|  |  |
| --- | --- |
| **Ex no : 6** | **Implement LSTM to perform time series prediction** |
| **Date :** |

**Aim**

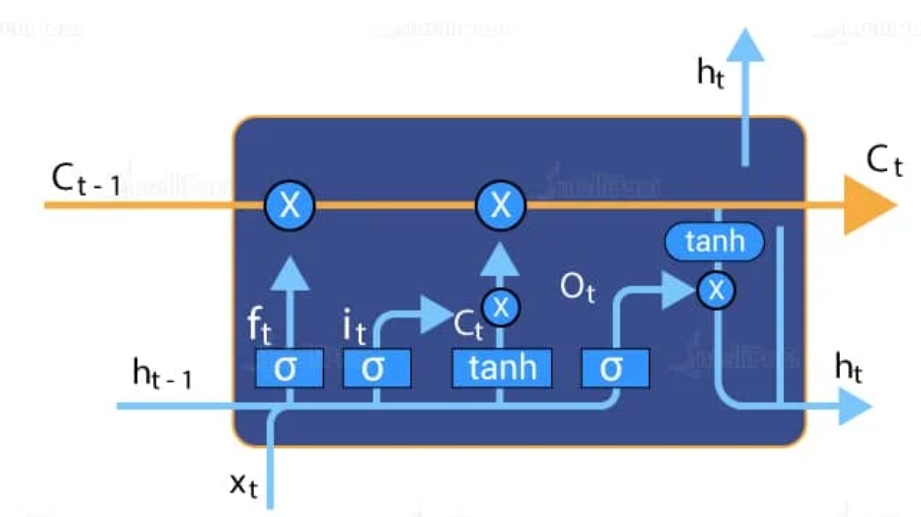
To implement LSTM to perform time series prediction.

**Basic Theory of LSTM**

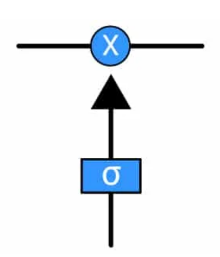
LSTM stands for long short-term memory networks is a variety of recurrent neural networks (RNNs) that are capable of learning long-term dependencies, especially in sequence prediction problems.

LSTM has feedback connections, i.e., it is capable of processing the entire sequence of data, apart from single data points such as images. This finds application in speech recognition, machine translation, etc. LSTM is a special kind of RNN, which shows outstanding performance on a large variety of problems.

The central role of an LSTM model is held by a memory cell known as a ‘cell state’ that maintains its state over time. The cell state is the horizontal line that runs through the top of the below diagram. It can be visualized as a conveyor belt through which information just flows, unchanged.



Information can be added to or removed from the cell state in LSTM and is regulated by gates. These gates optionally let the information flow in and out of the cell. It contains a point wise multiplication operation and a sigmoid neural net layer that assist the mechanism.



The sigmoid layer gives out numbers between zero and one, where zero means ‘nothing should be let through’, and one means ‘everything should be let through’.

**Code**

***# univariate lstm example***

**import** numpy **as** np

**from** tensorflow.keras.models **import** Sequential

**from** tensorflow.keras.layers **import** LSTM

**from** tensorflow.keras.layers **import** Dense

**from** tensorflow.keras.layers **import** Flatten

***# preparing independent and dependent features***

**def** prepare\_data(timeseries\_data, n\_features):

X, y **=**[],[]

**for** i **in** range(len(timeseries\_data)):

*# find the end of this pattern*

end\_ix **=** i **+** n\_features

*# check if we are beyond the sequence*

**if** end\_ix **>** len(timeseries\_data)**-**1:

**break**

*# gather input and output parts of the pattern*

seq\_x, seq\_y **=** timeseries\_data[i:end\_ix], timeseries\_data[end\_ix]

X**.**append(seq\_x)

y**.**append(seq\_y)

**return** np**.**array(X), np**.**array(y)

***# define input sequence***

timeseries\_data **=** [110, 125, 133, 146, 158, 172, 187, 196, 210]

*# choose a number of time steps*

n\_steps **=** 3

***# split into samples***

X, y **=** prepare\_data(timeseries\_data, n\_steps)

print(X),print(y)

X**.**shape

*#* ***reshape from [samples, timesteps] into [samples, timesteps, features]***

n\_features **=** 1

X **=** X**.**reshape((X**.**shape[0], X**.**shape[1], n\_features))

*#* ***define model***

model **=** Sequential()

model**.**add(LSTM(50, activation**=**'relu', return\_sequences**=True**, input\_shape**=**(n\_steps, n\_features)))

model**.**add(LSTM(50, activation**=**'relu'))

model**.**add(Dense(1))

model**.**compile(optimizer**=**'adam', loss**=**'mse')

***# fit model***

model**.**fit(X, y, epochs**=**300, verbose**=**1)

***# demonstrate prediction for next 10 days***

x\_input **=** array([187, 196, 210])

temp\_input**=**list(x\_input)

lst\_output**=**[]

i**=**0

**while**(i**<**10):

**if**(len(temp\_input)**>**3):

x\_input**=**array(temp\_input[1:])

print("{} day input {}"**.**format(i,x\_input))

*#print(x\_input)*

x\_input **=** x\_input**.**reshape((1, n\_steps, n\_features))

*#print(x\_input)*

yhat **=** model**.**predict(x\_input, verbose**=**0)

print("{} day output {}"**.**format(i,yhat))

temp\_input**.**append(yhat[0][0])

temp\_input**=**temp\_input[1:]

*#print(temp\_input)*

lst\_output**.**append(yhat[0][0])

i**=**i**+**1

**else**:

x\_input **=** x\_input**.**reshape((1, n\_steps, n\_features))

yhat **=** model**.**predict(x\_input, verbose**=**0)

print(yhat[0])

temp\_input**.**append(yhat[0][0])

lst\_output**.**append(yhat[0][0])

i**=**i**+**1

print(lst\_output)

timeseries\_data

len(timeseries\_data)

lst\_output

lst

***# Visualizing The Output***

**import** matplotlib.pyplot **as** plt

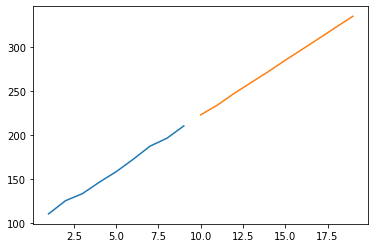
day\_new**=**np**.**arange(1,10)

day\_pred**=**np**.**arange(10,20)

plt**.**plot(day\_new,timeseries\_data)

plt**.**plot(day\_pred,lst\_output)

**OUTPUT**



**RESULT**

Thus implementation of LSTM to perform time series prediction has been carried out successfully.