Neha Panchal (NUID: 001491279)

Human-Activity-Recognition Using Smartphones Data Set Report

The project includes 2 ipython notebooks:

1. Human Activity Recognization_EDA.ipynb which does Data pre-processing
2. HAR_LSTM_1.15.ipynb which have 2 LSTM models using TensorFlow 1.15 on raw timeseries data

Dataset

The dataset can be downloaded from

https://archive.ics.uci.edu/ml/datasets/human+activity+recognition+using+smartphones

Problem and need

This project aims to build a model that predicts the human activities such as Walking, Walking Upstairs, Walking Downstairs, Sitting, Standing and Laying from the Sensor data of smart phones.

Every modern Smart Phone has a number of [sensors] we are interested in two of the sensors Accelerometer and Gyroscope. The data is recorded with the help of sensors. This is a 6-class classification problem as we have 6 activities to detect.

The second part uses the raw time series windowed data to train (Long Short-term Memory) LSTM models. The LSTM models are semi tuned manually to fast forward the tuning task.

Human Activity Recognition database is built from the recordings of 30 persons performing activities of daily living (ADL) while carrying a waist-mounted smartphone with embedded inertial sensors (accelerometer and Gyroscope).

Activities

- Walking
- Walking Upstairs
- Walking Downstairs
- Sitting
- Standing
- Laying

Understanding the dataset

How data was recorded: By using the sensors (Gyroscope and accelerometer) in a smartphone, they have captured '3-axial linear acceleration'(_tAcc-XYZ_) from accelerometer and '3-axial angular velocity' (_tGyro-XYZ_) from Gyroscope with several variations.

Prefix 't' in those metrics denotes time.
Suffix 'XYZ' represents 3-axial signals in X, Y, and Z directions.

We get a feature vector of 561 features and these features are given in the dataset. Each window of readings is a datapoint of 561 features, and we have 10299 readings.

Feature names

1. These sensor signals are preprocessed by applying noise filters and then sampled in fixed-width windows (sliding windows) of 2.56 seconds each with 50% overlap. ie., each window has 128 readings.

2. From Each window, a feature vector was obtained by calculating variables from the time and frequency domain. In our dataset, each datapoint represents a window with different readings

3. The acceleration signal was separated into Body and Gravity acceleration signals (***tBodyAcc-XYZ*** and ***tGravityAcc-XYZ***) using some low pass filter with corner frequecy of 0.3Hz.

4. After that, the body linear acceleration and angular velocity were derived in time to obtian *jerk signals* (***tBodyAccJerk-XYZ*** and ***tBodyGyroJerk-XYZ***).

5. The magnitude of these 3-dimensional signals was calculated using the Euclidian norm. These magnitudes are represented as features with names like *tBodyAccMag_, _tGravityAccMag_, _tBodyAccJerkMag_, _tBodyGyroMag* and *tBodyGyroJerkMag*.

6. Finally, we've got frequency domain signals from some of the available signals by applying a FFT (Fast Fourier Transform). These signals obtained were labeled with ***prefix 'f'*** just like original signals with ***prefix 't'***. These signals are labeled as ***fBodyAcc-XYZ***, ***fBodyGyroMag*** etc.

7. These are the signals that we got so far.
    - tBodyAcc-XYZ
    - tGravityAcc-XYZ
    - tBodyAccJerk-XYZ
    - tBodyGyro-XYZ
    - tBodyGyroJerk-XYZ
    - tBodyAccMag
    - tGravityAccMag
    - tBodyAccJerkMag
    - tBodyGyroMag
    - tBodyGyroJerkMag
    - fBodyAcc-XYZ
    - fBodyAccJerk-XYZ
    - fBodyGyro-XYZ
    - fBodyAccMag
    - fBodyAccJerkMag
    - fBodyGyroMag
    - fBodyGyroJerkMag
8. We can esitmate some set of variables from the above signals. ie., We will estimate the following properties on each and every signal that we recoreded so far.
    - ***mean()***: Mean value
    - ***std()***: Standard deviation
    - ***mad()***: Median absolute deviation
    - ***max()***: Largest value in array
    - ***min()***: Smallest value in array
    - ***sma()***: Signal magnitude area
    - ***energy()***: Energy measure. Sum of the squares divided by the number of values.
    - ***iqr()***: Interquartile range
    - ***entropy()***: Signal entropy
    - ***arCoeff()***: Autorregresion coefficients with Burg order equal to 4
    - ***correlation()***: correlation coefficient between two signals
    - ***maxInds()***: index of the frequency component with largest magnitude
    - ***meanFreq()***: Weighted average of the frequency components to obtain a mean frequency

- ***skewness()***: skewness of the frequency domain signal
- ***kurtosis()***: kurtosis of the frequency domain signal
- ***bandsEnergy()***: Energy of a frequency interval within the 64 bins of the FFT of each window.
- ***angle()***: Angle between to vectors.

9. *We can obtain some other vectors by taking the average of signals in a single window sample. These are used on the angle () variable*
    - *gravityMean*
    - *tBodyAccMean*
    - *tBodyAccJerkMean*
    - *tBodyGyroMean*
    - *tBodyGyroJerkMean*

Y_Labels(Encoded)

- In the dataset, Y_labels are represented as numbers from 1 to 6 as their identifiers.
    - WALKING as **1**
    - WALKING_UPSTAIRS as **2**
    - WALKING_DOWNSTAIRS as **3**
    - SITTING as **4**
    - STANDING as **5**
    - LAYING as **6**

Train, test and validation data were separated

- The readings from ***70%*** of the volunteers were taken as ***trianing data*** and remaining ***30%*** subjects recordings were taken for ***test data***
- Train data is further divided in train and valid for validation purpose after proper data cleaning

Data

- All the data is present in 'UCI_HAR_dataset/' folder in present working directory.
    - Feature names are present in 'UCI_HAR_dataset/features.txt'
    - ***Train Data***
        - 'UCI_HAR_dataset/train/X_train.txt'
        - 'UCI_HAR_dataset/train/subject_train.txt'
        - 'UCI_HAR_dataset/train/y_train.txt'
    - ***Test Data***
        - 'UCI_HAR_dataset/test/X_test.txt'
        - 'UCI_HAR_dataset/test/subject_test.txt'
        - 'UCI_HAR_dataset/test/y_test.txt'

Analysis

This part is included in Human Activity Recognization_EDA.ipynb file and it does data cleaning for null values and duplicate data. Here, I also tried finding if there is imbalance in data distribution. Replacement of feature names as mentioned above is implemented here.

Finally, I saved train and test files in csv format for better usability in further machine learning.

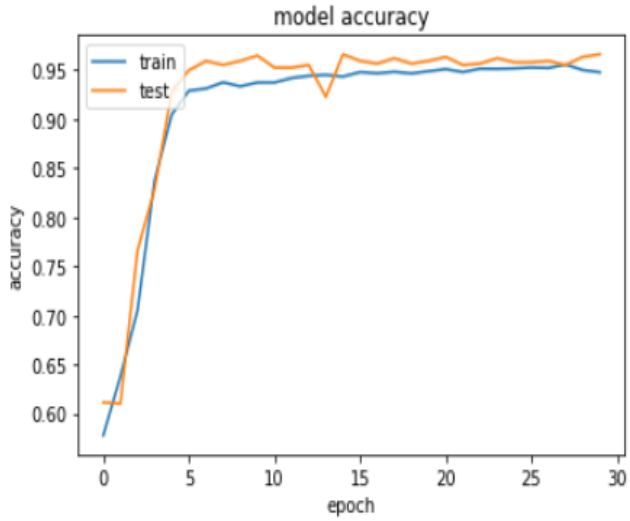
LSTM Model

LSTM model is implemented in this file: HAR_LSTM_1.15.ipynb which uses keras with tensorflow 1.15 backend. LSTM models need large amount of data to train properly, we also need to be cautious not to overfit.

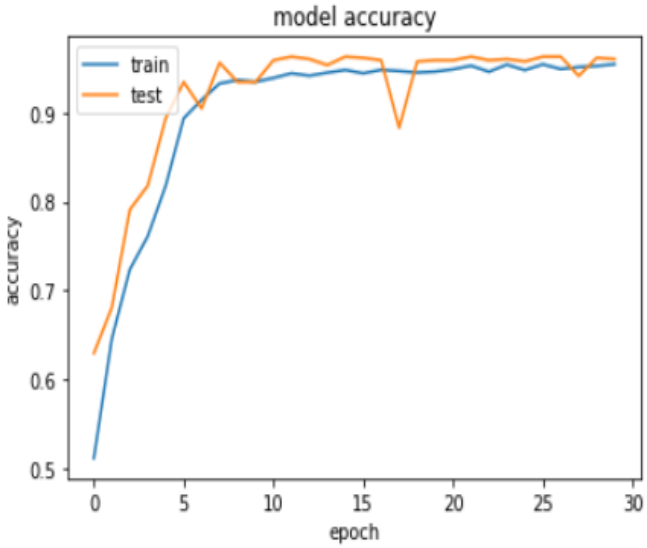
The raw series data is used to train the LSTM models, and not the heavily featured data. LSTM models require large amount of compute power.

Metrics for Accuracy:

Model 1:



Model 2:



Here, confusion matrix is evaluated for Pred(True) against Y_test data we have as below.

```
# Confusion Matrix
print(confusion_matrix(Y_test, model1.predict(X_test)))
```

```
Pred                LAYING  SITTING  STANDING  WALKING  WALKING_DOWNSTAIRS  \
True
LAYING                 537        0         0        0                   0
SITTING                  7      392        90        0                   1
STANDING                 0       99       433        0                   0
WALKING                  0        0         0      445                  47
WALKING_DOWNSTAIRS       0        0         0        0                 420
WALKING_UPSTAIRS         0        0         0        1                  10

Pred                WALKING_UPSTAIRS
True
LAYING                             0
SITTING                            1
STANDING                           0
WALKING                            4
WALKING_DOWNSTAIRS                 0
WALKING_UPSTAIRS                 460
```

```
score1 = model1.evaluate(X_test, Y_test)
```

```
2947/2947 [==============================] - 4s 1ms/step
```

```
score1
```

```
]: [0.3891164319865662, 0.9117746949195862]
```

With Finely tuned LSTM model, I achieved 0.91% accuracy and 0.38% score.

References:

http://colah.github.io/posts/2015-08-Understanding-LSTMs/

https://scikit-learn.org/stable/supervised_learning.html#supervised-learning

https://machinelearningmastery.com/dropout-regularization-deep-learning-models-keras/