CS 5600/6600: F24: Intelligent Systems
Assignment 6

Vladimir Kulyukin
Department of Computer Science
Utah State University

October 05, 2024

Learning Objectives

1. Discrete Time Series Forecasting (DUTS);
2. Training/Testing ANN, CNN, and LSTM Forecasters.

Introduction

You may want to review the Lecture 11 PDF (CS5600_6600_F24_RNN_LSTM_DUTS_Part_02.pdf) and/or class your notes for the general background on discrete time series forecasting. In this assignment, we will train and test several ANN, CNN, and LSTM forecasters to predict the hive weight and in-hive temperature of several honey bee colonies at a research apiary of the USDA Agricultural Research Service (USDA-ARS) in Tucson, AZ. The weight and temperature data were collected from 6/25/2022 to 9/22/2022. The data are in `WH_TH_HOURLY_MEANS_USDA_ARS_TUCSON_AZ_2022.csv`.

The time axis is the hour column. It starts at hour 0 and ends at hour 2160. The columns that start with `mwh` (mean weight of hive) contain the mean weight measurements for the hive whose id follows `mwh` (e.g., `mwh2123`) – one mean weight measurement (in kg) for every hour along the time axis. The columns that start with `mth` (mean temperature of hive in degrees Celsius) contain the mean in-hive temperature measurments for the hive whose id follows `mth` (e.g., `mth2123`).

Coding Lab

Let's go through a quick coding lab. The file `csv_aux.py` has a few auxiliary functions for extracting the weight and temperature measurements for specific hives. You do not have to change the code in this file.

We will use `keras` in this assignment. So, unless you have `keras` installed on your computer, you should start by installing it and making sure that the following imports work.

```
from keras.models import Sequential
from keras.models import load_model
from keras.layers import Dense
from keras.layers import Flatten
from keras.layers import LSTM
from keras.layers.convolutional import Conv1D
from keras.layers.convolutional import MaxPooling1D
```

You will write your code in `cs5600_6600_f24_hw06_duts_models.py` and in the hive-specific unit test files (more on this below). Let's open `cs5600_6600_f24_hw06_duts_models.py` and modify the value of the global variable `LOG_DIR`.

```
LOG_DIR = '/home/vladimir/Desktop/CS5600_6600_F24_LOG/'
```

This variable points to the directory where the trained models (i.e., `h5` files), the mean squared error (MSE) curve plots on the test data (`png` files), and the csv files that record the MSE of each forecaster will be saved. Change it to the location on your computer where you want that data to be.

Let's take a look at the function

```
def run_duts_wh_ann_aux(wh_train_series, wh_test_series,
                        num_in_steps=6, num_out_steps=2, num_epochs=3,
                        verbose=0,
                        hiveid=WH2059_MHR,
                        real_hive_id=2059,
                        save_flag=True).
```

This auxiliary function is called by

```
def run_duts_wh_ann(csv_fp, csv_index, train_percent=0.7,
                    hiveid=WH2120_MHR, real_hive_id=2120,
                    num_in_steps=6, num_out_steps=2, num_epochs=3,
                    verbose=0, save_flag=False)
```

defined right below it. The function `run_duts_wh_ann_aux` prepares the training and testing data; constructs an ANN forecasting model; trains the model on the training data; saves the model in an `h5` file; loads the trained model and tests the loaded model on the testing data; generates and saves the MSE plots; and logs the MSE in the csv MSE log file.

Let's quickly go through each step. Here's step 1.

```
# 1. Create a test dataset
X, y = split_univar_multi_step_sequence(wh_train_series,
                                        num_in_steps,
                                        num_out_steps)
```

The first argument is `wh_train_series` is an array of weight measurements. It looks like this.

```
[14.225025, 14.22438333, 14.22249167, 14.22125833, 14.219425, ...]
```

The second argument is the *intake* of the forecaster. The third argument is the *predictive horizon* of the forecaster. E.g., if `num_in_steps` is 6 and `num_out_steps` is 2, then we are training a weight forecaster that takes 6 previous mean hourly weight measurements and predicts the next 2 mean weight measurements. The span of this forecaster is $6 + 2 = 8$ hours.

After the split function is done, `X` will look like this

```
[[14.225025   14.22438333 14.22249167 14.22125833 14.219425   14.21149167]
 [14.22438333 14.22249167 14.22125833 14.219425   14.21149167 14.21420833]
 [14.22249167 14.22125833 14.219425   14.21149167 14.21420833 14.21578333]
 ...]
```

and `y` (i.e., the ground truth) will look as follows.

```
[[14.21420833 14.21578333]
 [14.21578333 14.214475  ]
 [14.214475   14.21693333]
 ...]
```

The second step is to construct an ANN forecaster.

```
# 2. Construct ANN model
model = Sequential()
model.add(Dense(5, input_shape=(num_in_steps, num_features), activation='relu'))
model.add(Flatten())
model.add(Dense(num_out_steps))
model.compile(optimizer='adam', loss='mse')
```

The `num_fatures` in our case is 1, because we are doing univariate time series forecasting.

In steps 3 and 4, we train the model on `X` and `y` for the specified number of epochs. We will limit the number of epochs to 10 to keep our training and testing times reasonable. The value of `verbose` is 1 or 0, If it is the former, we will see more `keras` diagnoistic messages. If it is the latter, there will be fewer messages. The model name is an automatically generated file name where the trained model will be saved.

```
# 3. fit model on the train data X, y
model.fit(X, y, epochs=num_epochs, verbose=verbose)

# 4. save model
sfx = str(datetime.datetime.today().timestamp()).replace('.', '_')
model_name = 'ann_duts_wh_{}_{}_{}_{}_{}.h5'.format(real_hive_id,
                                                    num_in_steps,
                                                    num_out_steps,
                                                    num_epochs, sfx)
model.save(LOG_DIR+model_name)
loaded_model = load_model(LOG_DIR + model_name)
```

In step 5, we split the testing data in the same way that we split the training data.

```
# 5. Construct test dataset
X2, y2 = split_univar_multi_step_sequence(wh_test_series,
                                          num_in_steps,
                                          num_out_steps)
```

In step 6, we test the trained model on the new (i.e., test) data. The data are new, because the model has not been trained on them. As we run the tests, we save the predictions and ground truths in two separate arrays that we will use in step 7 to compute the MSE.

```
# 6. test the model on new data
ground_truth, preds = [], []
for i in range(len(X2)):
    x_input_2 = X2[i].reshape((1, num_in_steps, num_features))
    y_hat_2   = loaded_model.predict(x_input_2)
    preds.append(y_hat_2[0][num_out_steps-1])
    ground_truth.append(y2[i][num_out_steps-1])
```

Here's step 7.

```
# 7. compute mse and plot
mse = (np.array(ground_truth) - np.array(preds))**2
mse = np.mean(mse)
mse_str = "%0.4f" % mse
model_name = model_name[:-3]
plot_title = '{}: wh; mse={}'.format(model_name, mse_str)
if save_flag:
   log_csv_mse(model_name, num_in_steps, num_out_steps, real_hive_id, mse_str)
plot_y_yhat(plot_title, 'x', 'y/yhat',
            ground_truth, preds,
            model_name, save_flag=save_flag)
```

Now we can run some unit tests. Let's open `cs5600_6600_f24_hw06_2130_uts.py` and uncomment the first test.

```
def test_run_duts_ann_hive_weight_forecaster_for_2130(self):
    run_duts_forecaster('WH_TH_HOURLY_MEANS_USDA_ARS_TUCSON_AZ_2022.csv',
                        'ann', 0.7, 2130, 10, 6, 2, 'wh')
```

This call does the 70/30 train test split of th data for hive 2130, trains the 6-2 (i.e., intake=6,horizon=2) ANN weight forecaster for 10 epochs on the 70% of the data and then tests it on 30% of the data. After we run this unit test, the `LOG_DIR` should have three files: the `h5` file with the saved ANN forecaster. The file name should be something like `ann_duts_wh_2130_6_2_10_XXX.h5`, where `XXX` is a time stamp. The prefix `ann_duts_wh` means that it is a ANN discrete univariate time series (DUTS) weight (wh) forecaster. The next number, i.e., 2130, is the hiveid. The numbers 6 and 2 denote the intake and the horizon, and the number 10 is the number of epochs.

The PNG MSE file should look something like the top curve or the bottom curve in Figure 1. If the model predicts well on the test data, the curve will be like the top one. Of course, the actual shape of the curve will depend on a particular hive and the topology of its test data. If the model does not predict well, the curve should look like the bottom curve.

The newly generated file `LOG_6_2_WT_TH.csv` should have a new row that should look as follows.

```
ann_duts_wh_2130_6_2_10_XXX, 6, 2, 2130, 0.0017
```

The first entry is the model name, followed by the intake, the horizon, the hiveid, and the MSE on the test data.

Now you can take a look at the function

```
def run_duts_th_ann_aux(th_train_series, th_test_series,
                        num_in_steps=12, num_out_steps=2,
                        num_epochs=3, verbose=1,
                        hiveid=TH2120_MHR, real_hive_id=2120,
                        save_flag=True)
```

and run the unit test

```
def test_run_duts_ann_hive_temp_forecaster_for_2130(self):
    run_duts_forecaster('WH_TH_HOURLY_MEANS_USDA_ARS_TUCSON_AZ_2022.csv',
                        'ann', 0.7, 2130, 10, 6, 2, 'th')
```

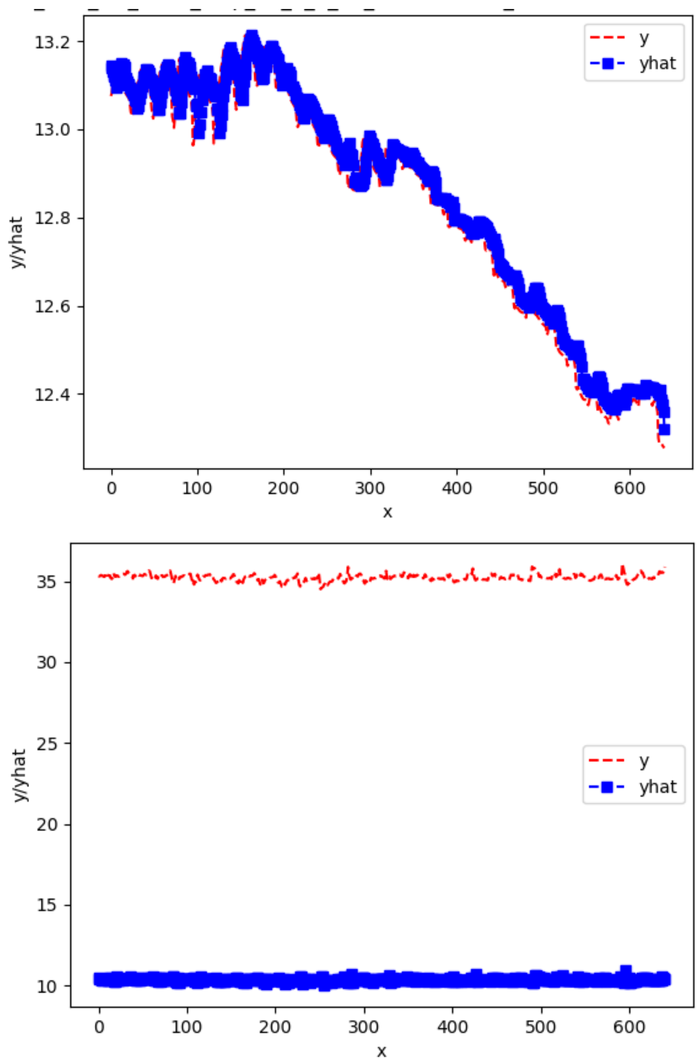

Figure 1: **Two Sample MSE Weight Curves.** The top curve shows that the trained model predicts well on the test data. The bottom curve demonstrates that the trained model is a weak predictor on the test data.

to train and test an ANN 6-2 temperature forecaster for hive 2130 with the same train/test split and the same number of epochs and examine the MSE test data plot and the entry in the log file.

If our forecaster does not show a good fit on the test data, we can attempt to train and test another one of the same kind and see if it predicts better on the test data.

Problem 1 (2 points)

In `cs5600_6600_f24_hw06_duts_models.py`, implement the functions

```
def run_duts_wh_cnn_aux(wh_train_series, wh_test_series,
                        num_in_steps=12, num_out_steps=2,
                        num_epochs=3, verbose=1,
                        hiveid=WH2120_MHR, real_hive_id=2120,
                        save_flag=True)
```

```
def run_duts_th_cnn_aux(th_train_series, th_test_series,
                        num_in_steps=12, num_out_steps=2,
                        num_epochs=3, verbose=1,
                        hiveid=TH2120_MHR, real_hive_id=2120,
                        save_flag=True):
```

to train and test CNN weight and temperature forecasters. The logical steps of these functions are the same as the logical steps of `run_duts_wh_ann_aux` `run_duts_th_ann_aux`. Your CNN forecasters should have the following architecture.

```
model = Sequential()
model.add(Conv1D(filters=5, kernel_size=2, activation='relu',
                 input_shape=(num_in_steps, num_features)))
model.add(MaxPooling1D(pool_size=2))
model.add(Flatten())
model.add(Dense(10, activation='relu'))
model.add(Dense(num_out_steps))
model.compile(optimizer='adam', loss='mse')
```

Now implement the functions

```
def run_duts_wh_lstm_aux(wh_train_series, wh_test_series,
                         num_in_steps=12, num_out_steps=2,
                         num_epochs=3, verbose=1,
                         hiveid=WH2120_MHR, real_hive_id=2120,
                         save_flag=True)

def run_duts_th_lstm_aux(th_train_series, th_test_series,
                         num_in_steps=12, num_out_steps=2,
                         num_epochs=3, verbose=1,
                         hiveid=TH2120_MHR, real_hive_id=2120,
                         save_flag=True):
```

to train and test LSTM weight and temperature forecasters. The logical steps of these functions are the same as the logical steps of `run_duts_wh_ann/cnn_aux` and `run_duts_th_ann/cnn_aux`. Your LSTM forecasters should have the following architecture.

```
model = Sequential()
model.add(LSTM(10, activation='relu', input_shape=(num_in_steps, num_features)))
model.add(Dense(num_out_steps))
model.compile(optimizer='adam', loss='mse')
```

The zip includes 10 unit test (ut) files, one unit test file per hive.

1. `cs5600_6600_f24_hw06_2059_uts.py`;
2. `cs5600_6600_f24_hw06_2120_uts.py`;
3. `cs5600_6600_f24_hw06_2123_uts.py`;
4. `cs5600_6600_f24_hw06_2129_uts.py`;
5. `cs5600_6600_f24_hw06_2130_uts.py`;

6. `cs5600_6600_f24_hw06_2137_uts.py`;
7. `cs5600_6600_f24_hw06_2141_uts.py`;
8. `cs5600_6600_f24_hw06_2142_uts.py`;
9. `cs5600_6600_f24_hw06_2146_uts.py`;
10. `cs5600_6600_f24_hw06_2158_uts.py`.

I have commented out all the tests. Run the unit tests for all hives and generate the MSE plots for the 6-2 (intake=6,horizon=2) forecasters and write two pages of your observations. Your documentation for this problem should contain the following 2 tables. The first one is for the 6-2 weight forecasters, where each cell is your best MSE value for the corresponding forecaster and hive. The second table is an analogous table for your 6-2 temperature forecasters.

`MSE for 6-2 weight foreacsters`

	2123	2120	2137	2059	2141	2142	2129	2158	2130	2146
ANN										
CNN										
LSTM										

`MSE for 6-2 temperature foreacsters`

	2123	2120	2137	2059	2141	2142	2129	2158	2130	2146
ANN										
CNN										
LSTM										

Throw in a couple of your MSE plots to illustrate your points. Did you see any performance differences between ANNs, CNNs, and LSTMs on this time span (8 hours)? How many times did you have to retrain a model to get a better fit on the test data?

Problem 2 (3 points)

Let's extend our time span to 18 hours and train and test the 6-12 forecasters. Add the appropriate unit tests for each hive-specific unit test file to generate the MSE plots and get the MSE values for all three types of forecasters. Limit the number of training epochs to 10. You can write one giant unit test that includes all cases or follow my example and write a single unit test for each case. I find the latter case-specific approach conceptually easier. But, hacking patterns differ from researcher to researcher. Add another 2 pages to your report by including in it the following tables for 6-12 weight and temperature forecasters.

`MSE for 6-12 weight foreacsters`

	2123	2120	2137	2059	2141	2142	2129	2158	2130	2146
ANN										
CNN										
LSTM										

MSE for 6-12 temperature foreacsters

	2123	2120	2137	2059	2141	2142	2129	2158	2130	2146
ANN										
CNN										
LSTM										

Again, you may want to include a couple MSE plots to illustrate your points. Did you see any performance differences between ANNs, CNNs, and LSTMs on the longer time span? Was it better than the performance of the 6-2 forecasters? Why do you think there is a difference or no difference? How many times did you have to retrain a model to get a better fit on the test data?

What to Submit

1. `cs5600_6600_f24_hw06_duts_models.py` with your forecaster training and tesing functions;
2. the unit test files with your code to generate MSE test plots; it took me about 1.5 hours to run all of them on my laptop;
3. the 4-page report `hw06_report.pdf` on the performance of your 6-2 and 6-12 ANN, CNN, and LSTM forcasters.

Happy Hacking, Thinking, and Writing!