DLM_basic2

February 28, 2019

```
In [1]:
from keras.layers import Input, Dense, Dropout
from keras.models import Model
from keras.callbacks import EarlyStopping, ReduceLROnPlateau, CSVLogger
from sklearn.metrics import mean_squared_error
import numpy as np
import h5py
from matplotlib import rcParams  # next 3 lines set font family for plotting
rcParams['font.family'] = 'serif'
rcParams['font.sans-serif'] = ['TImes New Roman']
import matplotlib.pyplot as plt
import os
import time

# set working directory (change the following path to match your directory structure)
main = 'C:\\Users\\Kathy_Breen\\Documents\\DL_Seminar\\Week3'  # set directory path where
# this file is saved
os.chdir(main)  # make sure the Spyder is pointing to the correct folder
```

```
Using TensorFlow backend.
```

```
In [2]:
# SETTINGS FOR REPRODUCIBLE RESULTS DURING DEVELOPMENT

#import numpy as np
import tensorflow as tf
import random as rn

# The below is necessary in Python 3.2.3 onwards to
# have reproducible behavior for certain hash-based operations.
# See these references for further details:
# https://docs.python.org/3.4/using/cmdline.html#envvar-PYTHONHASHSEED
# https://github.com/keras-team/keras/issues/2280#issuecomment-306959926

#import os
os.environ['PYTHONHASHSEED'] = '0'
```

```
# The below is necessary for starting Numpy generated random numbers
# in a well-defined initial state.

np.random.seed(42)

# The below is necessary for starting core Python generated random numbers
# in a well-defined state.

rn.seed(12345)

# Force TensorFlow to use single thread.
# Multiple threads are a potential source of
# non-reproducible results.
# For further details, see: https://stackoverflow.com/questions/42022950/which-seeds-have-to-be-

session_conf = tf.ConfigProto(intra_op_parallelism_threads=1, inter_op_parallelism_threads=1)

from keras import backend as K

# The below tf.set_random_seed() will make random number generation
# in the TensorFlow backend have a well-defined initial state.
# For further details, see: https://www.tensorflow.org/api_docs/python/tf/set_random_seed

tf.set_random_seed(1234)

sess = tf.Session(graph=tf.get_default_graph(), config=session_conf)
K.set_session(sess)
```

In [3]:

```
# Read in *.hdf5 data sets

with h5py.File('X.hdf5','r') as f:
    X_train = np.array(f["X_train"])
    X_test = np.array(f["X_test"])

with h5py.File('Y.hdf5','r') as f:
    Y_train = np.array(f["Y_train"])
    Y_test = np.array(f["Y_test"])
```

In [4]:

```
#%% Build the DLM

tic = time.time()  # start a timer

# define number of X and Y features
X_Nfeatures = X_train.shape[1]
Y_Nfeatures = Y_train.shape[1]
```

```python
# HYPERPARAMETERS
epochs = 5000
batch_size = 5000
do = 0.2
N_nodes = 64

# create input layer..........
main_input = Input(shape=(X_Nfeatures),
                   dtype='float',
                   batch_shape=(batch_size,X_Nfeatures),
                   name='main_input'
                   )
#create hidden layer..........
hidden_layer1 = Dense(N_nodes, activation='relu', name='hidden_layer1')(main_input)
# add dropout to hidden layer
Dropout(do)(hidden_layer1)

hidden_layer2 = Dense(N_nodes, activation='relu', name='hidden_layer2')(hidden_layer1)
# add dropout to hidden layer
Dropout(do)(hidden_layer2)

hidden_layer3 = Dense(N_nodes, activation='relu', name='hidden_layer3')(hidden_layer2)
# add dropout to hidden layer
Dropout(do)(hidden_layer3)

hidden_layer4 = Dense(N_nodes, activation='relu', name='hidden_layer4')(hidden_layer3)
# add dropout to hidden layer
Dropout(do)(hidden_layer4)

# calculate auxiliary loss
aux_output1 = Dense(Y_Nfeatures, name='aux_output1')(hidden_layer4)

hidden_layer5 = Dense(N_nodes, activation='relu', name='hidden_layer5')(hidden_layer4)
# add dropout to hidden layer
Dropout(do)(hidden_layer5)

hidden_layer6 = Dense(N_nodes, activation='relu', name='hidden_layer6')(hidden_layer5)
# add dropout to hidden layer
Dropout(do)(hidden_layer6)

hidden_layer7 = Dense(N_nodes, activation='relu', name='hidden_layer7')(hidden_layer6)
# add dropout to hidden layer
Dropout(do)(hidden_layer7)

hidden_layer8 = Dense(N_nodes, activation='relu', name='hidden_layer8')(hidden_layer7)
# add dropout to hidden layer
Dropout(do)(hidden_layer8)
```

```
# calculate auxiliary loss
aux_output2 = Dense(Y_Nfeatures, name='aux_output2')(hidden_layer8)

hidden_layer9 = Dense(N_nodes, activation='relu', name='hidden_layer9')(hidden_layer8)
# add dropout to hidden layer
Dropout(do)(hidden_layer9)

hidden_layer10 = Dense(N_nodes, activation='relu', name='hidden_layer10')(hidden_layer9)
# add dropout to hidden layer
Dropout(do)(hidden_layer10)

hidden_layer11 = Dense(N_nodes, activation='relu', name='hidden_layer11')(hidden_layer10)
# add dropout to hidden layer
Dropout(do)(hidden_layer11)

hidden_layer12 = Dense(N_nodes, activation='relu', name='hidden_layer12')(hidden_layer11)
# add dropout to hidden layer
Dropout(do)(hidden_layer12)

# create output layer
main_output = Dense(Y_Nfeatures, name='main_output')(hidden_layer12)  # default activation is li

# feed datasets into model for training
model = Model(inputs=[main_input],
              outputs=[main_output, aux_output1, aux_output2]
              )

# compile the model with desired configuration
model.compile(loss='mean_squared_error',
              optimizer='adagrad',
              metrics=['mae'])

# one of several callbacks available in Keras, csv_logger saves metrics for every epoch to a csv
csv_logger = CSVLogger('training_' + str(epochs) + '.log')

early_stop = EarlyStopping(monitor='val_loss', # quantity to monitor
                           min_delta=0.0001,  # min change to qualify as an improvement
                           patience=10, # stop after #epochs with no improvement
                           verbose=1)  # print messages

reduce_lr = ReduceLROnPlateau(monitor='val_loss',
                              factor=0.2,  # reduction factor (new_lr = lr * factor)
                              patience=5,
                              verbose=1)

# train the model, and store training information in the history object
history = model.fit([X_train],[Y_train, Y_train, Y_train],
                    epochs=epochs,
                    batch_size = batch_size,
                    validation_data=([X_test], [Y_test, Y_test, Y_test]),
                    callbacks=[reduce_lr,early_stop,csv_logger]
                    )
model.summary()  # print out a summary of layers/parameters
config = model.get_config()  # detailed information about the configuration of each layer

# evaluate the trained model on the test data set
predict = model.predict([X_test],batch_size=batch_size)
Y_final_mse = mean_squared_error(predict[0],Y_test)
Y_aux1_mse = mean_squared_error(predict[1],Y_test)
Y_aux2_mse = mean_squared_error(predict[2],Y_test)
```

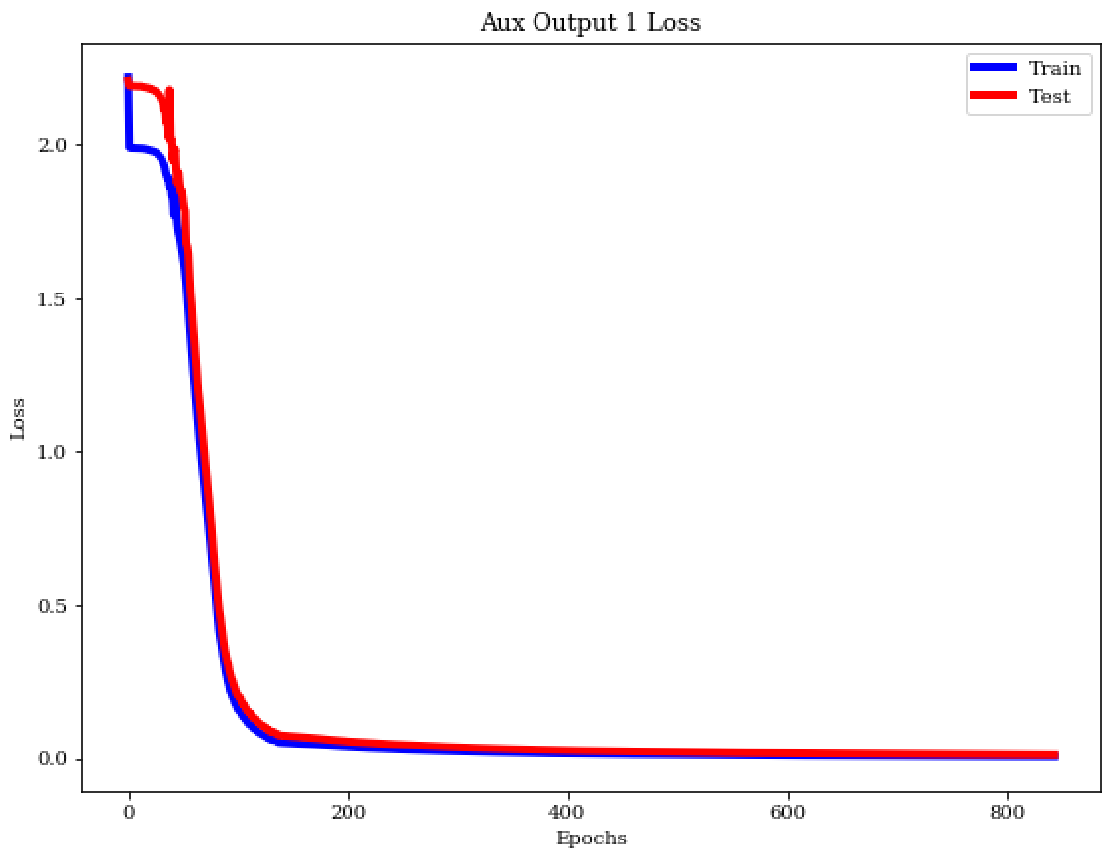
```
WARNING:tensorflow:From C:\Users\Kathy_Breen\AppData\Local\Continuum\anaconda3\lib\site-packages
Instructions for updating:
keep_dims is deprecated, use keepdims instead
Train on 90000 samples, validate on 10000 samples
Epoch 1/5000
90000/90000 [==============================] - 1s 10us/step
- loss: 7.0707
- main_output_loss: 2.6413
- aux_output1_loss: 2.2195
- aux_output2_loss: 2.2099
- main_output_mean_absolute_error: 1.0591
- aux_output1_mean_absolute_error: 0.8961
- aux_output2_mean_absolute_error: 0.9781
- val_loss: 6.5892
- val_main_output_loss: 2.1905
- val_aux_output1_loss: 2.2081
- val_aux_output2_loss: 2.1905
- val_main_output_mean_absolute_error: 0.9977
- val_aux_output1_mean_absolute_error: 0.9575
- val_aux_output2_mean_absolute_error: 0.9968
Epoch 2/5000
90000/90000 [==============================] - 0s 2us/step
- loss: 5.9686
- main_output_loss: 1.9889
- aux_output1_loss: 1.9909
- aux_output2_loss: 1.9888
- main_output_mean_absolute_error: 0.9711
- aux_output1_mean_absolute_error: 0.9575
- aux_output2_mean_absolute_error: 0.9738
- val_loss: 6.5748
- val_main_output_loss: 2.1919
- val_aux_output1_loss: 2.1915
- val_aux_output2_loss: 2.1914
- val_main_output_mean_absolute_error: 0.9880
- val_aux_output1_mean_absolute_error: 0.9892
- val_aux_output2_mean_absolute_error: 0.9900

...

...

Epoch 843/5000
90000/90000 [==============================] - 0s 2us/step
- loss: 0.0067
- main_output_loss: 2.0446e-04
- aux_output1_loss: 0.0063
- aux_output2_loss: 2.3128e-04
- main_output_mean_absolute_error: 0.0062
- aux_output1_mean_absolute_error: 0.0249
- aux_output2_mean_absolute_error: 0.0064
- val_loss: 0.0122
- val_main_output_loss: 3.9839e-04
- val_aux_output1_loss: 0.0113
- val_aux_output2_loss: 4.6801e-04
- val_main_output_mean_absolute_error: 0.0067
- val_aux_output1_mean_absolute_error: 0.0264
- val_aux_output2_mean_absolute_error: 0.0069
Epoch 844/5000
90000/90000 [==============================] - 0s 2us/step
- loss: 0.0067
- main_output_loss: 2.0446e-04
- aux_output1_loss: 0.0063
- aux_output2_loss: 2.3129e-04
- main_output_mean_absolute_error: 0.0062
- aux_output1_mean_absolute_error: 0.0249
- aux_output2_mean_absolute_error: 0.0064
- val_loss: 0.0122
- val_main_output_loss: 3.9836e-04
- val_aux_output1_loss: 0.0113
- val_aux_output2_loss: 4.6798e-04
- val_main_output_mean_absolute_error: 0.0067
- val_aux_output1_mean_absolute_error: 0.0264
- val_aux_output2_mean_absolute_error: 0.0069
Epoch 00844: early stopping
--------------------------------------------------------------------------------------------------
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
main_input (InputLayer)         (5000, 1)            0

--------------------------------------------------------------------------------------------------
hidden_layer1 (Dense)           (5000, 64)           128         main_input[0][0]

--------------------------------------------------------------------------------------------------
hidden_layer2 (Dense)           (5000, 64)           4160        hidden_layer1[0][0]

--------------------------------------------------------------------------------------------------
```

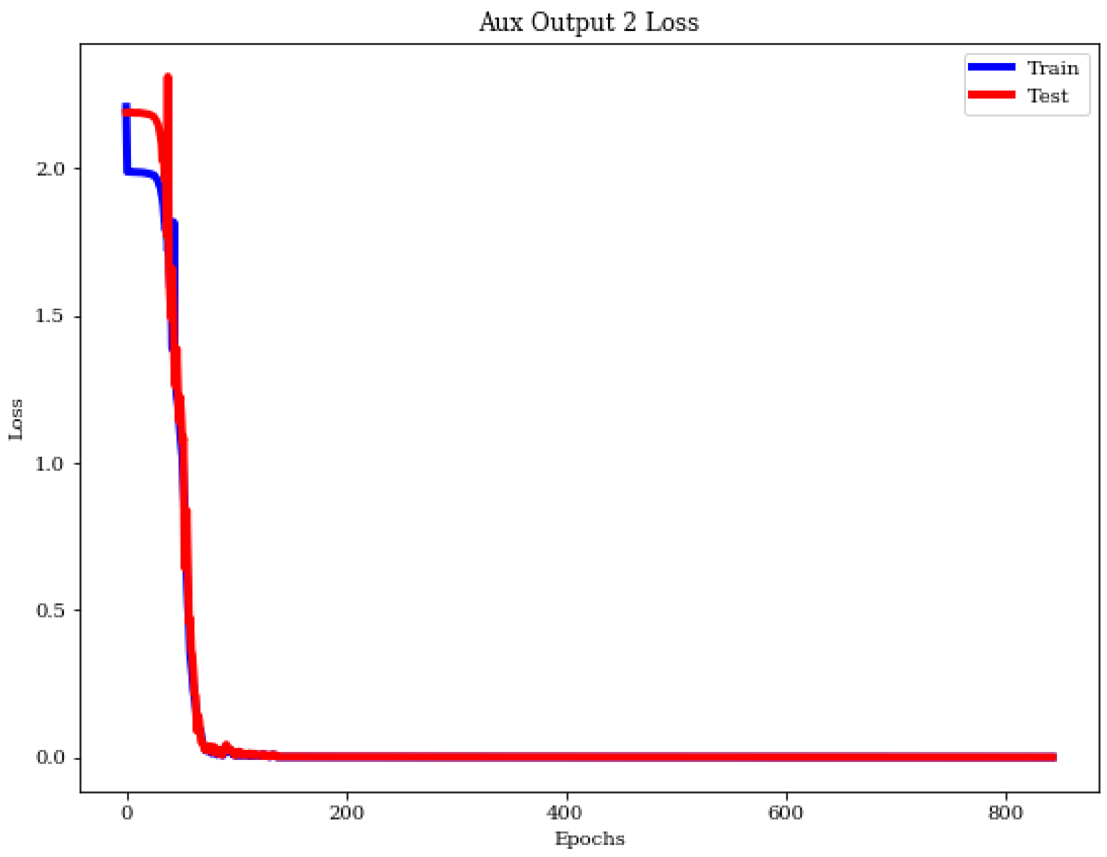
```
hidden_layer3 (Dense)           (5000, 64)          4160        hidden_layer2[0][0]
________________________________________________________________________________________________
hidden_layer4 (Dense)           (5000, 64)          4160        hidden_layer3[0][0]
________________________________________________________________________________________________
hidden_layer5 (Dense)           (5000, 64)          4160        hidden_layer4[0][0]
________________________________________________________________________________________________
hidden_layer6 (Dense)           (5000, 64)          4160        hidden_layer5[0][0]
________________________________________________________________________________________________
hidden_layer7 (Dense)           (5000, 64)          4160        hidden_layer6[0][0]
________________________________________________________________________________________________
hidden_layer8 (Dense)           (5000, 64)          4160        hidden_layer7[0][0]
________________________________________________________________________________________________
hidden_layer9 (Dense)           (5000, 64)          4160        hidden_layer8[0][0]
________________________________________________________________________________________________
hidden_layer10 (Dense)          (5000, 64)          4160        hidden_layer9[0][0]
________________________________________________________________________________________________
hidden_layer11 (Dense)          (5000, 64)          4160        hidden_layer10[0][0]
________________________________________________________________________________________________
hidden_layer12 (Dense)          (5000, 64)          4160        hidden_layer11[0][0]
________________________________________________________________________________________________
main_output (Dense)             (5000, 1)           65          hidden_layer12[0][0]
________________________________________________________________________________________________
aux_output1 (Dense)             (5000, 1)           65          hidden_layer4[0][0]
________________________________________________________________________________________________
aux_output2 (Dense)             (5000, 1)           65          hidden_layer8[0][0]
================================================================================================
Total params: 46,083
Trainable params: 46,083
Non-trainable params: 0
________________________________________________________________________________________________
```

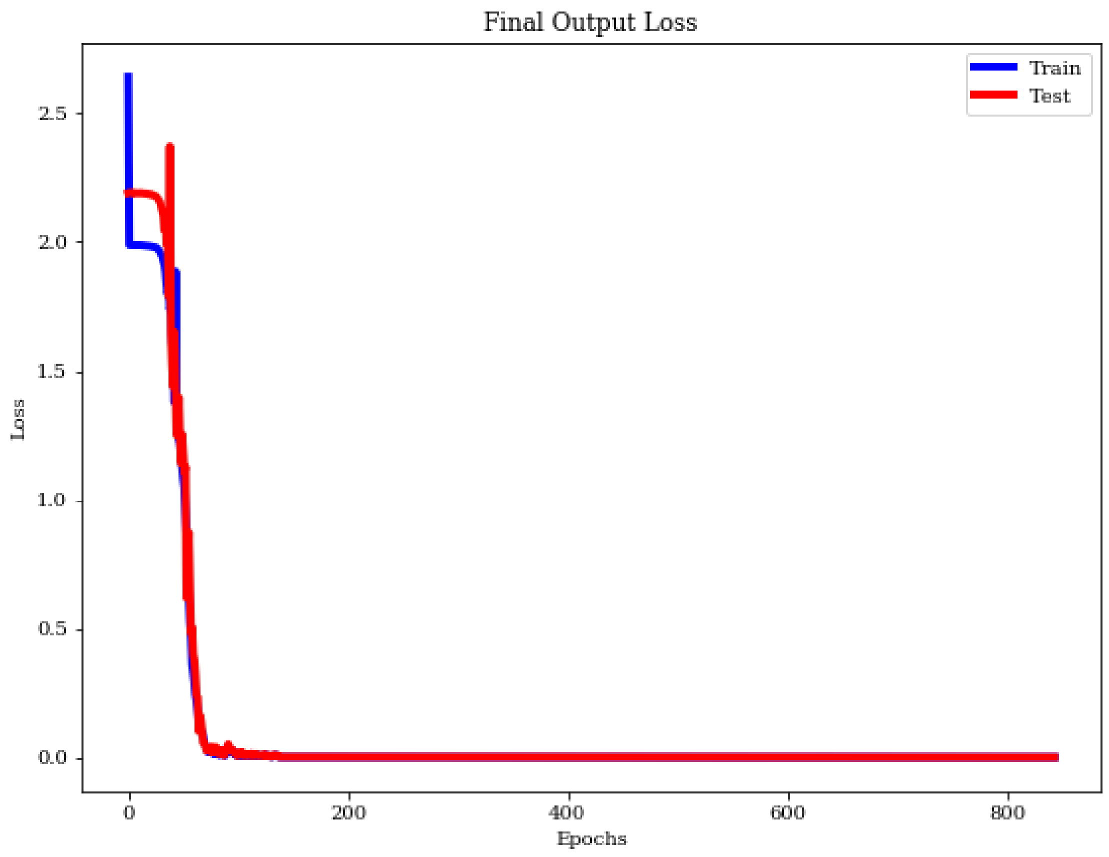
In [5]:

```
#%% plot

main_loss_train = history.history['main_output_loss']
main_loss_test = history.history['val_main_output_loss']
aux1_loss_train = history.history['aux_output1_loss']
aux1_loss_test = history.history['val_aux_output1_loss']
aux2_loss_train = history.history['aux_output2_loss']
aux2_loss_test = history.history['val_aux_output2_loss']
xplot = list(range(len(main_loss_train)))

fig = plt.figure(num=1, figsize=(8,6))
ax = fig.add_subplot(111)
train = ax.plot(xplot,aux1_loss_train,'b-',label='Train',linewidth=4)
test = ax.plot(xplot,aux1_loss_test,'r-',label='Test',linewidth=4)
ax.set_xlabel('Epochs')
ax.set_ylabel('Loss')
curves = train+test
labels = [c.get_label() for c in curves]
ax.legend(curves, labels, loc=0)
plt.tight_layout()
plt.title('Aux Output 1 Loss')
plt.savefig('aux1_loss' + str(epochs) + 'epochs2.pdf')
plt.show()

fig = plt.figure(num=2, figsize=(8,6))
ax = fig.add_subplot(111)
train = ax.plot(xplot,aux2_loss_train,'b-',label='Train',linewidth=4)
test = ax.plot(xplot,aux2_loss_test,'r-',label='Test',linewidth=4)
ax.set_xlabel('Epochs')
ax.set_ylabel('Loss')
curves = train+test
labels = [c.get_label() for c in curves]
ax.legend(curves, labels, loc=0)
plt.tight_layout()
plt.title('Aux Output 2 Loss')
plt.savefig('aux2_loss' + str(epochs) + 'epochs2.pdf')
plt.show()

fig = plt.figure(num=3, figsize=(8,6))
ax = fig.add_subplot(111)
train = ax.plot(xplot,main_loss_train,'b-',label='Train',linewidth=4)
test = ax.plot(xplot,main_loss_test,'r-',label='Test',linewidth=4)
ax.set_xlabel('Epochs')
ax.set_ylabel('Loss')
curves = train+test
labels = [c.get_label() for c in curves]
ax.legend(curves, labels, loc=0)
plt.tight_layout()
plt.title('Final Output Loss')
plt.savefig('main_loss' + str(epochs) + 'epochs2.pdf')
plt.show()

fig = plt.figure(num=4, figsize=(8,6))
ax = fig.add_subplot(111)
y_true = ax.plot(X_test,Y_test,'ko',markersize=16,label=r'$Y_{true}$')
y_aux1 = ax.plot(X_test,predict[1],'^',color='#F16D00',markersize=12,label=r'$\hat{Y}_{aux1}$')
y_aux2 = ax.plot(X_test,predict[2],'s',color='#F1D300',markersize=11,label=r'$\hat{Y}_{aux2}$')
y_pred = ax.plot(X_test,predict[0],'*',color='#009191',markersize=10,label=r'$\hat{Y}_{main}$')
ax.set_xlabel(r'$X$')
ax.set_ylabel(r'$Y$')
curves = y_true+y_aux1+y_aux2+y_pred
labels = [c.get_label() for c in curves]
ax.legend(curves, labels, loc=0)
```

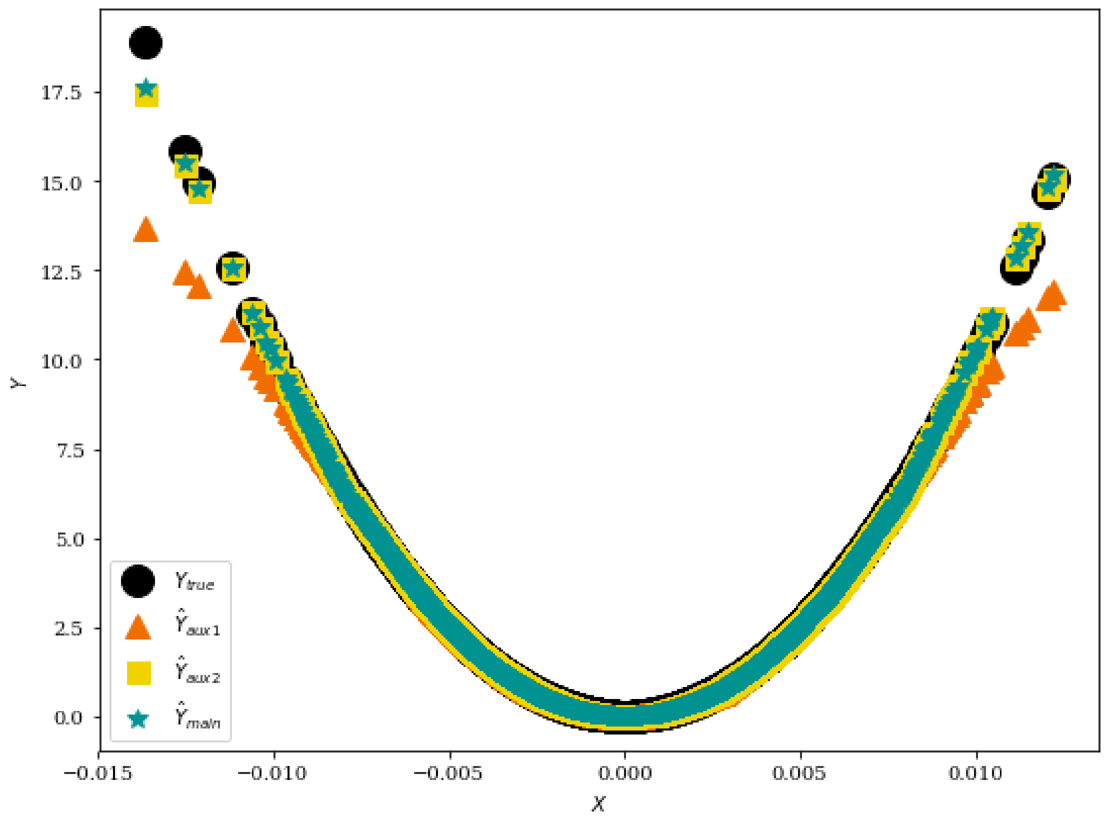
```
plt.tight_layout()
plt.savefig('ypred2.pdf')
plt.show()

#%% VISUALIZATION - this was really hard to get to work.  The following two forums helped me
 figure out the tricky package install...
# https://stackoverflow.com/questions/40632486/dot-exe-not-found-in-path-pydot-on
-python-windows-7
# https://www.codesofinterest.com/2017/02/visualizing-model-structures-in-keras.html
from keras.utils import plot_model
os.environ["PATH"] += os.pathsep + 'C:/Program Files (x86)/Graphviz2.38/release/bin'
plot_model(model, to_file='DLM_basic2_structure.pdf', show_shapes=True, show_layer_names=True)
```




Aux Output 2 Loss
Train
Test
Loss
Epochs
0
200
400
600
800
0.0
0.5
1.0
1.5
2.0



Final Output Loss
Train
Test
Loss
Epochs
0.0
0.5
1.0
1.5
2.0
2.5
0
200
400
600
800



17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
Y
Y_{true}
$\hat{Y}_{aux1}$
$\hat{Y}_{aux2}$
$\hat{Y}_{main}$
−0.015
−0.010
−0.005
0.000
0.005
0.010
X