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

Nhóm 19
Phan Tiến Đạt - 20195854
Dương Đình Văn - 20206185
Nguyễn Ngọc Quang - 20185395
Giảng viên hướng dẫn: TS. Nguyễn Thị Ngọc Anh

Hanoi, 3/2023

LSTM&ARIMA trong dự báo ô nhiễm môi trường

LSTM&ARIMA trong dự báo ô nhiễm môi trường

Giới thiệu bài toán

- Sử dụng hai mô hình ARIMA và LSTM để dự báo chỉ số PM-10 đo được trong không khí
- So sánh kết quả của hai mô hình với nhau
- Đối tượng nghiên cứu: Chỉ số PM-10
- Phạm vi nghiên cứu: từ ngày 14/4/2019 tới ngày 5/9/2019.

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.

LSTM&ARIMA trong dự báo ô nhiễm môi trường

Giới thiệu mô hình ARIMA

Mô hình tự hồi quy tich hợp trung binh trươt (Autoregressive integrated moving average):

- Còn được gọi là mô hình Box-Jenkins
- Là một phương pháp nghiên cứu độc lập thông qua việc dự đoán theo các chuỗi thời gian
- Thường được áp dụng cho dữ liệu chuỗi thời gian tự tương quan
- 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

Giới thiệu mô hình ARIMA

Bao gồm 4 bước:

- Nhận dạng mô hình thử nghiệm
- Ước lượng tham số
- Kiểm định bằng chẩn đoán
- Dự báo

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.

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 (AR): $y_t = c + \sum_{i=1}^{p} \phi_i y^{t-i}$
- Moving average (MA): $MA(q) = \mu + \sum_{i=1}^{q} \theta_i \epsilon_{t-i}$
- Intergrated

Auto regression

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:

$$y_t = c + \phi_1 y_{t-1} + \epsilon_t \tag{2.1}$$

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:

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} ... + \phi_p y_{t-p} + \epsilon_t \tag{2.2}$$

hay

$$y_t = c + \sum_{i=1}^{p} \phi_i y_{t-i} \tag{2.3}$$

Trong đó Φ_i là hệ số tương ứng của mỗi giá trị y_{t-i}.

Moving average

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:

$$E(\epsilon_t) = 0 \tag{2.4}$$

$$\sigma(\epsilon_t) = \alpha \tag{2.5}$$

$$\rho(\epsilon_t, \epsilon_{t-s)} = 0, \forall t \geq s \tag{2.6}$$

Vế (2.4) 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 (2.5) là phương sai của chuỗi không đổi. Do kì vọng và phương sai không đỏi nên chúng ta gọi 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:

$$MA(q) = \mu + \sum_{i=1}^{q} \theta_i \epsilon_{t-i} \tag{2.7}$$

Intergrated

Là quá trình đồng tích hợp hoặc lấy sai phân. 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:

$$\Delta x_t = \phi_1 \Delta x_{t-1} + \phi_2 \Delta x_{t-2} + ...\phi_p \Delta x_{t-p} + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + ... + \theta_q \epsilon_{t-q} \quad (2.8)$$

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

Cách xác định hệ số d

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 không thể quyết định giữa hai thứ tự sai lệch, thì ta chọn thứ tự cho độ lệch chuẩn nhỏ nhất trong chuỗi sai lệch.

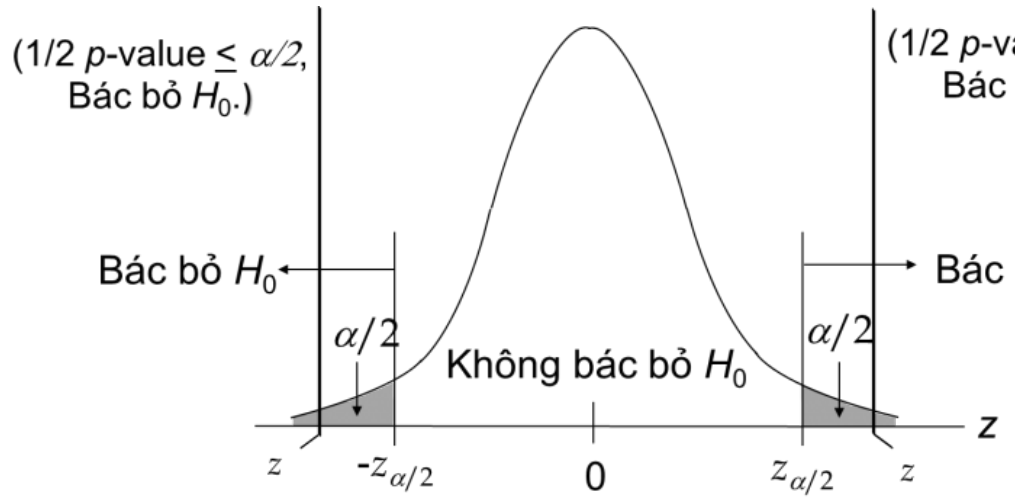
Cách xác định hệ số d

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.

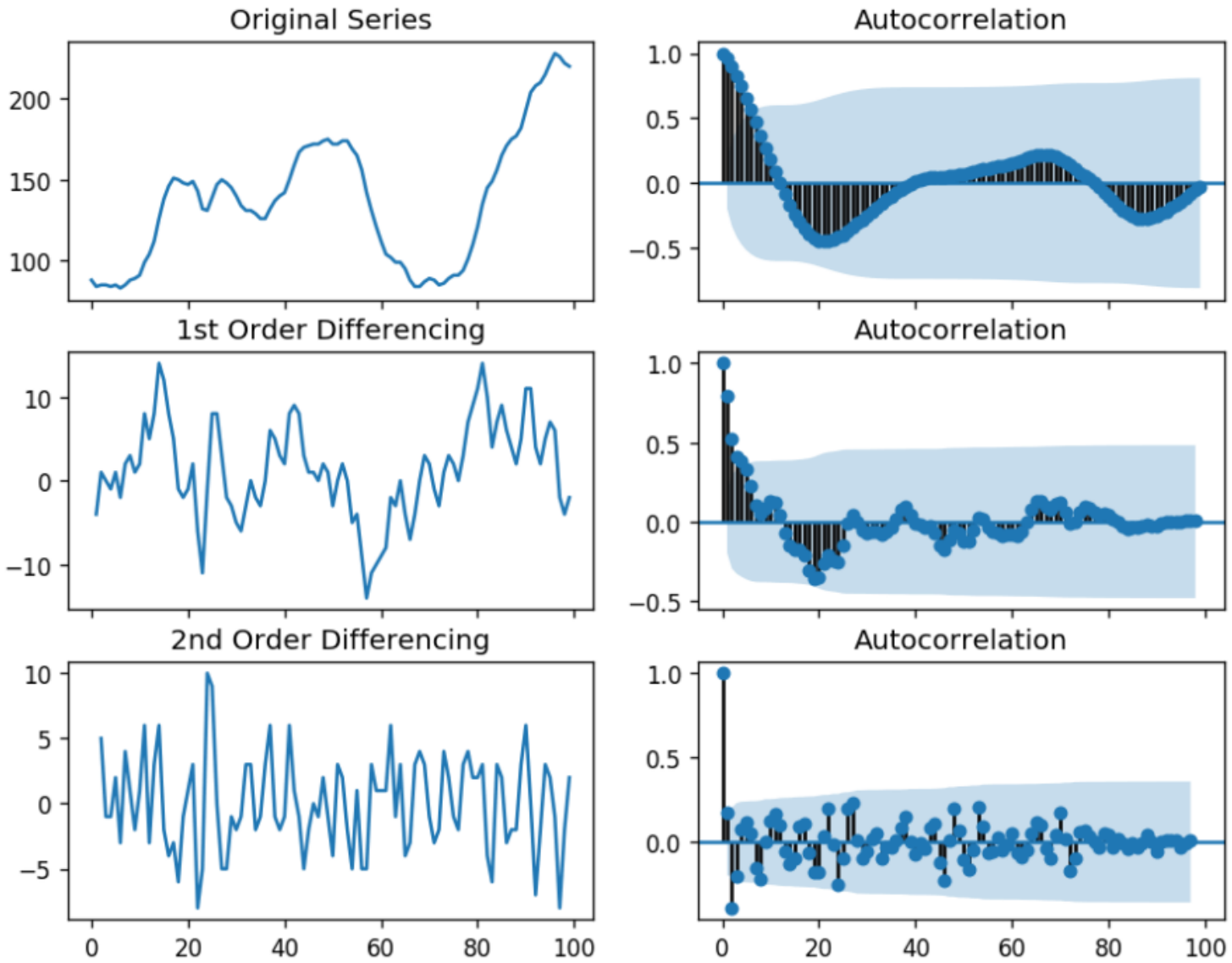
Kiểm định tham số p



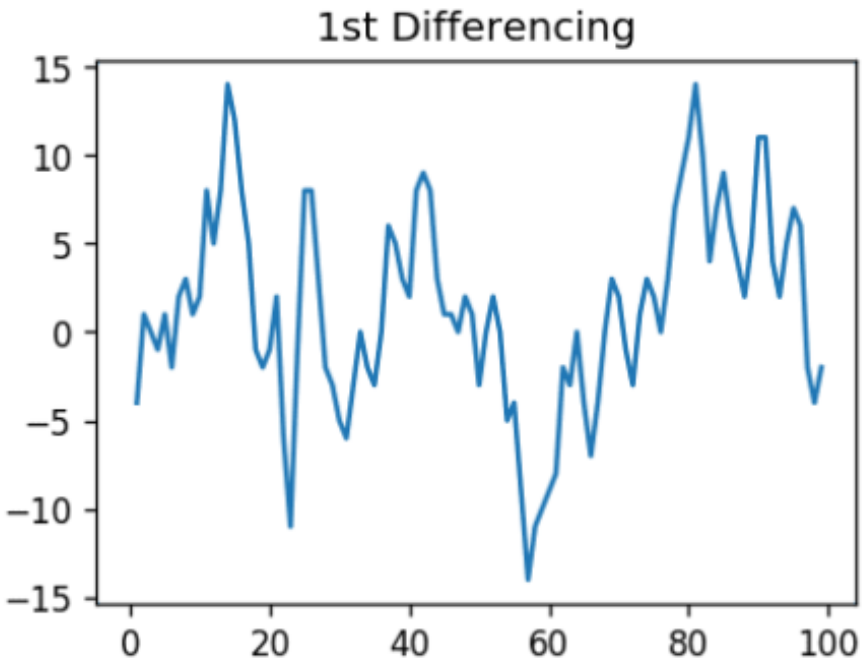
Giá trị kiểm định

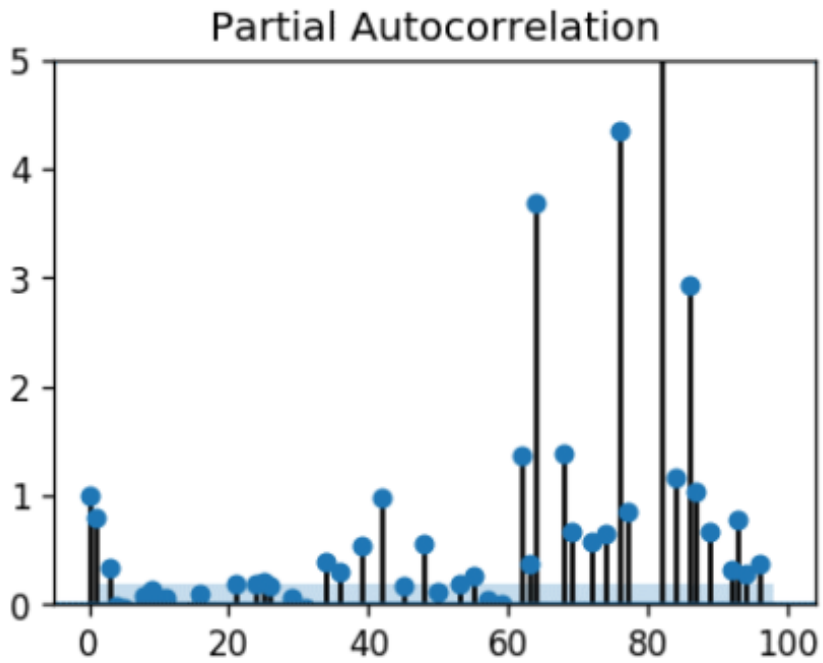
$$z = \frac{\bar{x} - \mu_0}{\sigma/\sqrt{n}}$$

Cách xác định hệ số d

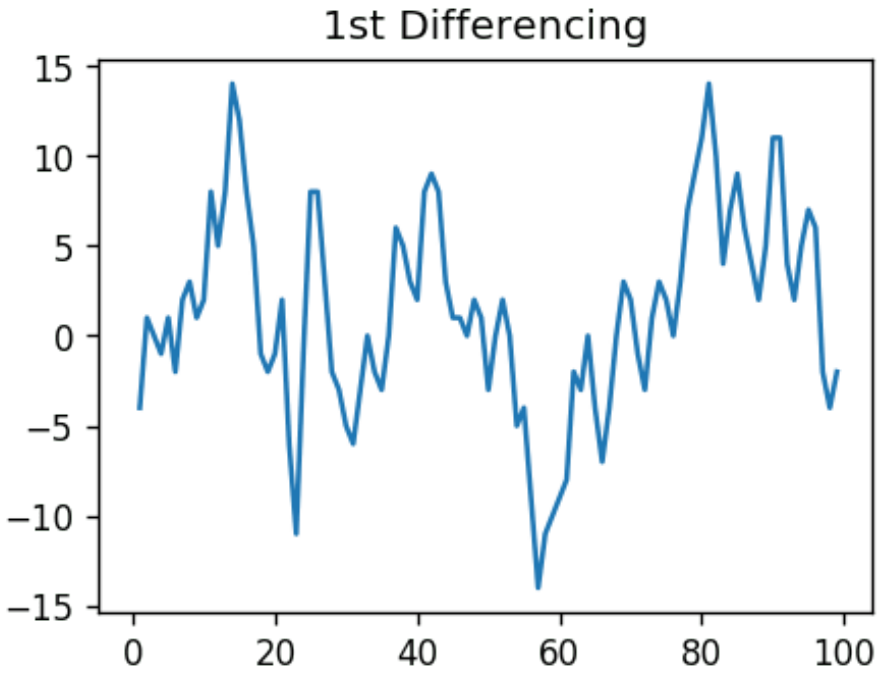


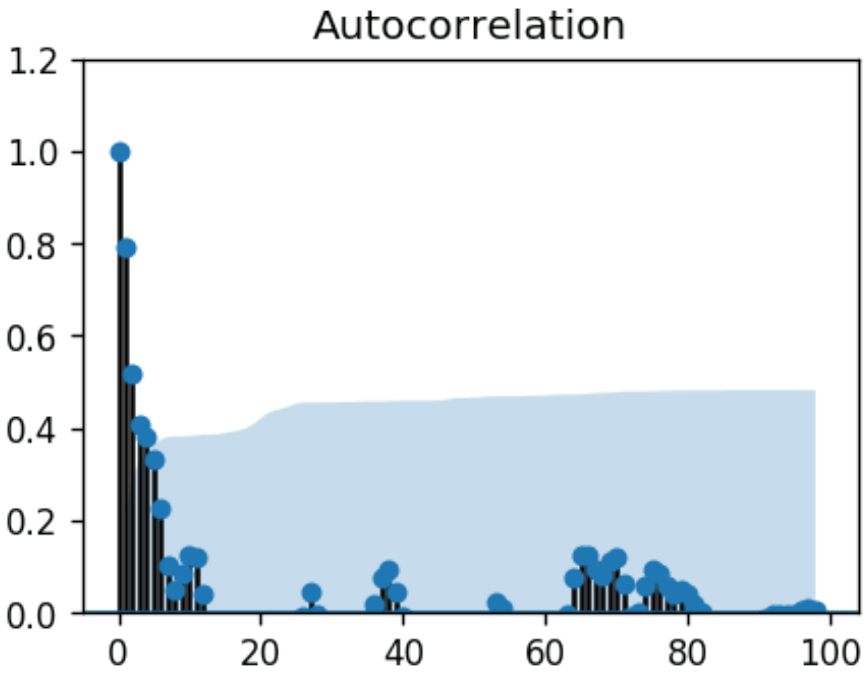
Cách xác định hệ số p của AR





Cách xác định hệ số q của MA





Giới thiệu mô hình SARIMA

- 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$(p, d, q)\,(P, D, Q)$

Giới thiệu mô hình SARIMA

Các giá trị trong mô hình SARIMA đại diện cho

- (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

Giới thiệu mô hình SARIMA

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:

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

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.

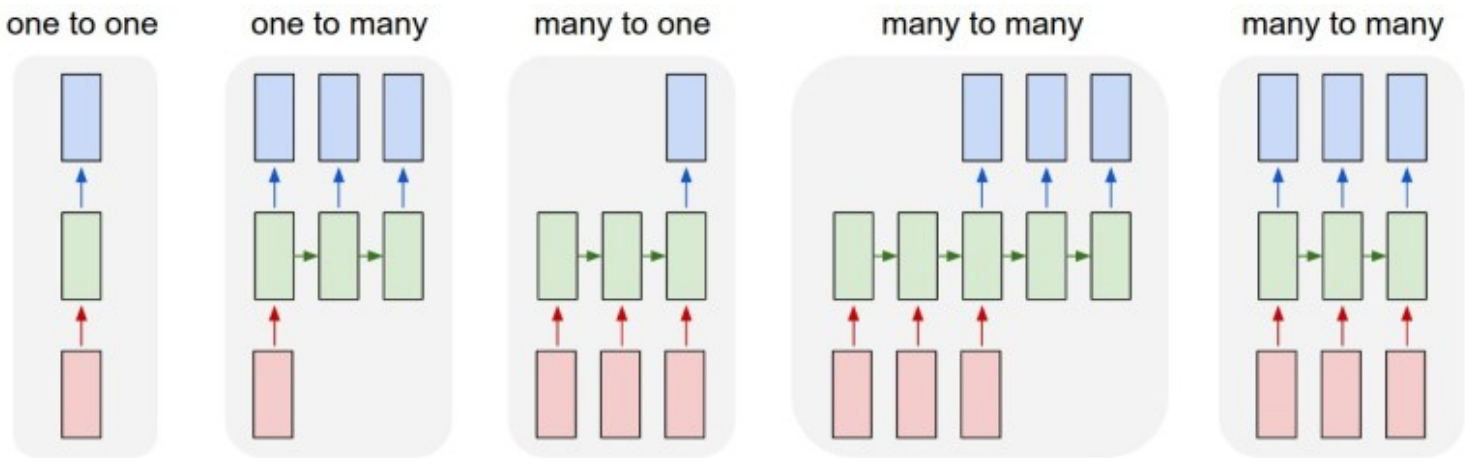
LSTM&ARIMA trong dự báo ô nhiễm môi trường

Tổng quan về RNN

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

- One to one: dạng toán 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.
- 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 và có nhiều output, ví dụ dạng toán dịch từ, "I love you " thành "Anh yêu em".

Các dạng bài toán RNN



Ứng dụng của bài toán RNN

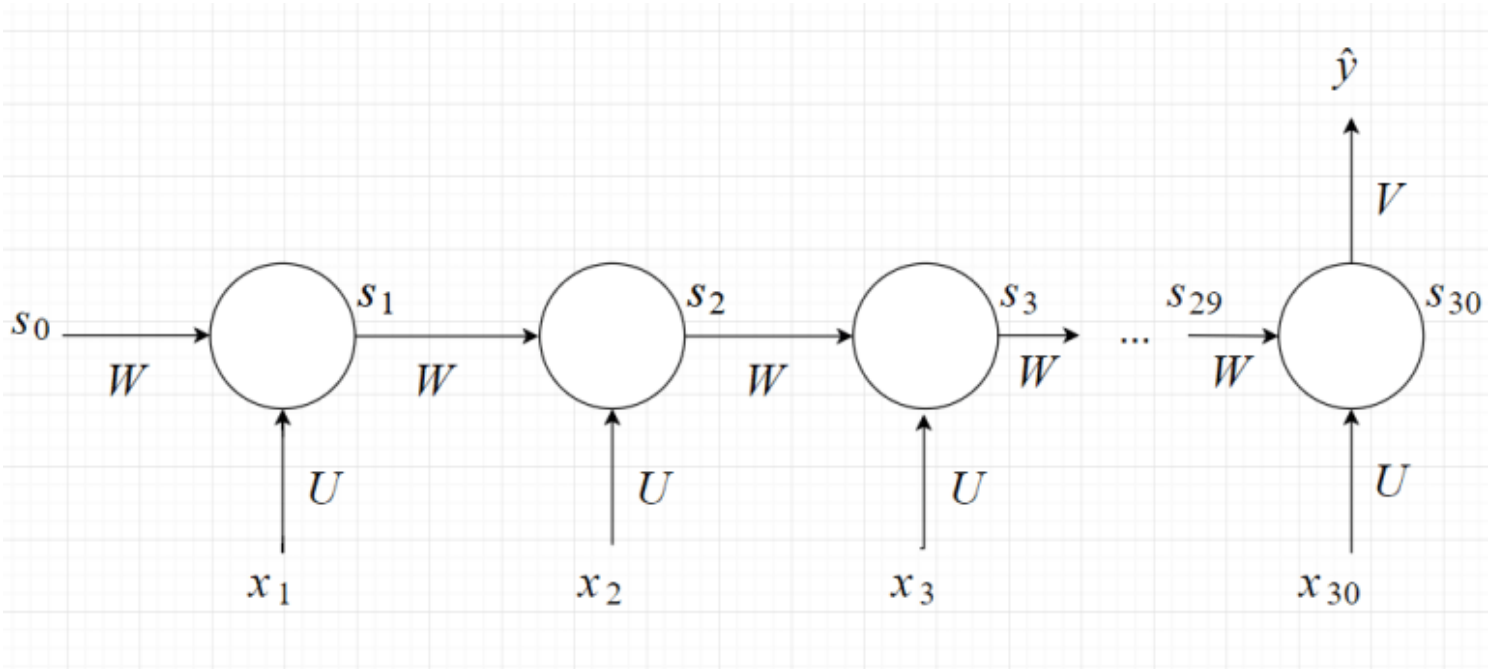
- Dự báo, phân loại các bài toán time series
- Chuyển giọng nói thành văn bản
- Phân loại cảm xúc trong văn bản
- Nhận diện hành động trong video
- Dịch tự động các ngôn ngữ

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.

- Có 30 input, 1 output, các input là $x_1, x_2, x_3, \ldots, x_{30}$
- 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 đẩy đủ thông tin của các input.

Mô hình bài toán RNN



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

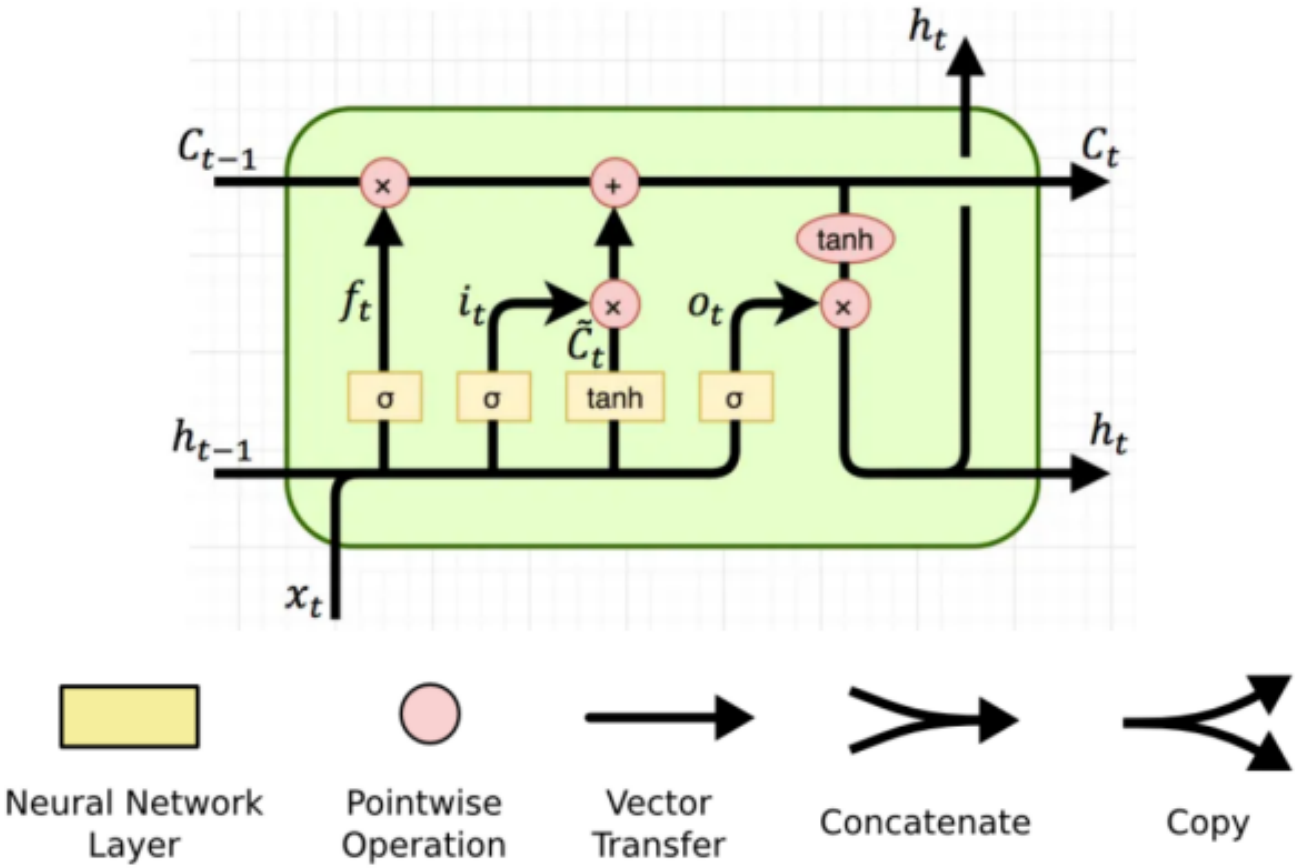
- $s_0 = 0, s_t = f(U * x_t + W * s_{t-1})$
- $o_t = g(V * s_t)$
- V là ma trận có kích thước d*m

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.

- 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.

Mô hình LSTM


h_t
C_{t-1}
C_t
tanh
f_t
i_t
o_t
$\tilde{C}_t$
σ
σ
tanh
σ
h_{t-1}
h_t
x_t
Neural Network Layer
Pointwise Operation
Vector Transfer
Concatenate
Copy

Input và Output

Input

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

Output

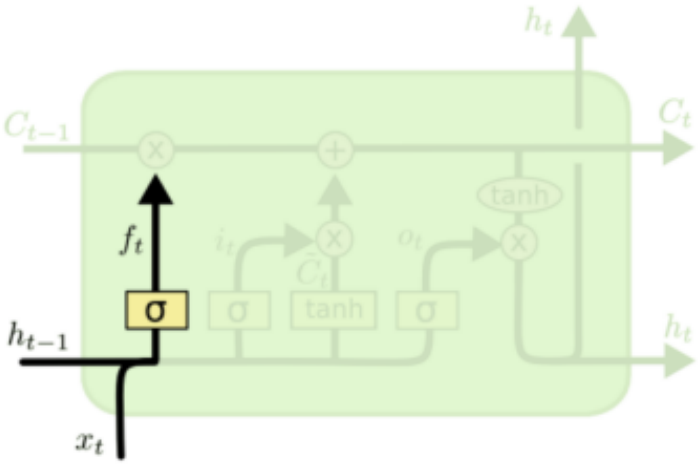
- Cell state C_t, hidden state h_t.
- Output của LSTM thêm thành phần Ct so với RNN.

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 $o_t = \sigma(U_o * x_t + W_o * h_{o-1} + b_o)$

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].



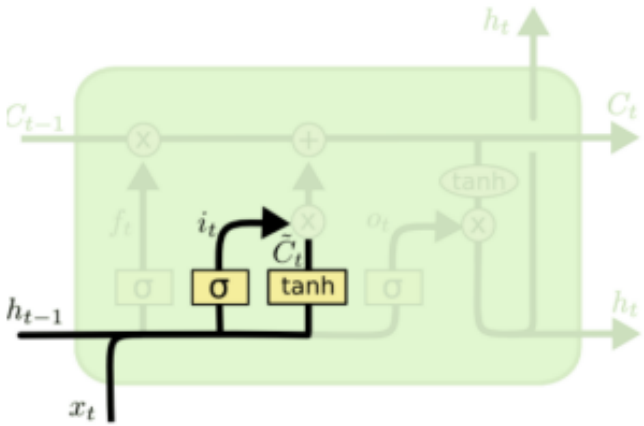
$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Bên trong mạng LSTM

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

1. Sử dụng một tầng sigmoid để quyết định giá trị nào sẽ cập nhập.
2. Tầng tanh tạo ra một vector cho giá trị mới $\tilde{C}_t$ nhằm thêm vào cho state.

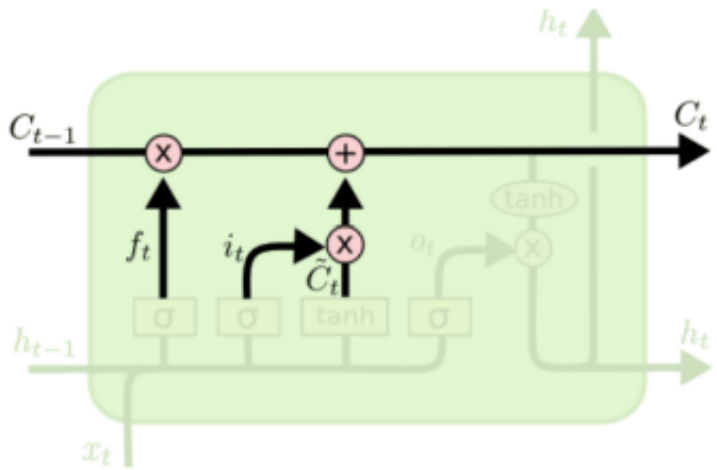
Sau đó, ta sẽ kết hợp 2 giá trị đó lại để tạo ra một cập nhập cho state.



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

Bên trong mạng LSTM

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.

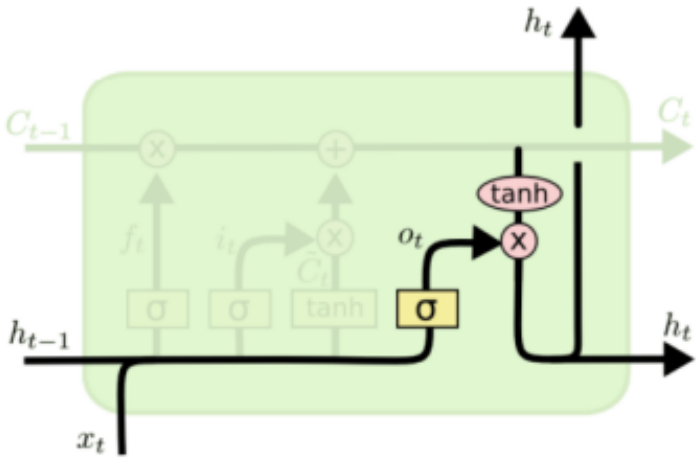


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Bên trong mạng LSTM

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.

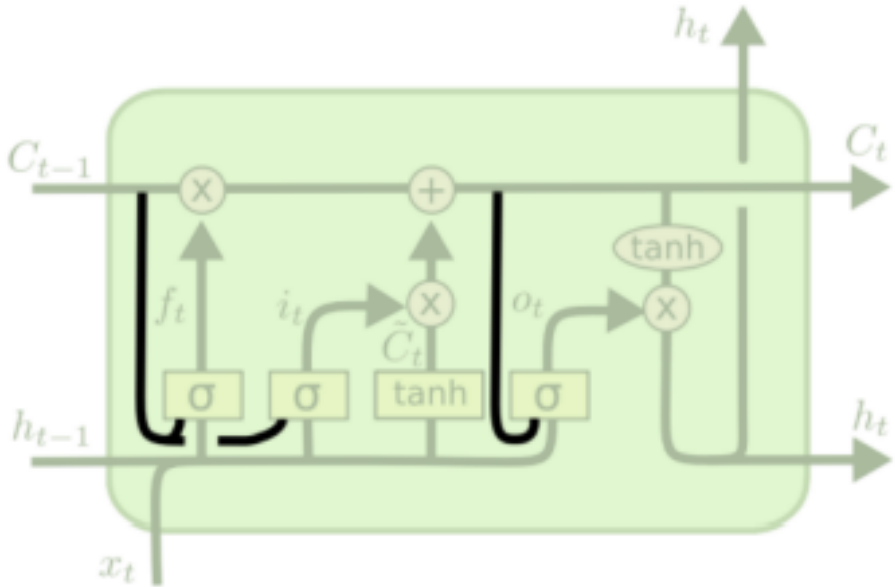


$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

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

Một số dạng biến thể của LSTM

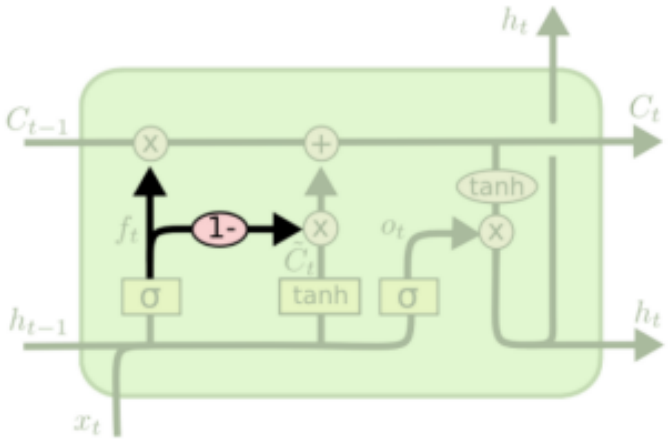
Dạng LSTM thêm một số đường liên kết, các tầng cổng nhận input là các giá trị của cell state.



Một số dạng biến thể của 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.



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

LSTM&ARIMA trong dự báo ô nhiễm môi trường

Mô tả bộ dữ liệu

- 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

Các metrics đánh giá

Root Mean Squared Error (RMSE):

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=0}^{n}(y_i - \hat{y}_i)^2} \tag{4.1}$$

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.

Các metrics đánh giá

Mean Absolute Error (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:

$$\text{MAE} = \frac{1}{n}\sum_{i=0}^{n}|y_i - \hat{y}_i| \tag{4.2}$$

Các metrics đánh giá

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:

$$\text{MAPE} = \frac{1}{n}\sum_{i=0}^{n}\frac{|y_i - \hat{y}_i|}{y_i} \tag{4.3}$$

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

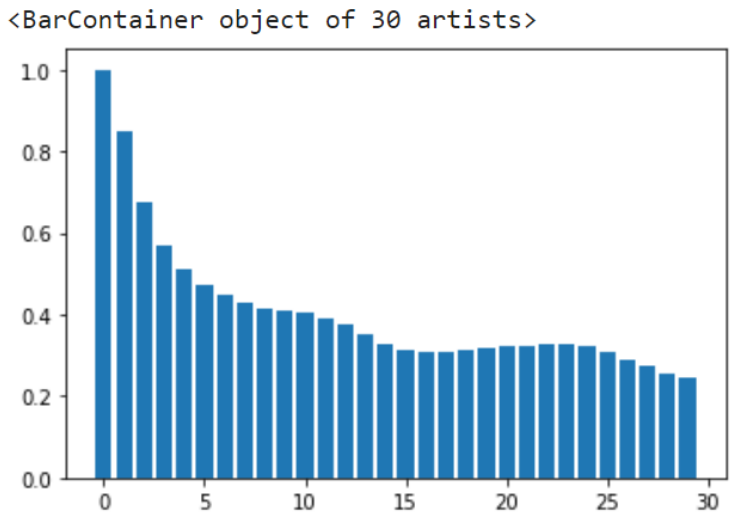
Mô hình SARIMA

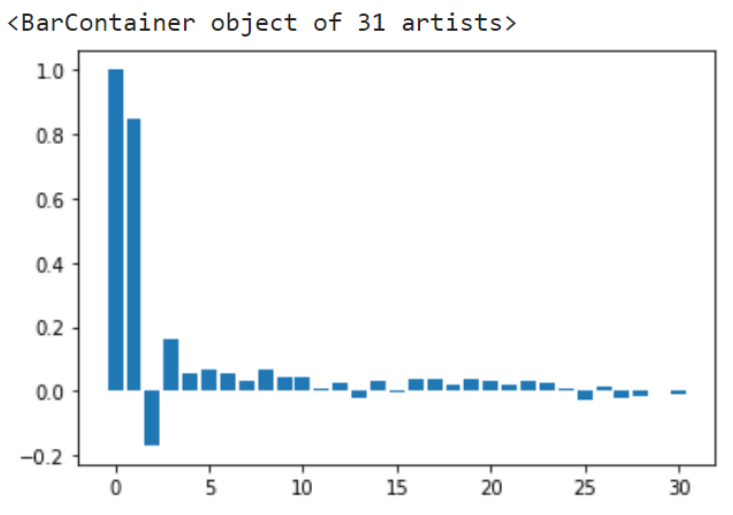
Đầu tiên ta xét xem chuỗi đã là chuỗi dừng chưa.
Thực hiện lệnh để tìm chỉ số d:

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

Mô hình SARIMA

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





Mô hình SARIMA

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)
```

Mô hình LSTM

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
```

Mô hình LSTM

```
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
```

Mô hình LSTM

```
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
```

Kết quả

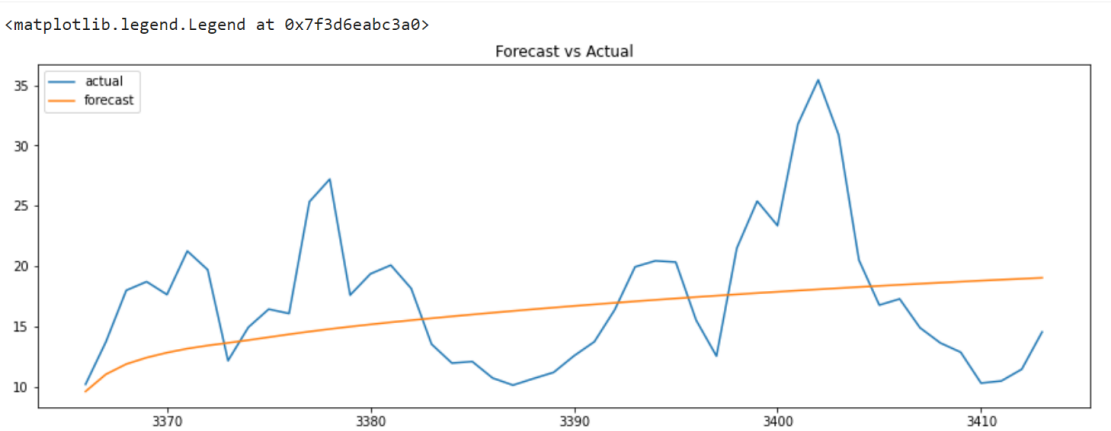


Figure: Mô hình ARIMA

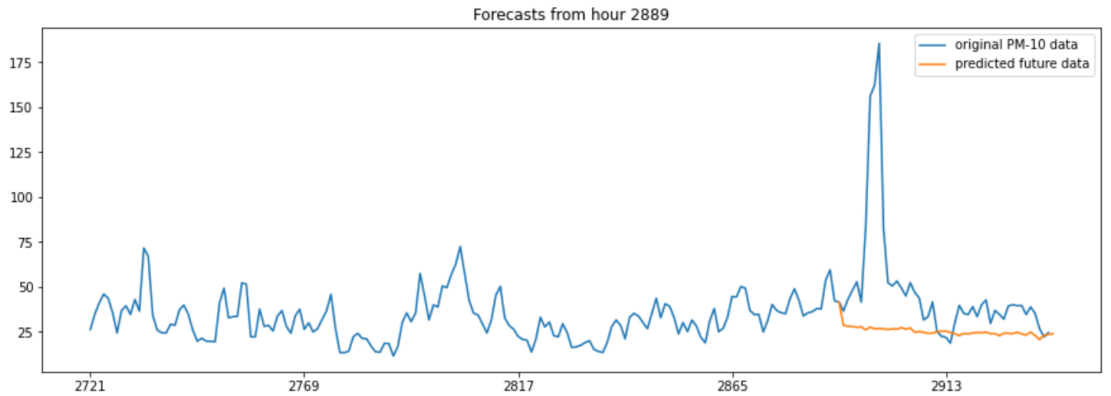


Figure: Mô hình LSTM

Kết quả

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

	MSE	MAE
ARIMA	326.97	11.66
LSTM	140.45	7.99

Nhận xét

Thuân 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.

LSTM&ARIMA trong dự báo ô nhiễm môi trường

Những điều đã làm được

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 thời tiết.

Các hướng phát triển tiếp của đồ án

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 [3] 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) [1], Depth Gated RNNs [4] hay Clockwork RNNs [2]

Thanks for listening!

[1] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. CoRR, abs/1406.1078, 2014.

[2] Jan Koutník, Klaus Greff, Faustino J. Gomez, and Jürgen Schmidhuber. A clockwork RNN. CoRR, abs/1402.3511, 2014.

[3] Chenghao Liu, Steven C.H. Hoi, Peilin Zhao, and Jianling Sun. Online arima algorithms for time series prediction. Proceedings of the AAAI Conference on Artificial Intelligence, 30(1), Feb. 2016.

[4] Kaisheng Yao, Trevor Cohn, Katerina Vylomova, Kevin Duh, and Chris Dyer. Depth-gated recurrent neural networks. 08 2015.