Time Series Forecasting - 201

PURDUE UNIVERSITY®

PREVIOUS SESSION

- Simple Linear Regression
- Multiple Linear Regression
- AR models – ARIMA, SARIMA
- Smoothing Methods

PURDUE UNIVERSITY

DATASET

- Number of units sold of different category of items in different Walmart stores situated in different cities, over the past 6 years

	id	item_id	dept_id	cat_id	store_id	state_id	d_1	d_2	d_3	d_4	...	d_1904	d_1905	d_1906	d_1907	d_
0	HOBBIES_1_001_CA_1_validation	HOBBIES_1_001	HOBBIES_1	HOBBIES	CA_1	CA	0	0	0	0	...	1	3	0	1	
1	HOBBIES_1_002_CA_1_validation	HOBBIES_1_002	HOBBIES_1	HOBBIES	CA_1	CA	0	0	0	0	...	0	0	0	0	
2	HOBBIES_1_003_CA_1_validation	HOBBIES_1_003	HOBBIES_1	HOBBIES	CA_1	CA	0	0	0	0	...	2	1	2	1	
3	HOBBIES_1_004_CA_1_validation	HOBBIES_1_004	HOBBIES_1	HOBBIES	CA_1	CA	0	0	0	0	...	1	0	5	4	
4	HOBBIES_1_005_CA_1_validation	HOBBIES_1_005	HOBBIES_1	HOBBIES	CA_1	CA	0	0	0	0	...	2	1	1	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
30485	FOODS_3_823_WI_3_validation	FOODS_3_823	FOODS_3	FOODS	WI_3	WI	0	0	2	2	...	2	0	0	0	
30486	FOODS_3_824_WI_3_validation	FOODS_3_824	FOODS_3	FOODS	WI_3	WI	0	0	0	0	...	0	0	0	0	
30487	FOODS_3_825_WI_3_validation	FOODS_3_825	FOODS_3	FOODS	WI_3	WI	0	6	0	2	...	2	1	0	2	
30488	FOODS_3_826_WI_3_validation	FOODS_3_826	FOODS_3	FOODS	WI_3	WI	0	0	0	0	...	0	0	1	0	
30489	FOODS_3_827_WI_3_validation	FOODS_3_827	FOODS_3	FOODS	WI_3	WI	0	0	0	0	...	0	0	0	0	

WHAT PROBLEM ARE WE GOING TO SOLVE?

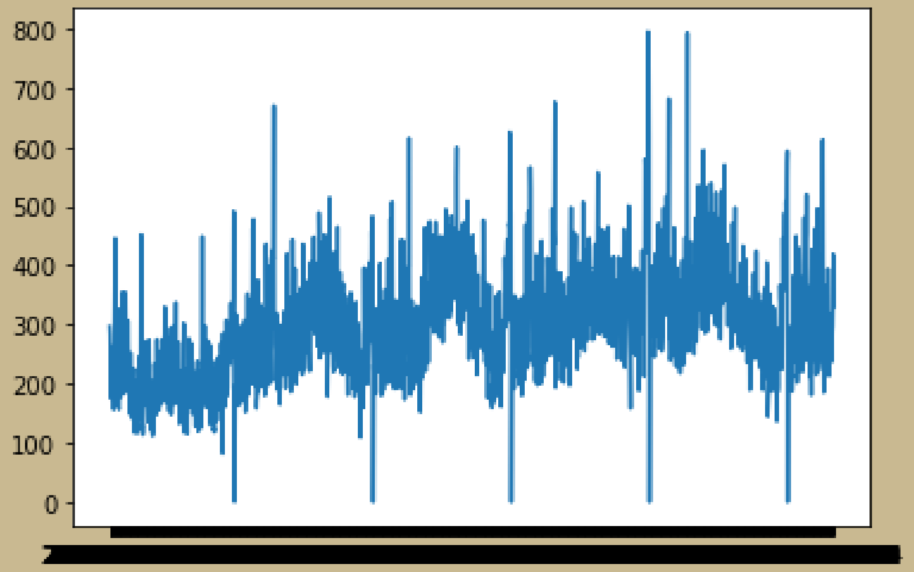
Out of 48,000 different Time Series, we choose the following aggregated Time Series to forecast

States x Categories x Departments x Items
48,000 Time Series

State =
CA

Category =
Foods

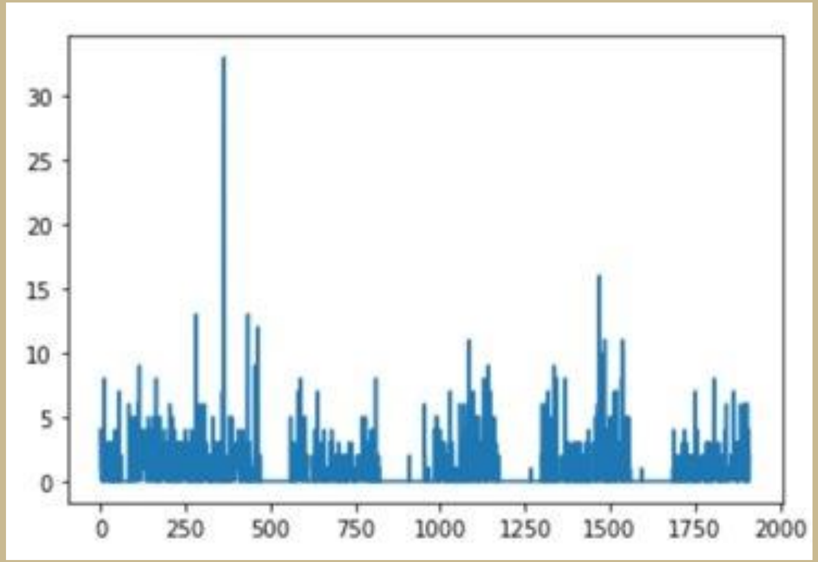
Department =
Foods_1



PURDUE
UNIVERSITY®

WHY DO WE MODEL THE FOOD-WISE TIME SERIES AND NOT ITEM WISE TIME SERIES?!

Item X Sales

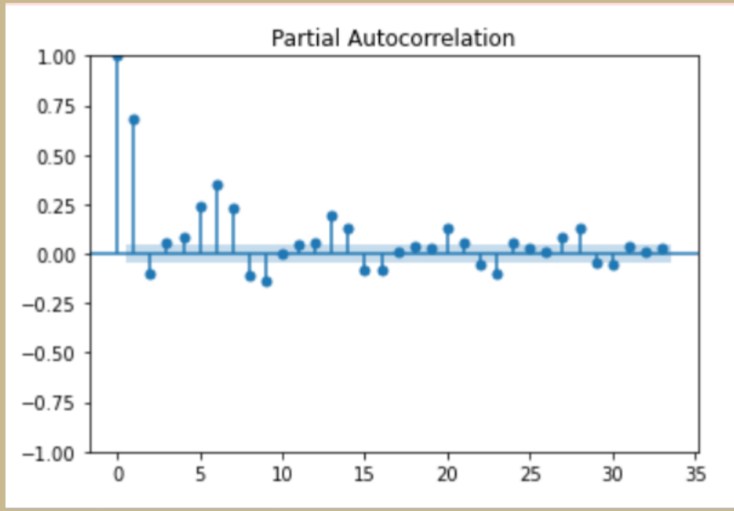


- Naked Eye Test: There is no discernable pattern that a model can capture
- Many inconsistent peaks and falls making it difficult to model

PURDUE UNIVERSITY®

AUTO ARIMA

- Auto ARIMA is simply automated ARIMA – the computer decides the (p,d,q) (P,D,Q) combination
- Helpful when we have several significant lags:




```
train = for_arima[:1800]
test = for_arima[1800:]
model = auto_arima(train, X=None, start_p=2, d=None, start_q=2, max_p=5, max_d=2,
                   max_q=5, start_P=1, D=None, start_Q=1, max_P=2, max_D=1, max_Q=2,
                   max_order=5, m=1, seasonal=True, stationary=False, information_criterion='aic',
                   alpha=0.05, test='kpss', seasonal_test='ocsb', stepwise=True, n_jobs=1,
                   start_params=None, trend=None, method='lbfgs', maxiter=50, offset_test_args=None,
                   seasonal_test_args=None, suppress_warnings=True, error_action='trace', trace=False,
                   random=False, random_state=None, n_fits=10, return_valid_fits=False, out_of_sample_size=0,
                   scoring='mse', scoring_args=None, with_intercept='auto', sarimax_kwargs=None)
```

PURDUE UNIVERSITY

AUTO ARIMA RESULTS

SARIMA Model Summary

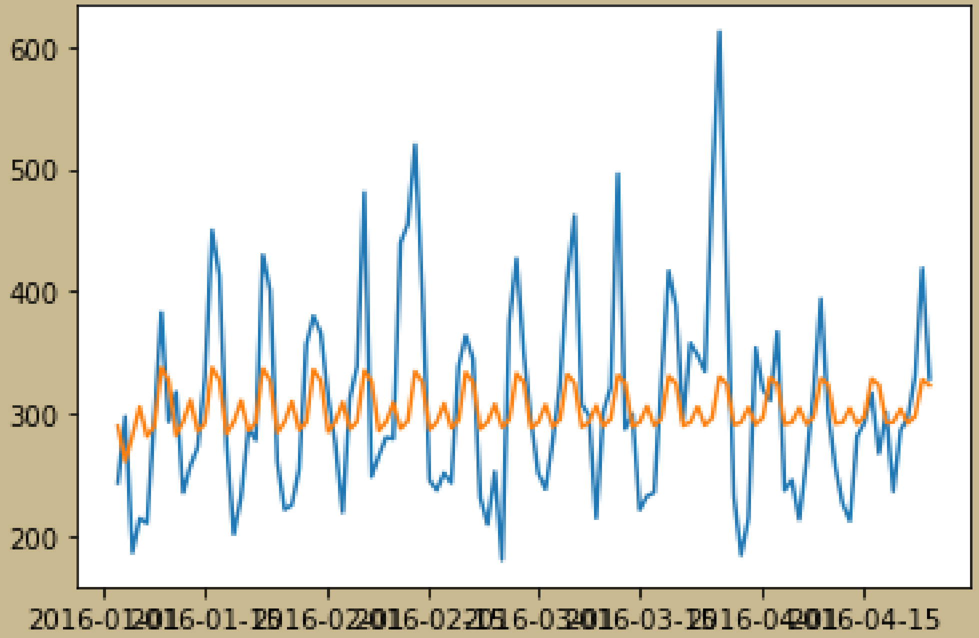
SARIMAX Results

Dep. Variable:	y	**No. Observations:**	1800
Model:	SARIMAX(5, 1, 5)	**Log Likelihood**	-9801.278
Date:	Sat, 04 Feb 2023	**AIC**	19624.555
Time:	22:09:37	**BIC**	19685.000
Sample:	01-29-2011	**HQIC**	19646.869
	- 01-02-2016		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	1.3554	0.027	49.492	0.000	1.302	1.409
ar.L2	-1.8813	0.024	-79.492	0.000	-1.928	-1.835
ar.L3	1.5842	0.039	40.654	0.000	1.508	1.661
ar.L4	-1.4232	0.022	-65.758	0.000	-1.466	-1.381
ar.L5	0.5431	0.025	21.629	0.000	0.494	0.592
ma.L1	-1.7744	0.021	-85.334	0.000	-1.815	-1.734
ma.L2	2.1876	0.034	63.552	0.000	2.120	2.255
ma.L3	-2.1313	0.043	-49.599	0.000	-2.216	-2.047
ma.L4	1.6664	0.035	47.125	0.000	1.597	1.736
ma.L5	-0.8912	0.019	-47.098	0.000	-0.928	-0.854
sigma2	3768.1895	74.640	50.485	0.000	3621.897	3914.482

Ljung-Box (L1) (Q):	7.60	**Jarque-Bera (JB):**	10390.27
Prob(Q):	0.01	**Prob(JB):**	0.00
Heteroskedasticity (H):	2.14	**Skew:**	-0.76
Prob(H) (two-sided):	0.00	**Kurtosis:**	14.68

Test Set Actual vs Prediction



Test MAPE = 30.37%

- This model uses 5 lags to forecast values
- Mediocre performance, but it captures some seasonality
- Maybe incorporating recent as well as not-so-recent past observations in the predictions, would yield better results. **But using ARIMA for that wouldn't make so much sense!**

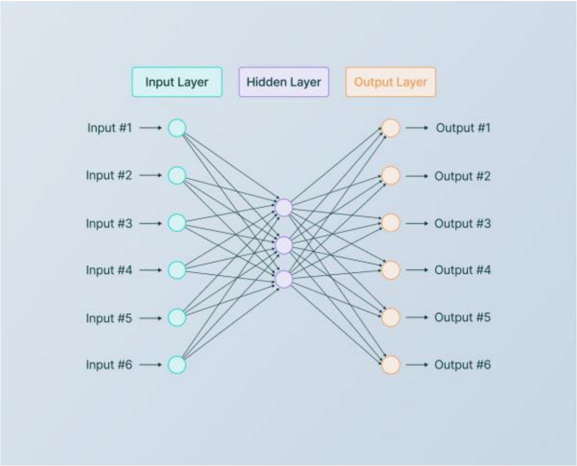
Deep Learning models for Time Series

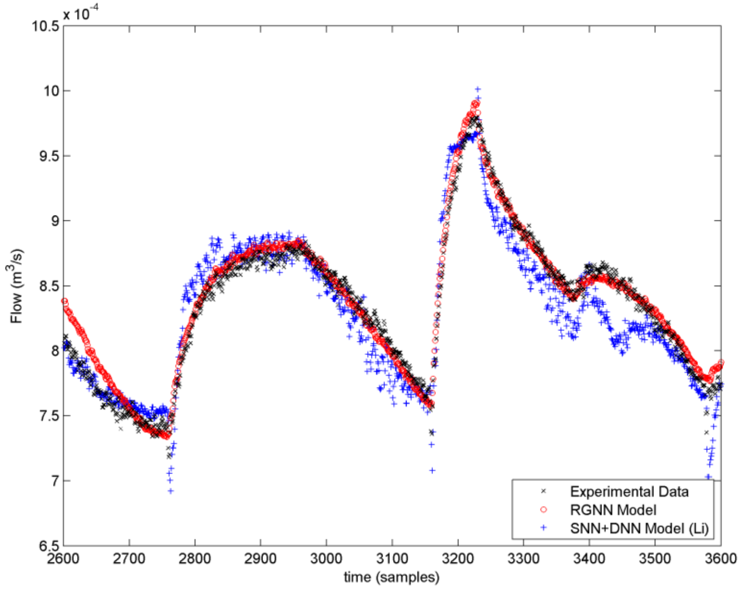
- Artificial (Deep) Neural Network
- Recurrent Neural Network
- Long-Short Term Memory
- Convolutional Neural Network

- In this tutorial, we focus on ANN and LSTM. LSTM is an improved version of RNN.

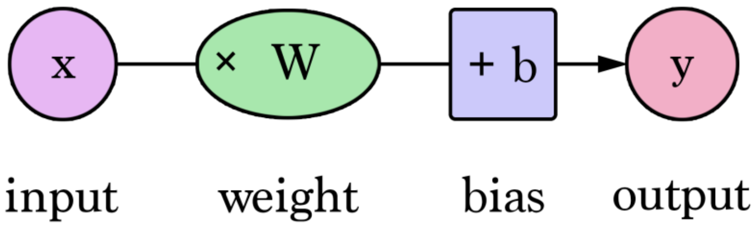
Artificial (deep) Neural Network

- A neural network (ANN or DNN) can be thought of as a complicated function fitter!

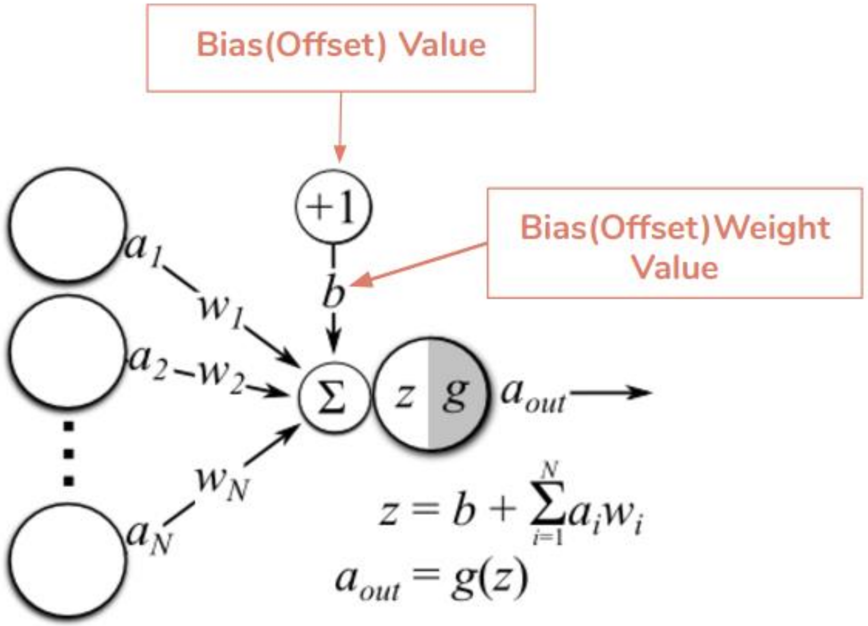




A Simple illustration



- Multiple linear functions are combined, passed into a non-linear activation function (a function used to fit non-linear trends), to produce the output
- Major hyperparameters of an ANN – Number of hidden layers, number of neurons in each layer, activation function



PURDUE UNIVERSITY

HOW DO WE USE AN ANN HERE?

- Feed in the lags of y(t) - y(t-1), y(t-2)…..y(t-n)
- Decision to make - What number of n yields the best results?
- Use exogenous features such as day of the week, month of the year etc.
- Approach is similar to ARIMA/SARIMA, but introducing a non-linear aspect (imagine, a non-linear form of ARIMA)

```python
def build_model():
    model = Sequential()
    model.add(InputLayer(input_shape=(71, )))
    model.add(Dense(32, activation=mish))
    model.add(Dense(64, activation=mish))
    model.add(Dense(128, activation=mish))
    model.add(Dense(256, activation=mish))
    model.add(Dense(512, activation=mish))
    model.add(Dense(1024, activation=mish))
    model.add(Dense(2048, activation=mish))
    model.add(Dense(1, activation='linear'))
    return model
```

```
In [42]: X_train = for_cnn_foods_1_fm.iloc[:1800,1:]
         y_train = for_cnn_foods_1_fm.iloc[:1800,0]
         X_test = for_cnn_foods_1_fm.iloc[1800:,1:]
         y_test = for_cnn_foods_1_fm.iloc[1800:,0]

In [43]: X_train
```

Out[43]:

_Tuesday	weekday_Wednesday	event_1_ChristmasNational	event_1_Cinco De MayoCultural	event_1_ColumbusDayNational	...	20	21	22	23	24	25	26	27	28	29
0	0	0	0	0	...	226	301	303	228	160	156	193	238	327	264
1	0	0	0	0	...	301	303	228	160	156	193	238	327	264	177
0	1	0	0	0	...	303	228	160	156	193	238	327	264	177	276
0	0	0	0	0	...	228	160	156	193	238	327	264	177	276	203
0	0	0	0	0	...	160	156	193	238	327	264	177	276	203	216

PURDUE
UNIVERSITY®

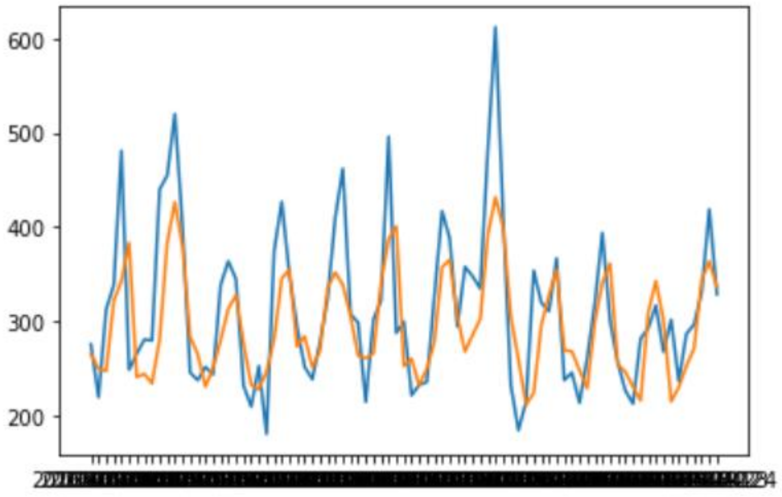
PERFORMANCE OF DIFFERENT ANNS ON THE TEST SET

```
mape(y_test,y_test_pred)
```

```
0.13935790900361686
```

```
plt.plot(y_test)
plt.plot(y_test_pred)
```

```
[<matplotlib.lines.Line2D at 0x2b5f8cd63d90>]
```

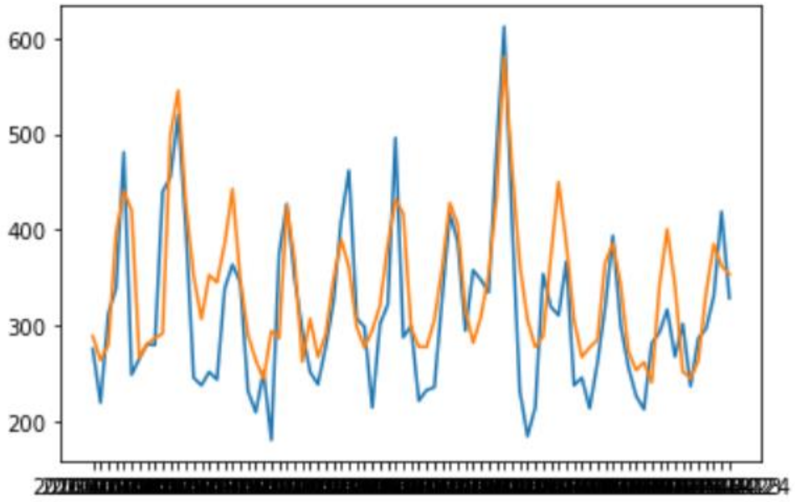



```
mape(y_test,y_test_pred)
```

```
0.17468749842659764
```

```
plt.plot(y_test)
plt.plot(y_test_pred)
```

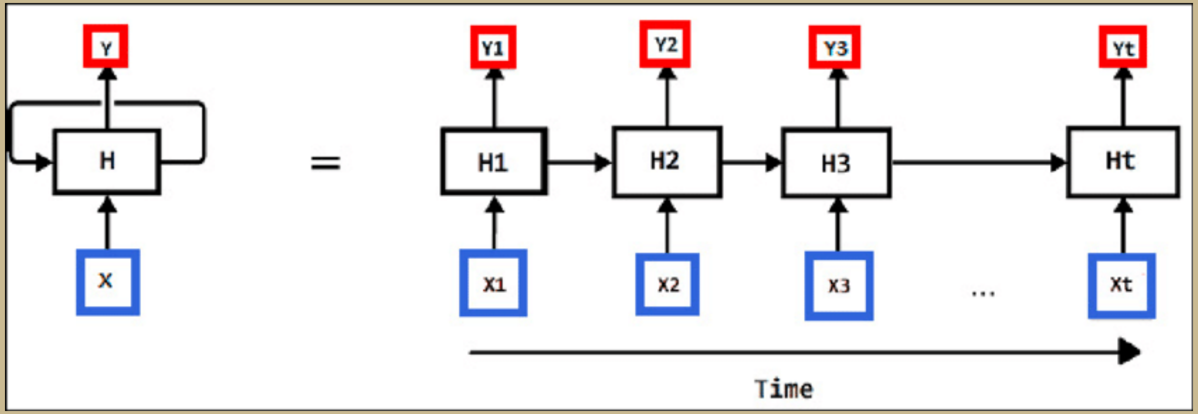
```
[<matplotlib.lines.Line2D at 0x2b5f8c6aeac0>]
```



PURDUE UNIVERSITY

RNN – RECURRENT NEURAL NETWORKS

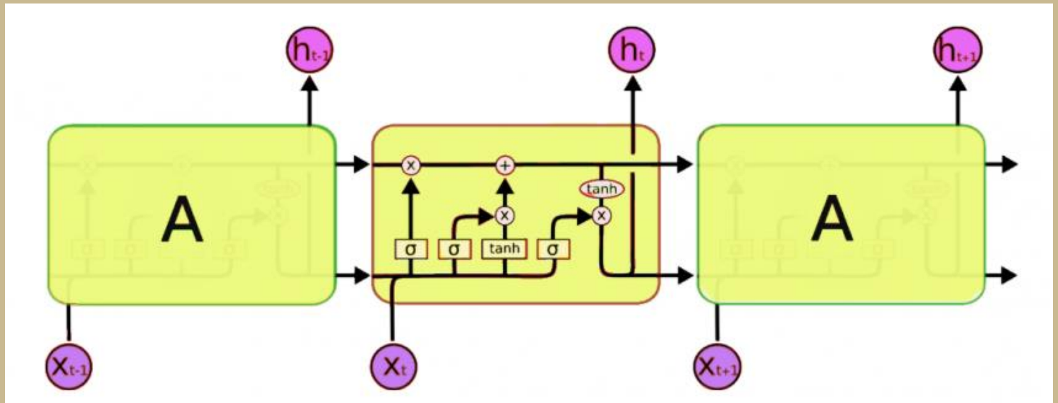
- RNN is an ANN whose architecture has slightly been modified, such that the network "remembers" what happened in the previous iteration. Apart from number of layers, and number of neurons, time-steps also come into play.



- Used in video classification tasks, speech recognition, time-series data etc.

LSTM – LONG SHORT TERM MEMORY

- A fancier version of RNNs, the difference being "how much" the network remembers at each time-step.
- Problem with RNNs – a) prone to vanishing/exploding gradients problems and b) does it remember too much/too less information?
- Solution – input, forget and output gates
- The forget gate determines which relevant information from the prior steps is needed. The input gate decides what relevant information can be added from the current step, and the output gates finalize the next hidden state



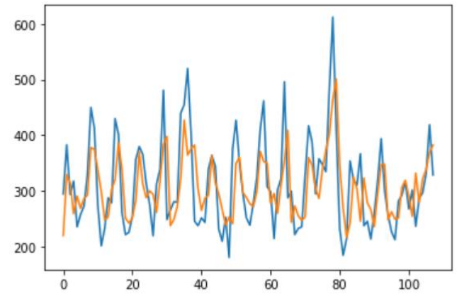
PURDUE UNIVERSITY

UNIVARIATE LSTM

- We feed in only y(t-1)…y(t-n), and no exogenous variables, and try different architectures

```
In [41]: model = tf.keras.Sequential()
         model.add(LSTM(50, activation='relu', input_shape=(5, 1)))
         model.add(Dense(1))
         model.compile(optimizer='adam', loss='mse')

         n_features = 1
         X_train = X_train.reshape((X_train.shape[0], X_train.shape[1], n_features))

In [44]: import time

In [44]: X_train

Out[44]: array([[[297],
                 [284],
                 [214],
                 [175],
                 [182]],

                [[284],
                 [214],
                 [175],
                 [182],
                 [191]],

                [[214],
```

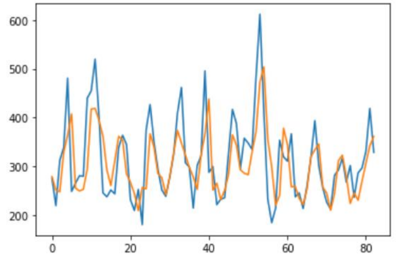
```
In [48]: mape(y_test, y_test_pred)

Out[48]: 0.16167876216782934

In [49]: plt.plot(y_test)
         plt.plot(y_test_pred)

Out[49]: [<matplotlib.lines.Line2D at 0x2b49f8aaab80>]
```



Architecture 1 – input variables are y(t-1) to y(t-5), with 50 LSTM units in the first layer

```
X_train, y_train = split_sequence(list(for_arima[:1800]['FOODS_1']), 30)
X_test, y_test = split_sequence(list(for_arima[1800:]['FOODS_1']), 30)
model = tf.keras.Sequential()
model.add(LSTM(50, activation='relu', input_shape=(30, 1)))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mse')
n_features = 1
X_train = X_train.reshape((X_train.shape[0], X_train.shape[1], n_features))

model.fit(X_train, y_train, epochs=500,callbacks=[PrintLoss()])
```

```
0.14915385259768094

plt.plot(y_test)
plt.plot(y_test_pred)

[<matplotlib.lines.Line2D at 0x2b4a60c8b250>]
```



Architecture 2 – input variables y(t-1) to y(t-30), with 50 LSTM units in the first layer

PURDUE UNIVERSITY®

MULTIVARIATE LSTM

- Include exogenous variables in the model
- Intuition - A regular ANN would consider that today is Christmas, and wouldn't care if yesterday or the day before were Christmas
- But wait – if yesterday were Christmas, and Christmas sales are still on, shouldn't the model care about Christmas until perhaps a week or two later?

```
trainX

array([[[297, 0, 0, ..., 0, 0, 1],
        [284, 0, 0, ..., 0, 0, 1],
        [214, 0, 1, ..., 0, 0, 1],
        ...,
        [238, 0, 0, ..., 0, 0, 1],
        [327, 0, 0, ..., 0, 0, 1],
        [264, 0, 0, ..., 0, 0, 1]],

       [[284, 0, 0, ..., 0, 0, 1],
        [214, 0, 1, ..., 0, 0, 1],
        [175, 1, 0, ..., 0, 0, 1],
        ...,
        [327, 0, 0, ..., 0, 0, 1],
        [264, 0, 0, ..., 0, 0, 1],
        [177, 0, 1, ..., 0, 0, 1]],

       [[214, 0, 1, ..., 0, 0, 1],
        [175, 1, 0, ..., 0, 0, 1],
        [182, 1, 0, ..., 0, 0, 1],
```

Input array structure

```python
input_shape=(trainX.shape[1],trainX.shape[2])
model = tf.keras.Sequential()
model.add(LSTM(64, activation='relu',input_shape=input_shape,
               return_sequences=True))
model.add(LSTM(32, activation='relu',return_sequences=False))
model.add(Dropout(0.2))
model.add(Dense(trainY.shape[1]))
model.compile(optimizer='adam',loss='mse')
filepath="cnn-{epoch:02d}-{val_loss:.2f}.hdf5"
checkpoint = ModelCheckpoint(filepath, monitor='val_loss', verbose=1, save_best_only=True, mode='min')
history = model.fit(trainX, trainY, epochs=300,batch_size=30,verbose=1,validation_split=0.2,callbacks=[checkpoint])
```

Architectur
e

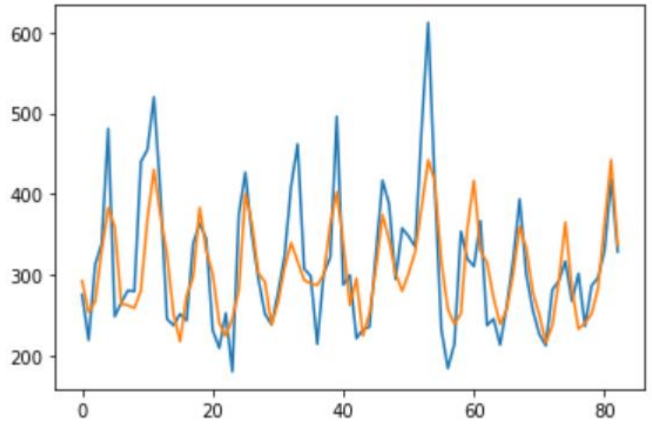
PURDUE UNIVERSITY

SUMMARY

Model	Time Series Intuition
ANN	Behaves like a non-linear version ARIMA! Use this when you want all your input variables to have a direct relationship with the output
RNN	We haven't demonstrated an RNN in this tutorial. RNNs have a permanent memory. All of the information is passed through to the next RNN unit
LSTM	Unlike RNNs, LSTM units do not pass on the entire information into the next unit. Rather, forget and input gates decide what go into the next LSTM unit

MULTIVARIATE LSTM PERFORMANCE

```
In [94]: model.load_weights("cnn-252-4237.62.hdf5")
         model.compile(optimizer='adam', loss='mse')
         y_test_pred= model.predict(testX)
         mape(testY, y_test_pred)

         3/3 [==============================] - 0s 11ms/step

Out[94]: 0.13471209087165434

In [95]: plt.plot(testY)
         plt.plot(y_test_pred)

Out[95]: [<matplotlib.lines.Line2D at 0x2b8353d83a30>]
```



Best MAPE so far!