưa ra nhận định về mối quan hệ tương quan giữa các chuỗi số. Tức là về bản chất các chuỗi không tương quan nhưng do chúng cùng có chung xu hướng theo thời gian nên chúng ta nhận định chúng là tương quan.

- Tính thời vụ (Seasonal Component)
  Tính thời vụ tồn tại khi một chuỗi hiển thị thường xuyên biến động theo mùa (ví dụ: hàng tháng / quý / năm). Tính thời vụ luôn có thời hạn cố định và được biết trước.

- Tính chu kỳ (Cyclic)
  Tính chu kỳ tồn tại khi triển lãm dữ liệu tăng và giảm không có thời hạn cố định. Độ dài trung bình của các chu kỳ dài hơn độ dài của mô hình theo

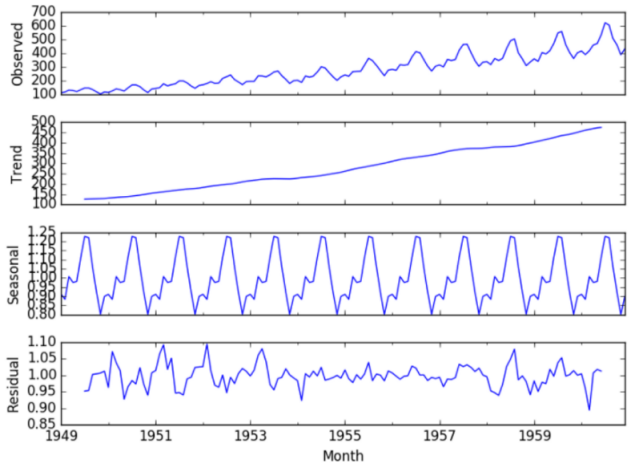
mùa. Trong thực tế, thành phần xu hướng là giả định bao gồm cả thành phần chu kỳ. Đôi khi các thành phần xu hướng và chu kỳ cùng nhau được gọi là xu hướng-chu kỳ.

- Phần dư (Residual)
  Phần dư tồn tại khi ta loại bỏ được tính xu hướng (Trend) và tính thời vụ (Seasonal Component).

Các tính năng chính của nhiều chuỗi thời gian là xu hướng và sự thay đổi theo mùa. Một đặc điểm khác của hầu hết các chuỗi thời gian là các quan sát gần nhau trong thời gian có xu hướng tương quan với nhau.

Các thành phần này kết hợp theo một cách nào đó để cung cấp chuỗi thời gian quan sát được. Ví dụ, chúng có thể được thêm vào với nhau để tạo thành một mô hình như:

$$\text{Y} = \text{Xu hướng} + \text{Mùa} + \text{Nhiễu} + \text{Chu kỳ}$$



Hình 2.1 Biểu đồ thể hiện 4 thành phần của chuỗi thời gian

Những thành phần này là cách hiệu quả nhất để đưa ra dự đoán về các giá trị trong tương lai, nhưng có thể không phải lúc nào cũng hoạt động. Điều đó phụ thuộc vào lượng dữ liệu chúng ta có về quá khứ.

2.1.2 Chuỗi dừng và hàm ACF, ACVF

Xét chuỗi thời gian $\{\mathbf{X}_t\}$ thỏa mãn $E\left(\mathbf{X}_t^2\right) < \infty$, ta định nghĩa hàm trung bình $\mu_X(t)$ và hàm hiệp phương sai $\gamma_X(r,s)$ của $\{\mathbf{X}_t\}$ như sau:

Định nghĩa. *Với* $\{\mathbf{X}_t\}$ *thỏa mãn* $E\left(\mathbf{X}_t^2\right) < \infty$*, hàm trung bình của* $\{\mathbf{X}_t\}$ *được định nghĩa bởi:*

$$\mu_X(t) = E(\mathbf{X}_t)$$

Và hàm hiệp phương sai của $\{\mathbf{X}_t\}$ *được định nghĩa bởi:*

$$\gamma_X(r,s) = Cov(\mathbf{X}_r, \mathbf{X}_s) = E[(\mathbf{X}_r - \mu_X(r))(\mathbf{X}_s - \mu_X(s))]$$

Khi đó, ta có định nghĩa về chuỗi thời gian dừng yếu:

Định nghĩa. *Chuỗi thời gian* $\{\mathbf{X}_t\}$ *được gọi là dừng yếu nếu* $\mu_X(t)$ *độc lập* t *và* $\gamma_X(t+h,t)$ *độc lập* t *với mọi* h

Lưu ý rằng tính dừng chặt của một chuỗi thời gian được xác định bởi điều kiện $(\mathbf{X}_1, \ldots, \mathbf{X}_n)$ và $(\mathbf{X}_{1+h}, \ldots, \mathbf{X}_{n+h})$ có cùng phân phối chung với mọi số nguyên h và $n > 0$. Dễ thấy rằng nếu với mọi t, $\{\mathbf{X}_t\}$ dừng chặt và $E\left(\mathbf{X}_t^2\right) < \infty$ thì $\{\mathbf{X}_t\}$ hiển nhiên cũng dừng yếu. Với chuỗi thời gian $\{\mathbf{X}_t\}$ dừng yếu, ta có định nghĩa về hàm tự hiệp phương sai $\gamma_X(h)$ và hàm tự tương quan $\rho_X(h)$ của $\{\mathbf{X}_t\}$ như sau:

Định nghĩa. *Xét* $\{\mathbf{X}_t\}$ *là một chuỗi thời gian dừng yếu. Hàm tự hiệp phương sai (ACVF) của* $\{\mathbf{X}_t\}$ *với độ trễ* h *được định nghĩa bởi:*

$$\gamma_X(h) = Cov(\mathbf{X}_{t+h}, \mathbf{X}_t)$$

Hàm tự tương quan (ACF) của $\{\mathbf{X}_t\}$ *với độ trễ* h *được định nghĩa bởi:*

$$\rho_X(h) = \frac{\gamma_X(h)}{\gamma_X(0)} = Cor(\mathbf{X}_{t+h}, \mathbf{X}_t)$$

2.1.3 Toán tử lùi

Xét chuỗi thời gian $\{\mathbf{X}_t\}$ ta định nghĩa toán tử lùi B như sau:

$$B\mathbf{X}_t = \mathbf{X}_{t-1}$$

Toán tử lùi là một toán tử tuyến tính, khả nghịch và nghịch đảo của nó là $B^{-1} = F$, được gọi là toán tử tiến, được định nghĩa bởi:

$$F\mathbf{X}_t = X_{t+1}$$

Các toán tử B, F thỏa mãn:

$$B^j\mathbf{X}_t = X_{t-j}$$
$$F^j\mathbf{X}_t = X_{t+j}$$
$$\left(\sum_{i=0}^{n} a_i B_i\right)\mathbf{X}_t = \sum_{i=0}^{n} a_i \mathbf{X}_{t-i}$$

2.2 Tiếp cận theo hướng thống kê

2.2.1 Giới thiệu mô hình AR, MA, ARMA

Mô hình autoregressive moving average (ARMA), đôi khi được gọi là mô hình Box - Jenkins, thường được áp dụng cho dữ liệu chuỗi thời gian (time series) tự tương quan (autocorrelated).

Cho chuỗi dữ liệu thời gian $\{\mathbf{X}_t\}$, mô hình ARMA là một công cụ để dự đoán các giá trị tương lai của chuỗi này. Mô hình bao gồm hai phần, phần tự hồi quy (autoregressive - AR) và phần trung bình trượt (moving average - MA). Mô hình thường được coi là mô hình ARMA(p, q) khi p là bậc của AR và q là bậc của MA.

Định nghĩa. $\{\mathbf{X}_t\}$ *là một quá trình trung bình trượt (Moving Average) cấp* q*, viết là* $\mathbf{X}_t \sim MA(q)$*, nếu thỏa mãn:*

$$\mathbf{X}_t = \mathbf{Z}_t + \theta_1 \mathbf{Z}_{t-1} + \theta_2 \mathbf{Z}_{t-2} + \ldots + \phi_q \mathbf{Z}_{t-p} \tag{2.2.1}$$

Với $\mathbf{Z}_t \sim WN\left(0, \sigma^2\right), t = 0, \pm 1, \ldots$ *và* $\theta_1, \ldots, \theta_q$ *là các hằng số.*

Có thể viết lại dưới dạng toán tử lùi:

$$\mathbf{X}_t = \theta(B)\mathbf{Z}_t \tag{2.2.2}$$

trong đó $\phi(B)$ được gọi là đa thức trung bình trượt được định nghĩa bởi:

$$\theta(B) = 1 + \theta_1 B + \theta_2 B^2 + \ldots + \theta_p B^p$$

Định nghĩa. $\{\mathbf{X}_t\}$ *là một quá trình tự hồi quy (Autoregressive) cấp* p*, viết là* $\{\mathbf{X}_t\} \sim AR(p)$*, nếu nó là một chuỗi thời gian dừng thỏa mãn:*

$$\mathbf{X}_t = \phi_1 \mathbf{X}_{t-1} + \phi_2 \mathbf{X}_{t-2} + \ldots + \phi_p \mathbf{X}_{t-p} + \mathbf{Z}_t, t = 0, \pm 1, \ldots \tag{2.2.3}$$

Với $\mathbf{Z}_t \sim WN\left(0, \sigma^2\right)$ *và* $\phi_1, \phi_2, \ldots, \phi_p$ *là các hằng số.*

Phương trình trên có thể được viết lại theo toán tử lùi:

$$\phi(B)\mathbf{X}_t = \mathbf{W}_t \tag{2.2.4}$$

Với $\phi(B) = 1 - \phi_1 B - \phi_2 B^2 - \ldots - \phi_p B^p$

Định nghĩa. $\{\mathbf{X}_t\}$ *là một quá trình* $ARMA(p, q)$ *nếu nó là một quá trình dừng thỏa mãn:*

$$\mathbf{X} - \phi_1 \mathbf{X}_{t-1} - \ldots - \phi_p \mathbf{X}_{t-p} = \mathbf{Z}_t + \theta_1 \mathbf{Z}_{t-1} + \ldots + \theta_q \mathbf{Z}_{t-q} \tag{2.2.5}$$

Với $\mathbf{Z}_t \sim WN\left(0, \sigma^2\right)$*, các đa thức* $\left(1 - \phi_1 z - \ldots - \phi_p z^p\right), \left(1 + \theta_1 z - \ldots - \theta_q z^q\right)$ *không có nhân tử chung.*

Để thuận tiện ta có thể viết ngắn gọn:

$$\phi\left(B\right)\mathbf{X}_t = \theta\left(B\right)\mathbf{Z}_t \tag{2.2.6}$$

Với $\phi\left(.\right)$ là đa thức có bậc p.

$$\phi\left(z\right) = 1 - \phi_1 z - \ldots - \phi_p z^p$$

Và $\theta\left(.\right)$ là đa thức có bậc q.

$$\theta\left(z\right) = 1 + \theta_1 z + \ldots + \theta_q z^q$$

Và B là toán tử lùi

$$B^j \mathbf{X}_t = \mathbf{X}_{t-j}, B^j \mathbf{Z}_t = \mathbf{Z}_{t-j} \text{ với } j = 0, \pm 1, \ldots$$

Chuỗi thời gian $\{\mathbf{X}_t\}$ là quá trình tự hồi quy bậc p - $AR\left(p\right)$ nếu $\theta\left(z\right) \equiv 1$, là quá trình trung bình trượt bậc q - $MA\left(q\right)$ nếu $\phi\left(z\right) \equiv 1$.

2.2.2 Giới thiệu mô hình ARIMA

Chúng ta biết rằng hầu hết các chuỗi thời gian đều có sự tương quan giữa giá trị trong quá khứ đến giá trị hiện tại. Mức độ tương quan càng lớn khi chuỗi càng gần thời điểm hiện tại. Chính vì thể mô hình ARIMA sẽ tìm cách đưa vào các biến trễ nhằm tạo ra một mô hình dự báo fitting tốt hơn giá trị của chuỗi.

Mô hình tự hồi quy tích hợp trung bình trượt (Autoregressive integrated moving average - ARIMA), hay còn gọi là mô hình Box - Jenkins, là một phương pháp nghiên cứu độc lập thông qua việc dự đoán theo các chuỗi thời gian và thường được áp dụng cho dữ liệu chuỗi thời gian tự tương quan (autocorrelated). ARIMA là sự kết hợp giữa các phương pháp tiếp cận tự hồi quy (AR) và trung bình trượt (MA) trong việc xây dựng mô hình tổng hợp của chuỗi thời gian. Mô hình này khá đơn giản, nhưng có thể cho kết quả tốt. Nó bao gồm các tham số để tính đến tính thời vụ, xu hướng dài hạn, tự hồi quy và trung bình trượt, từ đó xử lý tự tương quan được nhúng trong dữ liệu.

Mô hình ARIMA bao gồm bốn bước:

- Nhận dạng mô hình thử nghiệm: ta sẽ quan sát xem chuỗi thời gian có phải chuỗi dừng hay không, nếu chuỗi thời gian có tính mùa hay tính xu hướng thì ta sẽ thực hiện các bước để đưa chuỗi về dạng dừng.

- Ước lượng tham số: d là bậc tích hợp và p,q sẽ được xác định bằng biểu đồ tự tương quan, thông qua nghiên cứu chiều hướng biến đổi của hàm tương quan toàn phần hay một phần. Cụ thể, p sẽ là bậc của đồ thị AR. Xét từ độ trễ đầu tiên, thanh nào nằm ngoài đường giới hạn và sau độ giảm một cách đáng kể sau một độ trễ thì hệ số tự tương quan riêng phần đó là p. Tương tự, q sẽ là bậc của MA.

- Kiểm định bằng chẩn đoán: Sau khi các tham số của mô hình tổng quát đã được xây dựng, ta sẽ kiểm tra mức độ chính xác và sự phù hợp của mô hình với dữ liệu đã lập, xem xét phần sai số có phải ngẫu nhiên thuần túy hay không. Nếu có thì mô hình đó thỏa mãn, nếu không thì ta sẽ phải thực hiện lại các bước trên.

- Dự báo: Ở bước cuối cùng này, khi mô hình phù hợp với dữ liệu đã tìm được, ta sẽ thực hiện dự báo tại thời điểm tiếp theo.

Mô hình ARIMA giúp dự báo với độ tin cậy cao hơn từ các phương pháp lập mô hình kinh tế lượng truyền thống, đặc biệt đối với dự báo ngắn hạn.

2.2.3 Mô hình ARIMA(p,d,q)

Mô hình ARIMA sẽ biểu diễn phương trình hồi quy tuyến tính đa biến của các biến đầu vào là 3 thành phần chính:

- Auto regression: Kí hiệu là AR. Đây là thành phần tự hồi quy bao gồm tập hợp các độ trễ của biến hiện tại. Mô hình AR có thể được biểu diễn như sau:

$$y_t = c + \phi_1 y_{t-1} + \epsilon_t \tag{2.2.7}$$

Trong đó: y_t là giá trị tại thời gian t, c là hằng số, Φ_1 là hệ số còn ϵ_t là nhiễu trắng với $\epsilon_t \sim N(0, \sigma^2)$.

Độ trễ bậc p chính là giá trị lùi về quá khứ p bước thời gian của chuỗi. Độ trễ dài hoặc ngắn trong quá trình AR phụ thuộc vào tham số trễ p. Mô hình AR(p) của chuỗi y_t được biểu diễn như sau:

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} ... + \phi_p y_{t-p} + \epsilon_t \tag{2.2.8}$$

hay

$$y_t = c + \sum_{i=1}^{p} \phi_i y_{t-i} \tag{2.2.9}$$

Trong đó Φ_i là hệ số tương ứng của mỗi giá trị $y_t - i$.

- Moving average: Quá trình trung bình trượt được hiểu là quá trình dịch chuyển hoặc thay đổi giá trị trung bình của chuỗi theo thời gian. Do chuỗi của chúng ta được giả định là dừng nên quá trình thay đổi trung bình dường như là 1 chuỗi nhiễu trắng. Quá trình moving average sẽ tìm mối liên hệ về mặt tuyến tính giữa các phần tử ngẫu nhiên ϵ_t chuỗi này là 1 chuỗi nhiễu trắng có các tính chất:

$$E(\epsilon_t) = 0 \tag{2.2.10}$$

$$\sigma(\epsilon_t) = \alpha \tag{2.2.11}$$

$$\rho(\epsilon_t, \epsilon_{t-s)} = 0, \forall t \geq s \tag{2.2.12}$$

Vế (2.2.10) có nghĩa rằng kì vọng của chuỗi dừng bằng 0 để đảm bảo chuỗi dừng không có sự thay đổi về trung bình theo thời gian. Vế (2.2.11) là phương sai của chuỗi không đổi. Do kì vọng và phương sai không đỏi nên chúng tagọi phân phối của nhiễu trắng là phân phối xác định và được kí hiệu $\epsilon_t \sim \mathcal{WN}(\ 0, \sigma^2)$. Nhiễu trắng là một thành phần ngẫu nhiên thể hiện cho yếu tố không thể dự báo của model và không có tính quy luật. Quá trình trung bình trượt được biểu diễn theo nhiễu trắng như sau:

$$MA(q) = \mu + \sum_{i=1}^{q} \theta_i \epsilon_{t-i} \tag{2.2.13}$$

- Intergrated: Là quá trình đồng tích hợp hoặc lấy sai phân. Yêu cầu chung của các thuật toán trong time series là chuỗi phải đảm bảo tính dừng. Hầu hết chuỗi đều tăng hoặc giảm theo thời gian. Do đó yếu tố tương quan giữa chúng chưa chắc là thực sự mà là do chúng cùng tương quan theo thời gian. Khi biến đổi sang chuỗi dừng, các nhân tố ảnh hưởng thời gian được loại bỏ và chuỗi sẽ dễ dự báo hơn. Để tạo thành chuỗi dừng, một phương pháp đơn giản nhất là chúng ta sẽ lấy sai phân. Một số chuỗi tài chính còn qui đổi sang logarit hoặc lợi nhuận. Bậc của sai phân để tạo thành chuỗi dừng

còn gọi là bậc của quá trình đồng tích hợp (order of intergrated). Quá trình sai phân bậc d của chuỗi được thực hiện như sau:

$$\text{Sai phân bậc } 1 : I(1) = \Delta(x_t) = x_t - x_{t-1}$$

$$\text{Sai phân bậc } 2 : I(d) = \Delta^d(x_t) = \underbrace{\Delta(\Delta(...\Delta(x_t)))}_{d\,times}$$

- Thông thường chuỗi sẽ dừng sau quá trình đồng tích hợp I(0) hoặc I(1). Rất ít chuỗi chúng ta phải lấy tới sai phân bậc 2. Một số trường hợp chúng ta sẽ cần biến đổi logarit hoặc căn bậc 2 để tạo thành chuỗi dừng. Phương trình ARIMA(p,d,q) có thể được biểu diễn dưới dạng:

$$\Delta x_t = \phi_1 \Delta x_{t-1} + \phi_2 \Delta x_{t-2} + ...\phi_p \Delta x_{t-p} + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + ... + \theta_q \epsilon_{t-q} \quad (2.2.14)$$

Trong đó Δ_{x_t} là giá trị sai phân bậc d và ϵ_t là các chuỗi nhiễu trắng.

2.2.4 Cách xác định hệ số p, d, q

Cách xác định hệ số sai phân d trong mô hình ARIMA

Mục đích của sai phân trong mô hình ARIMA là làm cho chuỗi thời gian dừng.

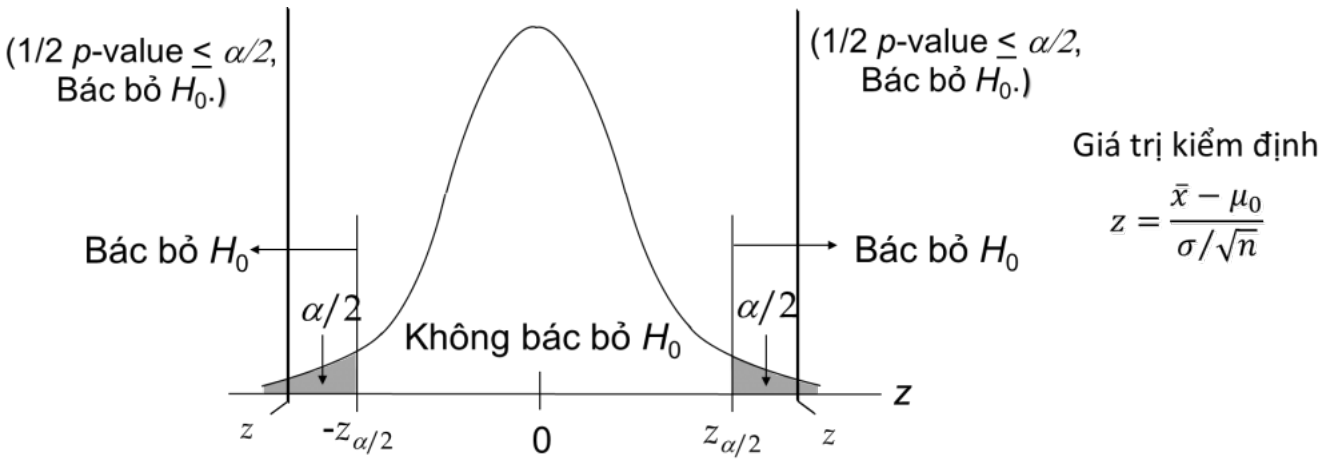
Thứ tự sai phân phù hợp là độ lệch tối thiểu cần thiết để cho một chuỗi thời gian dừng biến đổi xung quanh một giá trị kỳ vọng xác định và đồ thị của ACF đạt đến giá trị 0 khá nhanh.

Nếu tự tương quan dương với nhiều độ trễ (10 hoặc nhiều hơn), thì chuỗi đó cần phải sai phân thêm. Mặt khác, nếu chính tự tương quan với độ trễ 1 quá âm, thì chuỗi có thể bị sai phân quá mức.

Trong trường hợp, chúng ta thực sự không thể quyết định giữa hai thứ tự sai lệch, thì hãy chọn thứ tự cho độ lệch chuẩn nhỏ nhất trong chuỗi sai lệch. Đầu tiên, để xác định hệ số d cho mô hình ta cần xác định chuỗi đã cho dừng hay chưa, vì nếu chuỗi đã dừng thì ta không cần đến giá trị sai phân (d=0) còn chưa dừng thì ta cần phải đi xác định hệ số d.

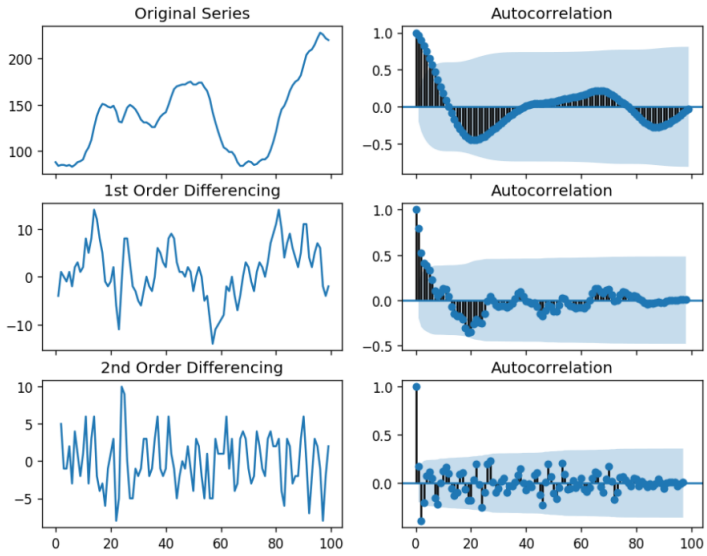
Ở đây, chúng em sử dụng Augmented Dickey Fuller test (adfuller()), trong gói statsmodels của python để kiểm định xem chuỗi đã dừng hay chưa.

Giả thiết không của kiểm định ADF là chuỗi thời gian không dừng. Vì vậy nếu giá trị p-value nhỏ hơn 0.05 thì bác bỏ giả thiết không và kết luận rằng chuỗi thời gian dừng.



Hình 2.2 Kiểm định tham số p

Với trường hợp p-value lớn hơn mức ý nghĩa ta sẽ đánh giá dựa trên đồ thị ACF.



Hình 2.3 Ví dụ cách xác định d

Đối với chuỗi trong ví dụ trên, chuỗi thời gian đạt đến trạng thái ổn định với hai bậc chênh lệch. Nhưng khi nhìn vào biểu đồ tự tương quan cho độ chênh lệch thứ 2, độ trễ đi vào vùng âm xa khá nhanh, điều này cho thấy, chuỗi có thể đã bị sai lệch quá mức.

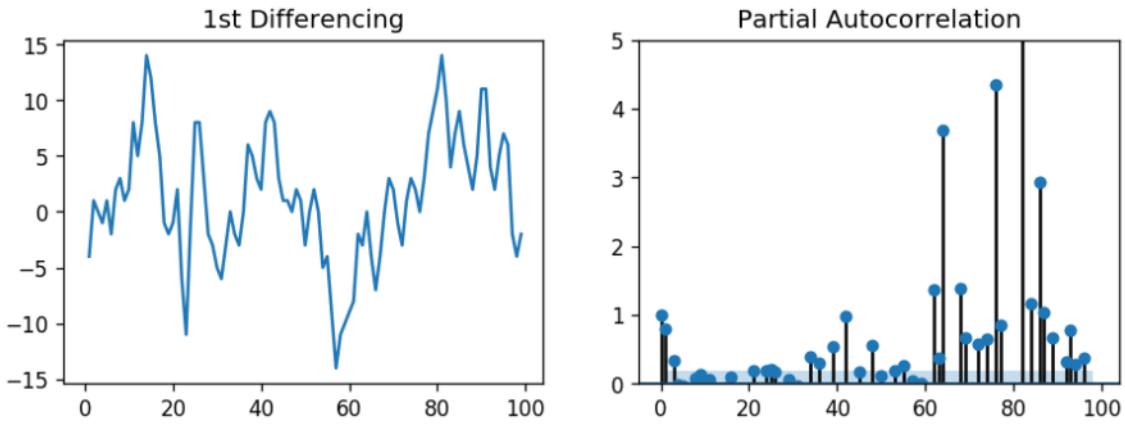
Vì vậy, tạm thời ta sẽ chọn thứ tự sai phân là 1 mặc dù loạt chuỗi không hoàn toàn dừng (tính ổn định yếu).

Cách xác định hệ số p của AR

Để kiểm tra hệ số q của mô hình AR ta kiểm tra biểu đồ tự tương quan 1 phần (PACF).

Tự tương quan một phần có thể được hình dung như mối tương quan giữa chuỗi và độ trễ của nó, sau khi loại trừ các đóng góp từ độ trễ trung gian. Vì vậy, PACF truyền tải mối tương quan thuần túy giữa độ trễ và chuỗi. Bằng cách đó, ta sẽ biết liệu độ trễ đó có cần thiết trong điều kiện AR hay không.

Bất kỳ sự tự tương quan nào trong một chuỗi dừng đều có thể được điều chỉnh bằng cách thêm đủ các thuật ngữ AR. Vì vậy, ban đầu ta coi thứ tự của số hạng AR bằng với càng nhiều độ trễ vượt qua giới hạn ý nghĩa trong biểu đồ PACF.

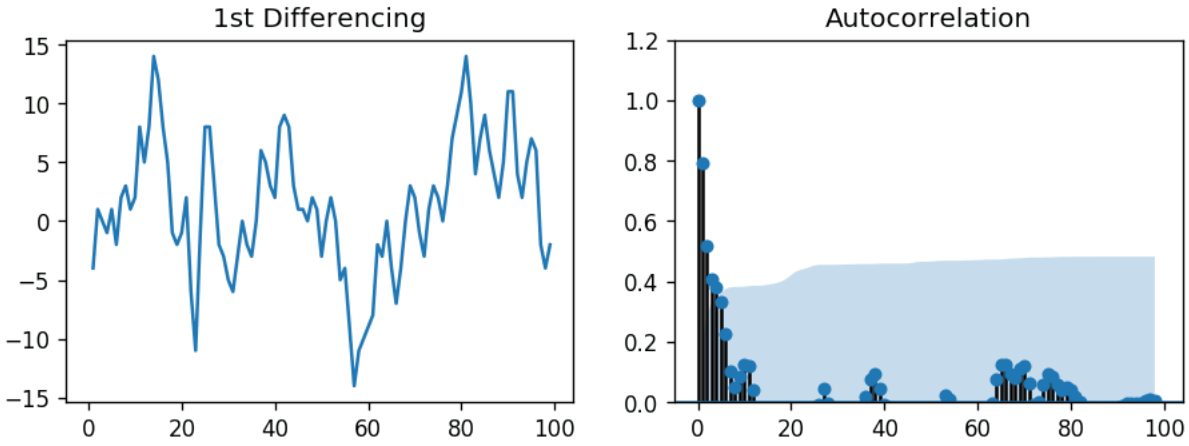


Hình 2.4 Ví dụ cách xác định p

Có thể quan sát thấy rằng độ trễ PACF bằng 1 là khá đáng kể vì nằm trên đường ý nghĩa (vùng màu xanh lam). Vì vậy ta chọn p bằng 1.

Cách xác định hệ số q của MA

Cũng giống như cách xem xét biểu đồ PACF cho hệ số của AR, chúng ta có thể xem biểu đồ ACF để biết hệ số của MA.



Hình 2.5 Ví dụ cách xác định q

Biểu đồ ACF cho ta biết cần phải thực hiện quá trình MA bao nhiêu lần để loại bỏ sự tự tương quan trong chuỗi thời gian dừng. Ở đây ta thấy nhiều lag nằm trên đường ý nghĩa, ta có thể chọn giá trị q là 1 hoặc 2.

2.2.5 Giới thiệu mô hình SARIMA

Lớp mô hình ARMA có vai trò quan trọng trong việc mô tả các chuỗi thời gian dừng. Để mô tả cả các chuỗi thời gian không dừng, người ta tổng quát hóa lớp mô hình ARMA thành mô hình trung bình trượt tích hợp tự hồi quy, hay ARIMA. Trước hết ta định nghĩa tính nhân quả của một mô hình ARMA như sau:

Định nghĩa. *Xét chuỗi thời gian* $\{\mathbf{X}_t\}$ *tuân theo mô hình* $ARMA(p,q)$. $\{\mathbf{X}_t\}$ *có tính nhân quả nếu tồn tại một dãy* $\psi_j, j \in \mathbb{N}_0$ *sao cho:*

$$\sum_{j=1}^{\infty} |\psi_j| < \infty$$

Và:

$$\mathbf{X}_t = \sum_{j=1}^{\infty} \psi_j \mathbf{Z}_{t-j}, t \in \mathbb{Z}$$

Mô hình ARIMA có dạng tổng quát như sau:

$$\phi_p(B)(1-B)^d \mathbf{X}_t = \theta_q(B)\mathbf{Z}_t$$

Với:

$$\phi_p(B) = 1 - \phi_1 B - \ldots - \phi_p B^p$$
$$\theta_q(B) = 1 - \theta_1 B - \ldots - \theta_q B^q$$

Định nghĩa. *Với một số nguyên không âm d, chuỗi thời gian $\{\mathbf{X}_t\}$ là một quá trình trung bình trượt tích hợp tự hồi quy bậc d, p, q hay $ARIMA(p,d,q)$ nếu $\mathbf{Y}_t := (1-B)^d \mathbf{X}_t$ là một quá trình $ARMA(p,q)$ có tính nhân quả.*

Mô hình SARIMA được phát triển tiếp từ mô hình ARIMA phù hợp với bất kỳ dữ liệu chuỗi thời gian mùa vụ nào, có thể là 4 quý trong năm, 7 ngày trong tuần, 11 hoặc 12 tháng trong một năm, ... Nếu chuỗi dữ liệu quan sát có tính mùa vụ, thì mô hình ARIMA tổng quát lúc này là $SARIMA(p,d,q)(P,D,Q)$ với (p,d,q) đại diện cho phần mô hình không có tính mùa. (P,D,Q) đại diện cho phần mô hình có tính mùa. Giá trị s đại diện cho số bước thời gian của một mùa. Ví dụ như với chuỗi thời gian theo từng tháng, thì s có thể bằng 12, hoặc bằng 4. Giá trị d và D thể hiện số lần lấy sai phân, nhằm giúp cho chuôi thời gian thành chuỗi dừng.

Ta có công thức tổng quát của mô hình SARIMA như sau:

$$\phi_p(B)\,\Phi_p(B)\,(1-B)^d\,(1-B^s)^D\,\mathbf{X}_t = \theta_q(B)\,\Theta_Q\left(B^S\right)\mathbf{Z}_t$$

Trong đó:

$$\Phi_p(B^s) = 1 - \Phi_1 B^s - \ldots - \Phi_p B^{sp}$$
$$\Theta_Q(B^s) = 1 - \Theta_1 B^s - \ldots - \Theta_p B^{sp}$$

Chú ý. Quá trình $\{\mathbf{X}_t\}$ là quá trình nhân quả khi và chỉ khi $\phi(z) \neq 0$ và $\theta(z) \neq 0$ với $|z| \leq 1$. D hiếm khi lớn hơn 1, P và Q thường nhỏ hơn 3.

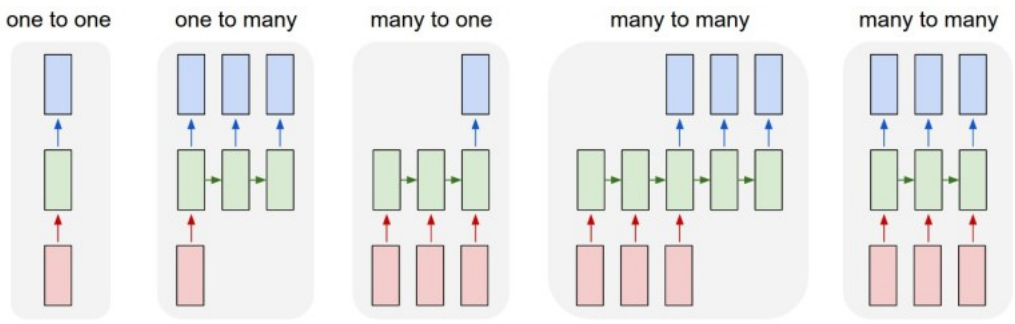
2.3 Tiếp cận theo hướng Machine Learning

2.3.1 Tổng quan về RNN

Chúng ta sẽ đi tìm hiểu qua về RNN, có thể hiểu đơn giản RNN sinh ra để xử lí các bài toán có dạng chuỗi tuần tự, ví dụ như các ảnh được tách từ video, dữ liệu time series.

Dựa theo số lượng chuỗi đầu ra và đầu vào, ta phân loại các bài toán RNN thành 4 dạng:

- One to one: đây là dạng bài toán như Convolutional Neural Network (CNN) và Neural Network (NN), 1 input chúng ta sẽ có 1 output, ví dụ với một ảnh thì ta có output là nhãn ảnh đó.
- One to many: dạng toán có 1 input và nhiều output, ví dụ đầu vào một ảnh và đầu ra là phân tích cho ảnh đó bằng chữ.



Hình 2.6 Các dạng bài toán RNN

- Many to one: dạng toán nhiều output và có 1 input, ví dụ điển hình là video là kết hợp của nhiều ảnh, và các ảnh đó khi kết hợp có thứ tự thì mô tả một hành động như người đang nhảy dây, và output là hành động của người đó.
- Many to many: bài toán có nhiều input nhưng có nhiều output, ví dụ dạng toán dịch từ, "I love you "thành "Anh yêu em".

2.3.2 Ứng dụng bài toán RNN

Các ứng dụng của RNN có thể kể đến như:

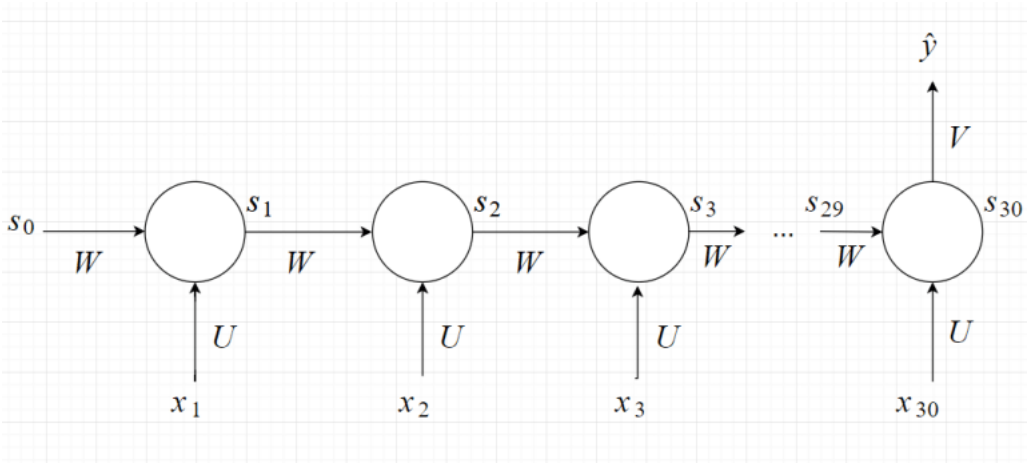
- Dự báo, phân loại các bài toán time series
- Chuyển giọng nói thành văn bản
- Phân loại cảm xúc trong văn bản
- Nhận diện hành động trong video
- Dịch tự động các ngôn ngữ

2.3.3 Mô hình bài toán RNN

Chúng ta sẽ lấy một ví dụ cụ thể, ở đây là bài toán nhận diện hành động trong video, mỗi giây chúng ta có 30 ảnh.

Ta có:

- ta có thể thấy có 30 input, 1 output, các input là $x_1, x_2, x_3, ..x_{30}$



Hình 2.7 Mô hình bài toán RNN

- mỗi state là một ô tròn, 1 state có input là x_t và s_{t-1}(là output của state trước), output là $s_t = f(U * x_t + W * s_{t-1})$. F là hàm kích hoạt, hàm kích hoạt có thể dùng ReLu hoặc tanh
- s_t mang cả thông tin của state hiện tại và các state trước, nên output cuối cùng sẽ mang đầy đủ thông tin của các input.

Các vector:

- x_i là vector kích thước n*1.
- s_i là vector kích thước m*1.
- 0_i là vector kích thước d*1

Các ma trận weight chung cần tìm cho cả quá trình:

- U là ma trận có kích thích m*n.
- W là ma trận có kích thước m*m.
- V là ma trận có kích thước d*m.

Các hàm:

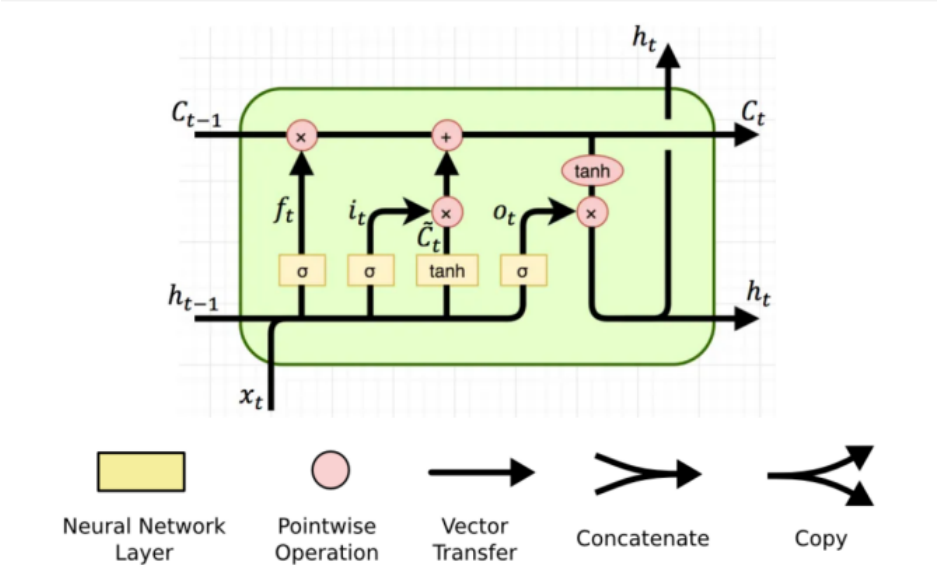
- $s_0 = 0, s_t = f(U * x_t + W * s_{t-1})$
- $o_t = g(V * s_t)$
- V là ma trận có kích thước d*m.

Loss function Loss function của bài toán bằng tổng loss của output, do mô hình có 1 output nên sử dụng cross entropy loss.

2.3.4 Tổng quan về LSTM

LSTM là mạng cải tiến của RNN nhằm giải quyết các vấn đề nhớ các bước dài của RNN, có thể nhớ các thông tin trong suốt thời gian dài là 1 đặc tính của nó.

Tùy thuộc vào khoảng cách tới thông tin trước đó để đưa ra dự đoán mà RNN cho ra các kết quả khác nhau. Với khoảng cách lớn dần thì RNN bắt đầu không nhớ được và học được nữa.



Hình 2.8 Mô hình LSTM

- Trong mỗi modul thì nó có 4 neural network layer tương tác với nhau, biểu diễn qua ô chữ nhật màu vàng, chứa các activation function.
- Các hình tròn màu hồng biểu diễn các phép toán học. Gồm phép cộng ma trận, phép nhân là phép element-wise multiplication.
- Vector transfer nối đầu ra của nút này làm đầu vào của nút tiếp theo.
- Các đường hợp với nhau kí hiệu sự kết hợp.
- Các đường rẽ nhánh chỉ nội dung được sao chép và chuyển đi những nơi khác.

2.3.5 Input và Output

Input

- C_{t-1}, h_{t-1}là output của module trước.
- x_t là input của module thứ t

Output
Cell state C_t, hidden state h_t.
Output của LSTM thêm thành phần Ct so với RNN.
Các hàm

- Forget gate $f_t = \sigma(U_f * x_t + W_f * h_{t-1} + b_f)$
- Input gate $i_t = \sigma(U_i * x_t + W_i * h_{t-1} + b_i)$
- Output gate $o_t = \sigma(U_o * x_t + W_o * h_{o-1} + b_o)$

Nhận xét

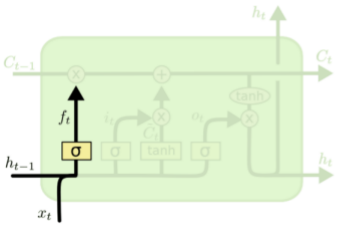
- $0 < f_t, i_t, o_t < 1$
- b_f, b_i, b_o là các bias.
- U là ma trận có kích thích m*n, W là ma trận có kích thước m*m.

2.3.6 Ý tưởng chính

- Nó được cấu tạo bởi cell state. Nó chạy qua tất cả các mắt xích và có 1 vài tương tác tuyến tính với 3 cổng(gate) để thêm bớt thông tin cần thiết.
- Các cổng dùng để sàng lọc thông tin, tạo bởi 1 tầng sigmoid và một phép nhân.
- Tầng sidmoid cho đầu ra là số trong khoảng [0,1], mô tả lượng thông tin có thể thông qua. Là 0 nếu tất loại bỏ tất cả thông tin, 1 nếu các thông tin đều được thông qua.

2.3.7 Bên trong mạng LSTM

- Thứ nhất, LSTM xem xét các thông tin cần loại bỏ từ cell state. Quyết định loại bỏ do tầng sidmoid quyết định. Nó lấy input là h_{t-1} và x_t rồi cho kết quả trong đoạn [0,1].



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

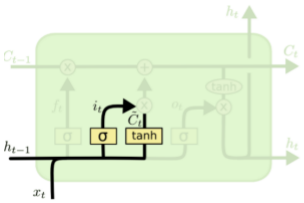
- Thứ hai, quyết định thông tin mới nào sẽ lưu vào cell state.

- Ta cần qua 2 bước:

Bước 1: Sử dụng một tầng sigmoid để quyết định giá trị nào sẽ cập nhập.

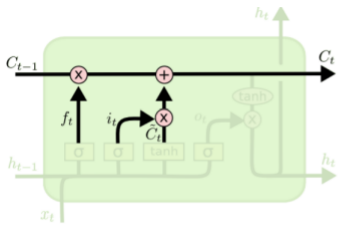
Bước 2: Tầng tanh tạo ra một vector cho giá trị mới $\tilde{C}_t$ nhằm thêm vào cho state.

- Sau đó, ta sẽ kết hợp 2 giá trị đó lại để tạo ra một cập nhập cho state.



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

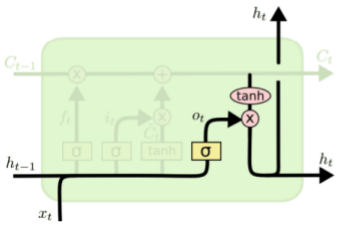
- Thứ ba, cập nhật cell state cũ C_{t-1} thành trạng thái mới C_t. Nhân trạng thái cũ với f_t để bỏ đi những thông tin ta quyết định loại bỏ trước đó và cộng với $i_t * \tilde{C}_t$. rạng thái mới phụ thuộc vào giá trị cập nhật ở bước trước.



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

- Cuối cùng, xác định đầu ra mong muốn. Giá trị đầu ra sẽ phụ thuộc vào cell state nhưng vẫn được lọc.

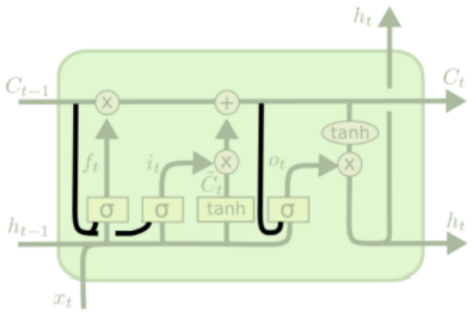
- Chạy tầng sigmoid để quyết định phần xuất ra của cell state.
- Đưa cell state qua hàm tanh để co giá trị nó về [-1,1] và nhân nó với đầu ra của tầng sigmoid.



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

2.3.8 Một số dạng biến thể phổ biến của LSTM

Dạng LSTM thêm một số đường liên kết, các tầng cổng nhận input là các giá trị của cell state.



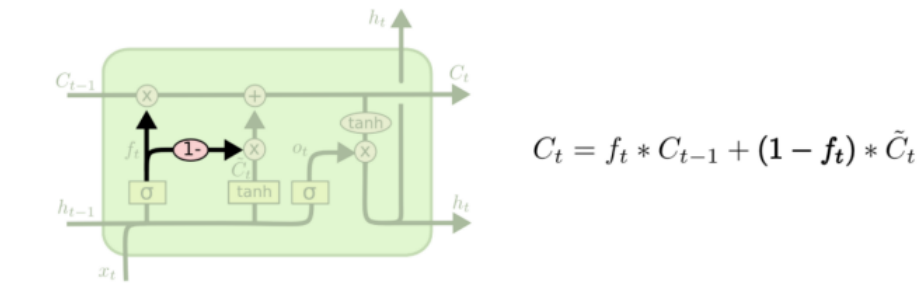
Hình 2.9 Dạng biến thể LSTM

Dạng LSTM nối 2 cổng loại trừ và đầu vào với nhau

- Thay vì phân tách làm 2 bước loại trừ và thêm mới thông tin thì ta gộp chúng lại.

- Chỉ loại bỏ thông tin khi ta thay nó bằng thông tin mới đưa vào.

- Chỉ thêm thông tin mới vào khi ta bỏ thông tin cũ đi.



Hình 2.10 Dạng biến thể LSTM

CHƯƠNG 3. CÀI ĐẶT VÀ ĐÁNH GIÁ KẾT QUẢ

3.1 Mô tả dữ liệu

Để so sánh độ chính xác của mô hình ARMIA và LSTM, chúng em sẽ thực nghiệm trên dữ liệu thời tiết chỉ số PM-10 từ ngày 14/04/2019 đến ngày 05/09/2019:

- Thời gian của dữ liệu: 3414 giờ từ ngày 14/04/2019 đến 05/09/2019.
- Kích thước dữ liệu: 76.0 Kb
- Giá trị cần dự đoán: chỉ số PM-10.
- Số ô dữ liệu trống: 70
- Phương pháp thay thế : điền bằng 0
- Tỉ lệ chia tập train, test: 0.8 : 0.2

3.2 Các metrics đánh giá

3.2.1 Root Mean Squared Error (RMSE)

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=0}^{n}(y_i - \hat{y}_i)^2} \tag{3.2.1}$$

với y_i là giá trị thực sự cần dự đoán, và $\hat{y}_i$ là giá trị mô hình dự đoán, n là kích thước của dữ liệu cần dự đoán.

3.2.2 Mean Absolute Error (MAE)

MAE là một phương pháp đo lường sự khác biệt giữa hai biến liên tục. Giả sử rằng $\hat{y}_i$ và y_i là hai biến liên tục thể hiện kết quả dự đoán của mô hình và kết quả thực tế. chúng ta có độ đo MAE được tính theo công thức sau:

$$\text{MAE} = \frac{1}{n}\sum_{i=0}^{n}|y_i - \hat{y}_i| \tag{3.2.2}$$

3.2.3 Mean Absolute Percent Error (MAPE)

Sai số tương đối mà một dự báo mắc phải có thể được đo lường bằng phần trăm sai số tuyệt đối trung bình (MAPE). MAPE được tính theo công thức sau:

$$\text{MAPE} = \frac{1}{n}\sum_{i=0}^{n}\frac{|y_i - \hat{y}_i|}{y_i} \tag{3.2.3}$$

MAPE phản ánh giá trị dự báo sai khác bao nhiêu phàn trăm so với giá trị trung bình.

3.3 Mô hình SARIMA

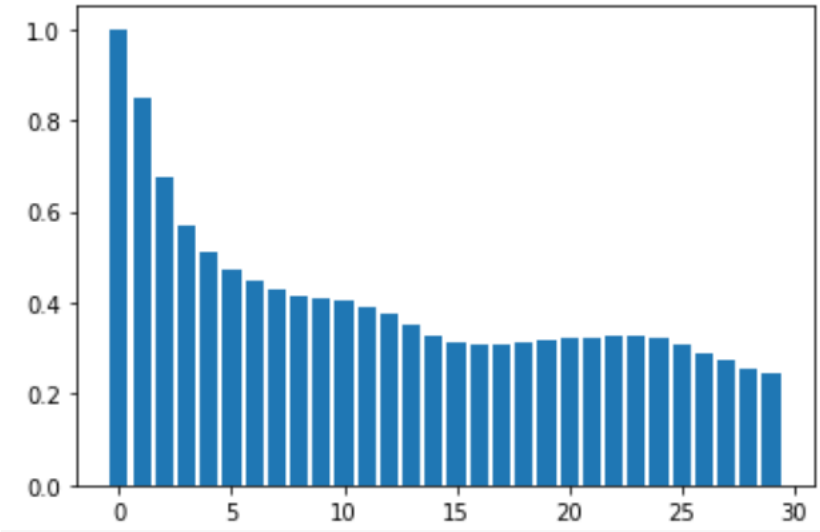
Đầu tiên ta xét xem chuỗi đã là chuỗi dừng chưa. Thực hiện lệnh để tìm chỉ số d:

```
ndiffs(df_sarima, test="adf")
```

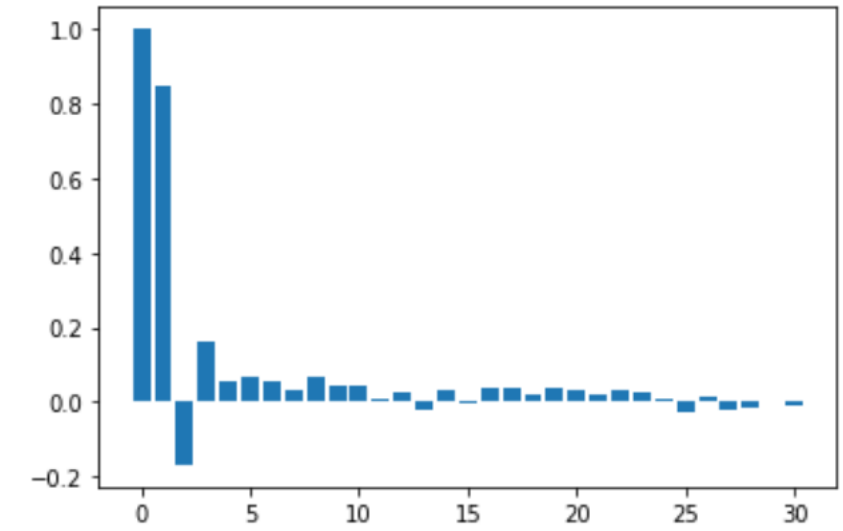
```
0
```

Sau đó, ta xét hàm ACF và PACF để tìm chỉ số p và q:

```
<BarContainer object of 30 artists>
```




```
<BarContainer object of 31 artists>
```



Sau đó, fit dữ liệu với model, dùng phương pháp inovation để tìm tham số.

```
step = len(test)
order = (10, 0, 1)
model = ARIMA(train, order=order)
result = model.fit(method='innovations_mle')

# Forecast
fc=result.forecast(steps=step)
print(fc)
```

3.4 Mô hình LSTM

Quá trình đào tạo sẽ được thực hiện trên Google Colab. Sau đó cài đặt các thư viện cần thiết, trong đồ án này, chúng ta sẽ sử dụng Tensorflow để cài đặt LSTM.

Mục tiêu của đào tạo là sau khi mô hình đã học được từ dữ liệu, thì loss của mô hình phải giảm, accuracy phải tăng

Xây dựng mô hình

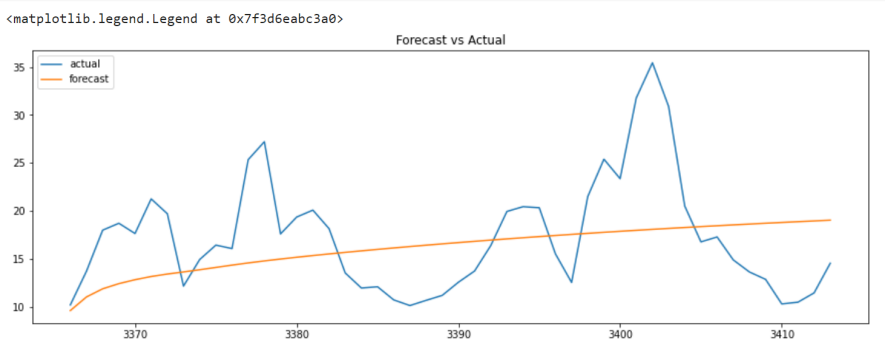
```
def lstm_model(X_train, Y_train):
  regressor = Sequential()
  regressor.add(LSTM(units = 50, return_sequences = True, input_shape = (X_train.shape[-2], X_train.shape[-1])))
  regressor.add(Dropout(0.2))
  regressor.add(LSTM(units = 50, return_sequences = True))
  regressor.add(Dropout(0.2))
  regressor.add(LSTM(units = 50, return_sequences = True))
  regressor.add(Dropout(0.2))
  regressor.add(LSTM(units = 50))
  regressor.add(Dropout(0.2))
  regressor.add(Dense(units = Y_train.shape[-1]))
  regressor.compile(optimizer = 'adam', loss = 'mean_squared_error')
  regressor.summary()


  return regressor
```

```
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_7 (LSTM)               (None, 48, 50)            10400

 dropout (Dropout)           (None, 48, 50)            0

 lstm_8 (LSTM)               (None, 48, 50)            20200

 dropout_1 (Dropout)         (None, 48, 50)            0

 lstm_9 (LSTM)               (None, 48, 50)            20200

 dropout_2 (Dropout)         (None, 48, 50)            0

 lstm_10 (LSTM)              (None, 50)                20200

 dropout_3 (Dropout)         (None, 50)                0

 dense_1 (Dense)             (None, 48)                2448

=================================================================
Total params: 73,448
Trainable params: 73,448
Non-trainable params: 0
```

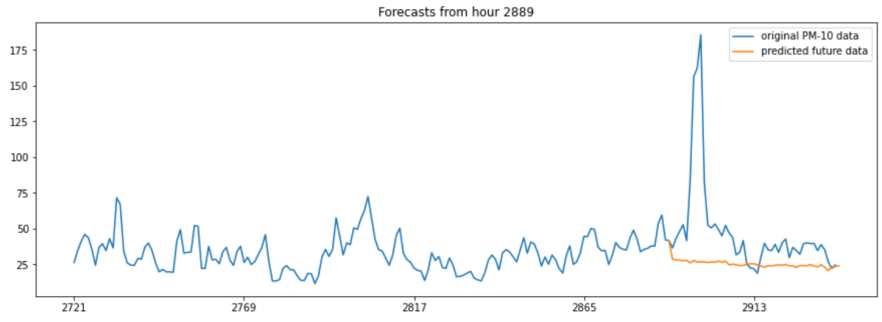
```
17/17 [==============================] - 4s 213ms/step - loss: 0.0027 - val_loss: 0.0035
Epoch 39/50
17/17 [==============================] - 3s 205ms/step - loss: 0.0027 - val_loss: 0.0035
Epoch 40/50
17/17 [==============================] - 4s 269ms/step - loss: 0.0027 - val_loss: 0.0035
Epoch 41/50
17/17 [==============================] - 4s 214ms/step - loss: 0.0026 - val_loss: 0.0035
Epoch 42/50
17/17 [==============================] - 4s 208ms/step - loss: 0.0026 - val_loss: 0.0037
Epoch 43/50
17/17 [==============================] - 5s 274ms/step - loss: 0.0026 - val_loss: 0.0035
Epoch 44/50
17/17 [==============================] - 4s 210ms/step - loss: 0.0026 - val_loss: 0.0035
Epoch 45/50
17/17 [==============================] - 3s 205ms/step - loss: 0.0026 - val_loss: 0.0034
Epoch 46/50
17/17 [==============================] - 4s 264ms/step - loss: 0.0026 - val_loss: 0.0034
Epoch 47/50
17/17 [==============================] - 4s 212ms/step - loss: 0.0026 - val_loss: 0.0035
Epoch 48/50
17/17 [==============================] - 3s 206ms/step - loss: 0.0026 - val_loss: 0.0035
Epoch 49/50
17/17 [==============================] - 4s 211ms/step - loss: 0.0026 - val_loss: 0.0036
Epoch 50/50
17/17 [==============================] - 4s 257ms/step - loss: 0.0026 - val_loss: 0.0035
```

Sau khi qua 50 epoch, mỗi epochs mất trung bình 3s, loss và val loss đã giảm đi đáng kể, đồng thời accuracy cũng tăng lên.

3.5 Kết quả



(a) Mô hình ARIMA



(b) Mô hình LSTM

Hình 3.1 So sánh kết quả

	MSE	MAE
ARIMA	326.97	11.66
LSTM	140.45	7.99

Bảng 3.1 So sánh kết quả mô hình ARIMA và LSTM trên tập test

3.6 Nhận xét

Thuận lợi

- Lượng dữ liệu đủ lớn để có thể đào tạo mô hình, để mô hình có thể học được một cách khái quát nhất.
- Với sự phát triển không ngừng của AI, các thư viện được cập nhật và nâng cấp hàng ngày, giúp quá trình thực hiện xây dựng mô hình đơn giản hơn.

Khó khăn

- Lượng dữ liệu mặc dù lớn nhưng thiếu rất nhiều. Vì vậy ảnh hưởng đến chất lượng dự báo
- Năng lực các thành viên còn chưa được tốt nên, các thành viên đều mới bắt đầu tìm hiểu về AI, deeplearning nên trong quá trình tìm hiểu bài toán và mô hình còn gặp nhiều khó khăn, cả về lí thuyết và code.

CHƯƠNG 4. KẾT LUẬN

Trong phạm vi nội dung của đồ án, một số nội dung mà nhóm chúng em đã đạt được:

- Thành công trong việc giới thiệu 2 mô hình dự đoán chuỗi thời gian ARIMA và LSTM.
- Ứng dụng được vào bài toán dự đoán thời tiết.
- So sánh được hiệu quả của mô hình ARIMA và LSTM trong dự báo giá thời tiết.

Với những kết quả đạt được, đồ án có nhiều tiềm năng ứng dụng trong nhiều bài toán khác nhau về chuỗi thời gian. Một số hướng phát triển tiếp theo của đồ án:

- Cải thiện độ chính xác của mô hình và ứng dụng vào nhiều lĩnh vực khác nhau, ví dụ: dự báo tỉ lệ người nhiễm covid, dự đoán doanh số bán hàng, v.v...
- Phát triển mô hình Online ARIMA [1] để cập nhật thêm thông tin trên tập test, cải thiện độ chính xác của mô hình.
- Phát triển thêm các biến thể của LSTM như Gated Recurrent Unit (GRU) [2], Depth Gated RNNs [3] hay Clockwork RNNs [4]

Tài liệu tham khảo

[1] C. Liu, S. C. Hoi, P. Zhao, and J. Sun, "Online arima algorithms for time series prediction," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30, Feb. 2016.

[2] K. Cho, B. van Merrienboer, Ç. Gülçehre, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," *CoRR*, vol. abs/1406.1078, 2014.

[3] K. Yao, T. Cohn, K. Vylomova, K. Duh, and C. Dyer, "Depth-gated recurrent neural networks," 08 2015.

[4] J. Koutník, K. Greff, F. J. Gomez, and J. Schmidhuber, "A clockwork RNN," *CoRR*, vol. abs/1402.3511, 2014.