

```

1 # LSTM (Many to Many Multiple Numeric Features)
2 # =====
3
4 # https://stackabuse.com/solving-sequence-problems-with-lstm-in-keras-part-2/
5
6 %tensorflow_version 2.x
7
8 import tensorflow as tf
9 tf.__version__

```

```

↳ TensorFlow 2.x selected.
   '2.0.0'

```

```

1 # univariate lstm example
2 import tensorflow as tf
3 import numpy as np
4 from numpy import array
5 from tensorflow.keras.models import Sequential
6 from tensorflow.keras.layers import LSTM, Bidirectional, Flatten, RepeatVector, TimeDistributed
7 from tensorflow.keras.layers import Dense, Dropout
8 from tensorflow.keras.callbacks import EarlyStopping
9 from tensorflow.python.keras.callbacks import TensorBoard
10 # from tensorflow.keras.regularizers import l2
11
12 import matplotlib.pyplot as plt
13 from time import time

```

```

1 # define dataset
2 X = list()
3 Y = list()
4 X1 = [x1 for x1 in range(5, 301, 5)]
5 X2 = [x2 for x2 in range(20, 316, 5)]
6 Y = [y for y in range(35, 331, 5)]
7
8 X = np.column_stack((X1, X2))
9 X[:3], Y[:3]

```

```

↳ (array([[ 5, 20],
          [10, 25],
          [15, 30]]), [35, 40, 45])

```

```

1 X = np.array(X).reshape(20, 3, 2)
2 Y = np.array(Y).reshape(20, 3, 1)
3
4 X = X.astype('float32')
5 y = Y.astype('float32')

```

```

1 X[:3], y[:3]

```

```

↳ (array([[[ 5., 20.],
            [10., 25.],
            [15., 30.]],

          [[20., 35.],
            [25., 40.],
            [30., 45.]],

          [[35., 50.],
            [40., 55.],
            [45., 60.]]], dtype=float32), array([[[35.],
            [40.],
            [45.]],

          [[50.],
            [55.],
            [60.]],

          [[65.],
            [70.],
            [75.]]], dtype=float32))

```

```

1 # X = tf.cast(X,tf.float32)
2 # y = tf.cast(y,tf.float32)

```

```

1 # %load_ext tensorboard
2 # tensorboard = TensorBoard(log_dir="logs/{}".format(time()), histogram_freq=1)
3 # %tensorboard --logdir logs

```

```

1 # es = EarlyStopping(monitor='val_loss', min_delta=0.1, patience=5, verbose=1, mode='min')

```

```

1 # define model
2
3 model = Sequential()
4 model.add(Bidirectional(LSTM(100, activation='relu', input_shape=(3, 2), return_sequences=True)))
5 model.add(RepeatVector(3))
6 model.add(Bidirectional(LSTM(100, activation='relu', return_sequences=True)))
7 model.add(TimeDistributed(Dense(1)))
8 model.compile(optimizer='adam', loss='mse', metrics=['mse'])
9 # history = model.fit(X, y, epochs=200, validation_split=0.2, batch_size=3, verbose=1)
10 history = model.fit(X, y, epochs=1000, validation_split=0.2, verbose=0)
11
12 model.summary()

```

```

↳

```

Model: "sequential\_1"

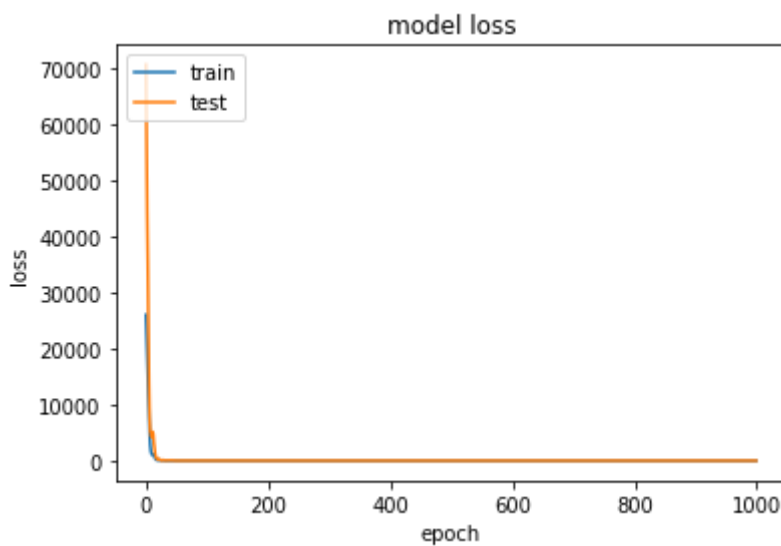
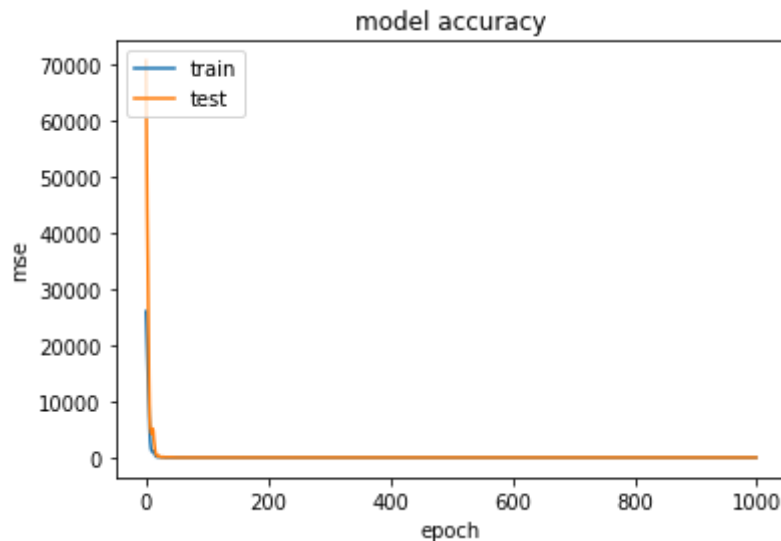
Layer (type)	Output Shape	Param #
bidirectional_2 (Bidirection multiple)		82400
repeat_vector_1 (RepeatVecto multiple)		0
bidirectional_3 (Bidirection multiple)		240800
time_distributed_1 (TimeDist multiple)		201
Total params: 323,401		
Trainable params: 323,401		
Non-trainable params: 0		

```
1 # fit model
2 # model.fit(X, y, epochs=500, validation_split=0.2, verbose=1, callbacks=[tensorboar
3 # history = model.fit(X, y, epochs=500, validation_split=0.2, verbose=0, callbacks=
```

```
1 # list all data in history
2 print(history.history.keys())
3
4 # summarize history for accuracy
5 plt.plot(history.history['mse'])
6 plt.plot(history.history['val_mse'])
7 plt.title('model accuracy')
8 plt.ylabel('mse')
9 plt.xlabel('epoch')
10 plt.legend(['train', 'test'], loc='upper left')
11 plt.show()
12
13 # summarize history for loss
14 plt.plot(history.history['loss'])
15 plt.plot(history.history['val_loss'])
16 plt.title('model loss')
17 plt.ylabel('loss')
18 plt.xlabel('epoch')
19 plt.legend(['train', 'test'], loc='upper left')
20 plt.show()
```



```
dict_keys(['loss', 'mse', 'val_loss', 'val_mse'])
```



```
1 # demonstrate prediction
2
3 X1 = [300, 305, 310]
4 X2 = [315, 320, 325]
5
6 test_input = np.column_stack((X1, X2))
7
8 test_input = test_input.reshape((1, 3, 2))
9 print(test_input)
10
11 test_input = test_input.astype('float32')
12 print(test_input)
13
14 print("expected : [330, 335, 340]")
15
16 yhat = model.predict(test_input, verbose=0)
17 print("yhat : ", yhat)
```

```
↳ [[ [300 315]
      [305 320]
      [310 325]]]
   [[ [300. 315.]
      [305. 320.]
      [310. 325.]]]
expected : [330, 335, 340]
yhat :    [[ [331.02054]
              [336.37103]
              [341.31903]]]
```

1