HAR_LSTM

March 16, 2019

```
In [1]: # Importing Libraries

In [2]: import pandas as pd
        import numpy as np

In [3]: # Activities are the class labels
        # It is a 6 class classification
        ACTIVITIES = {
            0: 'WALKING',
            1: 'WALKING_UPSTAIRS',
            2: 'WALKING_DOWNSTAIRS',
            3: 'SITTING',
            4: 'STANDING',
            5: 'LAYING',
        }

        # Utility function to print the confusion matrix
        def confusion_matrix(Y_true, Y_pred):
            Y_true = pd.Series([ACTIVITIES[y] for y in np.argmax(Y_true, axis=1)])
            Y_pred = pd.Series([ACTIVITIES[y] for y in np.argmax(Y_pred, axis=1)])

            return pd.crosstab(Y_true, Y_pred, rownames=['True'], colnames=['Pred'])
```

0.0.1 Data

```
In [4]: # Data directory
        DATADIR = 'UCI_HAR_Dataset'

In [5]: # Raw data signals
        # Signals are from Accelerometer and Gyroscope
        # The signals are in x,y,z directions
        # Sensor signals are filtered to have only body acceleration
        # excluding the acceleration due to gravity
        # Triaxial acceleration from the accelerometer is total acceleration
        SIGNALS = [
            "body_acc_x",
            "body_acc_y",
            "body_acc_z",
```

```
        "body_gyro_x",
        "body_gyro_y",
        "body_gyro_z",
        "total_acc_x",
        "total_acc_y",
        "total_acc_z"
    ]
```

In [6]:
```
# Utility function to read the data from csv file
def _read_csv(filename):
    return pd.read_csv(filename, delim_whitespace=True, header=None)

# Utility function to load the load
def load_signals(subset):
    signals_data = []

    for signal in SIGNALS:
        filename = f'UCI_HAR_Dataset/{subset}/Inertial Signals/{signal}_{subset}.txt'
        signals_data.append(
            _read_csv(filename).as_matrix()
        )

    # Transpose is used to change the dimensionality of the output,
    # aggregating the signals by combination of sample/timestep.
    # Resultant shape is (7352 train/2947 test samples, 128 timesteps, 9 signals)
    return np.transpose(signals_data, (1, 2, 0))
```

In [7]:
```
def load_y(subset):
    """
    The objective that we are trying to predict is a integer, from 1 to 6,
    that represents a human activity. We return a binary representation of
    every sample objective as a 6 bits vector using One Hot Encoding
    (https://pandas.pydata.org/pandas-docs/stable/generated/pandas.get_dummies.html)
    """
    filename = f'UCI_HAR_Dataset/{subset}/y_{subset}.txt'
    y = _read_csv(filename)[0]

    return pd.get_dummies(y).as_matrix()
```

In [8]:
```
def load_data():
    """
    Obtain the dataset from multiple files.
    Returns: X_train, X_test, y_train, y_test
    """
    X_train, X_test = load_signals('train'), load_signals('test')
    y_train, y_test = load_y('train'), load_y('test')

    return X_train, X_test, y_train, y_test
```

```
In [9]: # Importing tensorflow
        np.random.seed(42)
        import tensorflow as tf
        tf.set_random_seed(42)
```

```
In [10]: # Configuring a session
         session_conf = tf.ConfigProto(
             intra_op_parallelism_threads=1,
             inter_op_parallelism_threads=1
         )
```

```
In [11]: # Import Keras
         from keras import backend as K
         sess = tf.Session(graph=tf.get_default_graph(), config=session_conf)
         K.set_session(sess)
```

```
Using TensorFlow backend.
```

```
In [12]: # Importing libraries
         from keras.models import Sequential
         from keras.layers import LSTM
         from keras.layers.core import Dense, Dropout
```

```
In [13]: # Initializing parameters
         epochs = 30
         batch_size = 16
         n_hidden = 32
```

```
In [14]: # Utility function to count the number of classes
         def _count_classes(y):
             return len(set([tuple(category) for category in y]))
```

```
In [15]: # Loading the train and test data
         X_train, X_test, Y_train, Y_test = load_data()
```

```
c:\users\dell\appdata\local\programs\python\python36\lib\site-packages\ipykernel_launcher.py:1
  if sys.path[0] == '':
```

```
In [16]: timesteps = len(X_train[0])
         input_dim = len(X_train[0][0])
         n_classes = _count_classes(Y_train)

         print(timesteps)
         print(input_dim)
         print(len(X_train))
```

```
128
9
7352
```

- Defining the Architecture of LSTM

```
In [17]: # Initiliazing the sequential model
         model = Sequential()
         # Configuring the parameters
         model.add(LSTM(n_hidden, input_shape=(timesteps, input_dim)))
         # Adding a dropout layer
         model.add(Dropout(0.5))
         # Adding a dense output layer with sigmoid activation
         model.add(Dense(n_classes, activation='sigmoid'))
         model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_1 (LSTM)                (None, 32)                5376
_________________________________________________________________
dropout_1 (Dropout)          (None, 32)                0
_________________________________________________________________
dense_1 (Dense)              (None, 6)                 198
=================================================================
Total params: 5,574
Trainable params: 5,574
Non-trainable params: 0
_________________________________________________________________
```

```
In [18]: # Compiling the model
         model.compile(loss='categorical_crossentropy',
                       optimizer='rmsprop',
                       metrics=['accuracy'])
```

```
In [20]: # Training the model
         model.fit(X_train,
                   Y_train,
                   batch_size=batch_size,
                   validation_data=(X_test, Y_test),
                   epochs=epochs)
```

```
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 28s 4ms/step - loss: 0.6905 - acc: 0.6606 - val_lo
Epoch 2/30
7352/7352 [==============================] - 28s 4ms/step - loss: 0.6376 - acc: 0.6732 - val_lo
Epoch 3/30
7352/7352 [==============================] - 26s 3ms/step - loss: 0.6057 - acc: 0.6850 - val_lo
Epoch 4/30
7352/7352 [==============================] - 27s 4ms/step - loss: 0.5705 - acc: 0.7044 - val_lo
Epoch 5/30
```

```
7352/7352 [==============================] - 28s 4ms/step - loss: 0.5582 - acc: 0.7179 - val_lo
Epoch 6/30
7352/7352 [==============================] - 26s 4ms/step - loss: 0.5528 - acc: 0.7349 - val_lo
Epoch 7/30
7352/7352 [==============================] - 26s 3ms/step - loss: 0.4966 - acc: 0.7831 - val_lo
Epoch 8/30
7352/7352 [==============================] - 26s 3ms/step - loss: 0.4583 - acc: 0.7956 - val_lo
Epoch 9/30
7352/7352 [==============================] - 26s 4ms/step - loss: 0.4308 - acc: 0.8048 - val_lo
Epoch 10/30
7352/7352 [==============================] - 26s 4ms/step - loss: 0.4154 - acc: 0.8111 - val_lo
Epoch 11/30
7352/7352 [==============================] - 26s 4ms/step - loss: 0.4117 - acc: 0.8105 - val_lo
Epoch 12/30
7352/7352 [==============================] - 27s 4ms/step - loss: 0.4004 - acc: 0.8403 - val_lo
Epoch 13/30
7352/7352 [==============================] - 26s 4ms/step - loss: 0.3604 - acc: 0.8645 - val_lo
Epoch 14/30
7352/7352 [==============================] - 25s 3ms/step - loss: 0.3069 - acc: 0.9023 - val_lo
Epoch 15/30
7352/7352 [==============================] - 26s 3ms/step - loss: 0.2778 - acc: 0.9195 - val_lo
Epoch 16/30
7352/7352 [==============================] - 26s 4ms/step - loss: 0.2440 - acc: 0.9270 - val_lo
Epoch 17/30
7352/7352 [==============================] - 26s 4ms/step - loss: 0.2062 - acc: 0.9361 - val_lo
Epoch 18/30
7352/7352 [==============================] - 27s 4ms/step - loss: 0.2099 - acc: 0.9376 - val_lo
Epoch 19/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.2098 - acc: 0.9339 - val_lo
Epoch 20/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.1825 - acc: 0.9442 - val_lo
Epoch 21/30
7352/7352 [==============================] - 28s 4ms/step - loss: 0.2035 - acc: 0.9402 - val_lo
Epoch 22/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.2215 - acc: 0.9338 - val_lo
Epoch 23/30
7352/7352 [==============================] - 31s 4ms/step - loss: 0.1811 - acc: 0.9433 - val_lo
Epoch 24/30
7352/7352 [==============================] - 32s 4ms/step - loss: 0.1776 - acc: 0.9430 - val_lo
Epoch 25/30
7352/7352 [==============================] - 31s 4ms/step - loss: 0.1702 - acc: 0.9427 - val_lo
Epoch 26/30
7352/7352 [==============================] - 27s 4ms/step - loss: 0.1818 - acc: 0.9463 - val_lo
Epoch 27/30
7352/7352 [==============================] - 26s 4ms/step - loss: 0.1806 - acc: 0.9438 - val_lo
Epoch 28/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.2104 - acc: 0.9404 - val_lo
Epoch 29/30
```

```
7352/7352 [==============================] - 26s 4ms/step - loss: 0.2942 - acc: 0.9336 - val_lo
Epoch 30/30
7352/7352 [==============================] - 32s 4ms/step - loss: 0.1822 - acc: 0.9422 - val_lo
```

Out[20]: <keras.callbacks.History at 0x1a443cffe80>

In [21]:
```
# Confusion Matrix
print(confusion_matrix(Y_test, model.predict(X_test)))
```

```
Pred                LAYING  SITTING  STANDING  WALKING  WALKING_DOWNSTAIRS  \
True
LAYING                 510        0        27        0                   0
SITTING                  2      443        46        0                   0
STANDING                 0      166       365        1                   0
WALKING                  0        1         0      454                  22
WALKING_DOWNSTAIRS       0        0         0        0                 411
WALKING_UPSTAIRS         0        0         0       17                  21

Pred                WALKING_UPSTAIRS
True
LAYING                             0
SITTING                            0
STANDING                           0
WALKING                           19
WALKING_DOWNSTAIRS                 9
WALKING_UPSTAIRS                 433
```

In [22]:
```
score = model.evaluate(X_test, Y_test)
```

```
2947/2947 [==============================] - 1s 376us/step
```

In [23]:
```
score
```

Out[23]: [0.33535143116306826, 0.8876823888700374]

- With a simple 2 layer architecture we got 88.76% accuracy and a loss of 0.33
- We can further imporve the performace with Hyperparameter tuning

In [ ]: