Machine Learning Engineer Nanodegree

Capstone Proposal

Amol Pol
July 22nd, 2019

Proposal

Human Activity Recognition(HAR) Using Deep Learning

This project is to build a model that predicts the human activities such as Walking, Walking_Upstairs, Walking_Downstairs, Sitting, Standing or Laying.

Domain Background

Human activity recognition is the problem of classifying sequences of accelerometer data recorded into known well-defined movements. By automatically monitoring human activities, one can find many applications like security surveillance, healthcare, logistics support to location based services, wildlife observation, energy conservation etc.

It is a challenging problem given the large number of observations produced each second and lack of a clear way to relate accelerometer data to known movements.
One could devise a technique for HAR using classical machine learning. The performance(accuracy) of such methods largely depends on hand crafting features or good feature extraction methods. The difficulty is that this feature engineering requires deep expertise in the domain field.
Deep learning methods such as recurrent neural networks can also be tried on this challenging activity recognition tasks which would not require any feature engineering.

There have been many academic research on this problem of human activity recognition. I am highlighting a few of them over here –

- https://www.sciencedirect.com/science/article/pii/S1877050914008643
- https://www.hindawi.com/journals/js/2018/8580959/
- https://www.hindawi.com/journals/wcmc/2018/2618045/

Problem Statement

The Human Activity Recognition database was built from the recordings of 30 study participants performing activities of daily living (ADL) while carrying a waist-mounted smartphone with embedded inertial sensors.

So, in short the problem statement is a multi-class classification problem. We are given sensor readings of 30 study participants. These are accelerometer and gyroscope sensor readings. And our task is to predict the activity that the person is performing. These activities (class labels) are –

- Walking
- Walking_Upstairs

- Walking_Downstairs
- Sitting
- Standing
- Laying

The dataset contains 2 .csv files and raw time series of accelerometer and gyroscope sensor data with all the information necessary to make predictions.

1. Train.csv and test.csv contents:
    a. Columns 1 to column 561 are expert engineered features.
    b. The last column 564 is the activity name(output).
2. Train.csv contains 7352 data points.
3. Test.csv contains 2947 data points.

Data usage:

1. The above csv files will be used to make predictions using classical machine learning approach – Decision tree
2. Later half of the project, I will make use of accelerometer and gyroscope sensor data which will be directly fed to the RNN model(LSTM) to make predictions.

Datasets and Inputs

The experiments have been carried out with a group of 30 volunteers within an age bracket of 19-48 years. Each person performed six activities (WALKING, WALKING_UPSTAIRS, WALKING_DOWNSTAIRS, SITTING, STANDING, LAYING) wearing a smartphone (Samsung Galaxy S II) on the waist. Using its embedded accelerometer and gyroscope, we captured 3-axial linear acceleration and 3-axial angular velocity at a constant rate of 50Hz. The experiments have been video-recorded to label the data manually. The obtained dataset has been randomly partitioned into two sets, where 70% of the volunteers was selected for generating the training data and 30% the test data.

The sensor signals (accelerometer and gyroscope) were pre-processed by applying noise filters and then sampled in fixed-width sliding windows of 2.56 sec and 50% overlap (128 readings/window). The sensor acceleration signal, which has gravitational and body motion components, was separated using a Butterworth low-pass filter into body acceleration and gravity. The gravitational force is assumed to have only low frequency components, therefore a filter with 0.3 Hz cutoff frequency was used. From each window, a vector of features was obtained by calculating variables from the time and frequency domain.

Training data

```
In [16]: # get the data from txt files to pandas dataffame
X_train = pd.read_csv('UCI_HAR_dataset/train/X_train.txt', delim_whitespace=True, header=None, names=features)

# add subject column to the dataframe
X_train['subject'] = pd.read_csv('UCI_HAR_dataset/train/subject_train.txt', header=None, squeeze=True)

y_train = pd.read_csv('UCI_HAR_dataset/train/y_train.txt', names=['Activity'], squeeze=True)
y_train_labels = y_train.map({1: 'WALKING', 2:'WALKING_UPSTAIRS',3:'WALKING_DOWNSTAIRS',\
                              4:'SITTING', 5:'STANDING',6:'LAYING'})

# put all columns in a single dataframe
train = X_train
train['Activity'] = y_train
train['ActivityName'] = y_train_labels
train.head(3)

/Users/ampol/Virtualenvs/ai/lib/python3.6/site-packages/pandas/io/parsers.py:702: UserWarning: Duplicate names specified. This will raise an error in the future.
  return _read(filepath_or_buffer, kwds)
```

Out[16]:

	tBodyAcc-mean()-X	tBodyAcc-mean()-Y	tBodyAcc-mean()-Z	tBodyAcc-std()-X	tBodyAcc-std()-Y	tBodyAcc-std()-Z	tBodyAcc-mad()-X	tBodyAcc-mad()-Y	tBodyAcc-mad()-Z	tBodyAcc-max()-X	...	angle(tBodyAccMean,gravity)	angle
0	0.288585	-0.020294	-0.132905	-0.995279	-0.983111	-0.913526	-0.995112	-0.983185	-0.923527	-0.934724	...	-0.112754	
1	0.278419	-0.016411	-0.123520	-0.998245	-0.975300	-0.960322	-0.998807	-0.974914	-0.957686	-0.943068	...	0.053477	
2	0.279653	-0.019467	-0.113462	-0.995380	-0.967187	-0.978944	-0.996520	-0.963668	-0.977469	-0.938692	...	-0.118559	

3 rows × 564 columns

```
In [17]: train.shape
Out[17]: (7352, 564)
```

We have 7352 data points in our training data.

Test data

```
In [9]:
# get the data from txt files to pandas dataffame
X_test = pd.read_csv('UCI_HAR_dataset/test/X_test.txt', delim_whitespace=True, header=None, names=features)

# add subject column to the dataframe
X_test['subject'] = pd.read_csv('UCI_HAR_dataset/test/subject_test.txt', header=None, squeeze=True)

# get y labels from the txt file
y_test = pd.read_csv('UCI_HAR_dataset/test/y_test.txt', names=['Activity'], squeeze=True)
y_test_labels = y_test.map({1: 'WALKING', 2:'WALKING_UPSTAIRS',3:'WALKING_DOWNSTAIRS',\
                            4:'SITTING', 5:'STANDING',6:'LAYING'})


# put all columns in a single dataframe
test = X_test
test['Activity'] = y_test
test['ActivityName'] = y_test_labels
test.head()
```

Out[9]:

	tBodyAcc-mean()-X	tBodyAcc-mean()-Y	tBodyAcc-mean()-Z	tBodyAcc-std()-X	tBodyAcc-std()-Y	tBodyAcc-std()-Z	tBodyAcc-mad()-X	tBodyAcc-mad()-Y	tBodyAcc-mad()-Z	tBodyAcc-max()-X	...	angle(tBodyAccMean,gravity)	angle
0	0.257178	-0.023285	-0.014654	-0.938404	-0.920091	-0.667683	-0.952501	-0.925249	-0.674302	-0.894088	...	0.006462	
1	0.286027	-0.013163	-0.119083	-0.975415	-0.967458	-0.944958	-0.986799	-0.968401	-0.945823	-0.894088	...	-0.083495	
2	0.275485	-0.026050	-0.118152	-0.993819	-0.969926	-0.962748	-0.994403	-0.970735	-0.963483	-0.939260	...	-0.034956	
3	0.270298	-0.032614	-0.117520	-0.994743	-0.973268	-0.967091	-0.995274	-0.974471	-0.968897	-0.938610	...	-0.017067	
4	0.274833	-0.027848	-0.129527	-0.993852	-0.967445	-0.978295	-0.994111	-0.965953	-0.977346	-0.938610	...	-0.002223	

5 rows × 564 columns

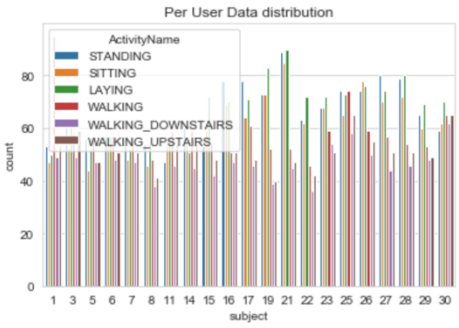
```
In [18]:
test.shape
```

Out[18]: (2947, 564)

We have 2947 data points in testing data.
Also, shown above few data points from test as well as training data using train.head and test.head as sample.

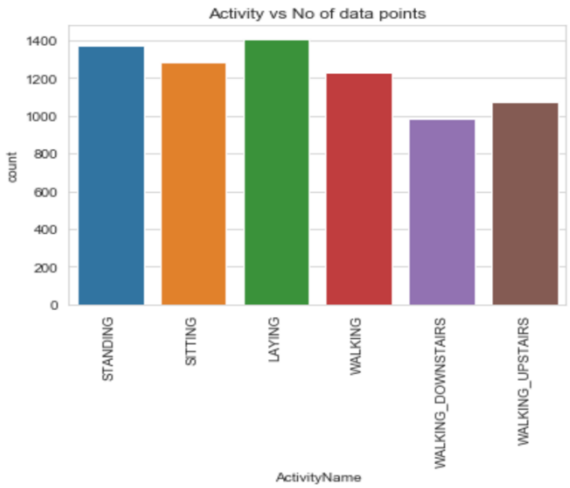
Check for data imbalance

```
In [14]:
sns.set_style('whitegrid')
plt.title('Per User Data distribution')
sns.countplot(x='subject',hue='ActivityName', data = train)
plt.show()
```



We have got almost same number of data points from all the subjects

```
In [15]:
plt.title('Activity vs No of data points')
sns.countplot(train.ActivityName)
plt.xticks(rotation=90)
plt.show()
```



Our data is almost well balanced across all class labels.
This dataset is provided by Kaggle and UCI Machine Learning Repository and can be obtained here.

Solution Statement

In this problem we have two kinds of data

- Expert engineered features (561 expert engineered features)
- Raw time series data (128 dimensional data).

This project is a trade-off between human engineered or expert engineered features and raw time series that is broken down into windows with overlaps. We will build models on expert engineered features as well as on raw time-series data.
When we build the models on raw time series, we will use deep learning models – Long Short Term Memory(LSTM). As we know, LSTM is a type of RNN and can handle time series data or any sequence data. When we have expert engineered features we will apply classical machine learning algorithms – Decision Tree.
We will see how different the Deep Learning performance would be as compared to the classical machine learning algorithm. Deep Learning can perform as well as the machine learning algorithm. This becomes very useful when we don't have domain knowledge and feature engineering isn't possible.
So, for the given problem we will build a classical machine learning model using decision trees and then try to build a deep learning model using LSTM. Later on we will try to optimize our LSTM and improve its accuracy.
High level steps are:

- Data Cleaning and pre-processing (applicable for Machine learning model - DT)
- Exploratory Data Analysis (applicable for Machine learning model - DT)
- Classical ML model – Decision Tree (applicable for Machine learning model - DT)
- Deep Learning model – LSTM (applicable for Deep learning model)

Benchmark Model

For this problem, the benchmark model will be a naïve predictor which will always predict the output class label as Walking_Downstairs. The reason I chose this was because no of data points for this class label were low across training and testing dataset.

So looking at following nos –

```
In [21]: print(train.ActivityName.value_counts())
         print(test.ActivityName.value_counts())

         LAYING                1407
         STANDING              1374
         SITTING               1286
         WALKING               1226
         WALKING_UPSTAIRS      1073
         WALKING_DOWNSTAIRS     986
         Name: ActivityName, dtype: int64
         LAYING                537
         STANDING              532
         WALKING               496
         SITTING               491
         WALKING_UPSTAIRS      471
         WALKING_DOWNSTAIRS    420
         Name: ActivityName, dtype: int64
```

The naïve predictor's accuracy will be 420/2947 = 14.25%. Hence, the benchmarking model will be this naïve predictor and my model has to perform better than this model.

Evaluation Metrics

The problem is a multi-class classification problem. Also, we have established earlier in the datasets and inputs section that the dataset is almost well balanced, I choose accuracy as my evaluation metrics.
Of course, I will also look at multi-class confusion matrix and we will also look at multi-class log-loss. The confusion matrix tells us what types of confusions are happening. For ex. Confusing between

sitting and standing up. So, confusion matrix is a very important way of understanding for which classes the model is doing well and for which classes is it getting confused. So, final conclusions will be made using the above metrics – accuracy, multi-class confusion matrix and multi-class log-loss with accuracy playing the major role.

Project Design

In accordance with the solution statement, the project design will have following components:

1. Data Cleaning and pre-processing: I will add checks for duplicates or null values in the given dataset. I will also test for data imbalance.
2. Exploratory Data Analysis: I will start with Univariate Analysis by drawing plots, box plots using one feature at a time.
   I will then try out Data visualization using t-SNE on the data using all the 561 expert engineered features.
3. Classical ML Model – Decision Tree
   Here I will build our decision tree using 561 expert engineered features.
4. Deep Learning Model – Long Short-term Model (LSTM)
   Here I wont use the hand crafted expert engineered features. I will use LSTM, a Deep Learning model which will automatically engineer features for me from raw timer series data. I will try out 2-3 different types of architecture. I will try out hyperparameter tuning like changing number of LSTM units, changing dropout rates or increasing number of LSTM layers.