#

Human Activity Recognition

Austin Koenig

Introduction

Data

The object of this project is to create a classifier which can differentiate between 14 different human activities from data collected via a triaxial accelerometer mounted to the subjects' wrists.

Following is the specifications of the device used to record the data:

Accelerometer Specifications

Property	Description
Type	tri-axial accelerometer
Measurement range	[- 1.5g; + 1.5g]
Sensitivity	6 bits per axis
Output data rate	32 Hz

| Location | attached to the right wrist of the user with:

x axis: pointing toward the hand

y axis: pointing toward the left

z axis: perpendicular to the plane of the hand

|

These data come from the UCI Machine Learning Repository.

Brief Discussion on Possible Issues and Solutions Consider the following table containing some statistics on the data.

Activity Name	# Sequences	% Sequences	Min Seq Length	Max Seq Length
Brush_teeth	12	0.014303	844	3199
Climb_stairs	102	0.121573	166	3199
Comb_hair	31	0.036949	166	3199
Descend_stairs	42	0.05006	156	3199
Drink_glass	100	0.11919	156	3199

Activity Name	# Sequences	% Sequences	Min Seq Length	Max Seq Length
Eat_meat	5	0.005959	156	9318
Eat_soup	3	0.003576	156	9318
Getup_bed	101	0.120381	156	9318
Liedown_bed	28	0.033373	156	9318
Pour_water	100	0.11919	156	9318
Sitdown_chair	100	0.11919	125	9318
Standup_chair	102	0.121573	125	9318
Use_telephone	13	0.015495	125	9318
Walk	100	0.11919	125	9318

Notice that some of the classes represent fewer than one percent of the entire dataset whereas others account for around ten percent. This is called class imbalance, which is certainly something to keep under consideration when creating models.

Furthermore, there are fewer than 1,000 samples in the entire dataset, although the samples are rather large in shape. In cases with so many features and so few samples, our performance is certainly hindered.

Proposed Solutions: 0. Do nothing 1. Test a variety of algorithms 2. Use a resampling technique 3. Generate synthetic samples 4. Slices observation sequences to create more samples 5. Omit underpopulated classes from study

The first option is what we'll call the null solution.

Solution 1 is the most obvious solution, and should be done in many cases to juxtapose model evaluations for the best results possible.

Solution 2 is more difficult to judge because of two contradicting factors: oversampling from a small class size may lead to overfitting, whereas undersampling from a large class size might induce a loss of signal that the model would otherwise learn. It would take more study to figure out how to optimally resample.

Solution 3 is similar to solution 2 for the same reasons described.

Solution 4 is also unknown. We may run into problems similar to those discussed with solutions 2 and 3. However, as most of the activities are cyclic in nature (that is, the activity appears similar over any two similarly-sized time intervals), we may be able to utilize this solution. For example, if we consider recording this data of a subject walking for 30 seconds, the subject might be moving between seconds 5 and 7 in a similar fashion as she might between seconds 22 and 24. In this sense, walking is cyclic: after two steps, it is the same movements over and over again. Eating is also cyclic, as is brushing teeth and descending stairs.

A subset of the samples are extraordinarily long, some containing close to 10,000 time steps, which translates to about five minutes recording at 32 Hz. It is easy to convince oneself that splitting an extremely long sequence into smaller

sequences to be used as individual observations might behoove our results without eliminating important long-term feature relationships. On the other hand, not all of our classes share this property of cyclicity: actions like pouring water, sitting down into a chair, and getting out of bed are not cyclic. Therefore, we might anticipate a shift in performance from non-cyclic activities to cyclic activities if we exercise this solution because cyclic activities will gain more samples.

Finally, solution 5 is the easy way out. It is not detrimental to the project since only half of our classes are underpopulated. It is, indeed, a viable solution.

Preprocessing

Model

This project uses a deep neural network with a one-dimensional convolutional base and a fully connected classifier top:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_1 (Conv1D)            (None, 1017, 256)         6400
_________________________________________________________________
conv1d_2 (Conv1D)            (None, 1010, 256)         524544
_________________________________________________________________
conv1d_3 (Conv1D)            (None, 1003, 256)         524544
_________________________________________________________________
max_pooling1d_1 (MaxPooling1 (None, 125, 256)          0
_________________________________________________________________
dropout_1 (Dropout)          (None, 125, 256)          0
_________________________________________________________________
conv1d_4 (Conv1D)            (None, 120, 256)          393472
_________________________________________________________________
conv1d_5 (Conv1D)            (None, 115, 256)          393472
_________________________________________________________________
conv1d_6 (Conv1D)            (None, 110, 256)          393472
_________________________________________________________________
max_pooling1d_2 (MaxPooling1 (None, 18, 256)           0
_________________________________________________________________
dropout_2 (Dropout)          (None, 18, 256)           0
_________________________________________________________________
conv1d_7 (Conv1D)            (None, 15, 256)           262400
_________________________________________________________________
```

```
conv1d_8 (Conv1D)            (None, 12, 256)           262400
_________________________________________________________________
conv1d_9 (Conv1D)            (None, 9, 256)            262400
_________________________________________________________________
max_pooling1d_3 (MaxPooling1 (None, 2, 256)            0
_________________________________________________________________
dropout_3 (Dropout)          (None, 2, 256)            0
_________________________________________________________________
flatten_1 (Flatten)          (None, 512)               0
_________________________________________________________________
dense_1 (Dense)              (None, 256)               131328
_________________________________________________________________
dense_2 (Dense)              (None, 256)               65792
_________________________________________________________________
dropout_4 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_3 (Dense)              (None, 14)                3598
=================================================================
Total params: 3,223,822
Trainable params: 3,223,822
Non-trainable params: 0
_________________________________________________________________
```

As it turns out, while both convolutional neural networks (CNN) and recurrent neural networks (RNN) work well on sequential data, they work well doing different tasks: RNNs are useful for regression whereas CNNs are useful for classification. Thus, this classification project utilized a CNN.

Metrics

This section describes the metrics used in this project. Let N_{tp}, N_{tn}, N_{fp}, and N_{fn} represent the numbers of true positives, true negatives, false positives, and false negatives, respectively. Finally, let N represent the total number of observations tested (i.e. $N = N_{tp} + N_{tn} + N_{fp} + N_{fn}$).

The metrics that will be examined are: - Accuracy - Precision - Recall - F1 Score - Confusion Matrix

Accuracy, A, is defined as follows:

$$A = \frac{N_{tp} + N_{tn}}{N}$$

Accuracy is a simple measure of the ratio of test observations for which the model made a correct positive classification to the total number of observations.

Precision, P, is defined as follows:

$$P = \frac{N_{tp}}{N_{tp} + N_{fp}}$$

Precision measures the ratio of correct positive classifications to the total number of positive classifications. It is different than accuracy because it can help differentiate between a naively-successful model and a truly-successful model. There's a frequently used example of predicting weather in a city where it is sunny 300 days out of the year. A naively-successful model might predict that everyday will be sunny, resulting in a 82% accuracy rate. While this rate is rather high, it is due to the nature of the data as opposed to the correctness of the model.

Recall, R, is defined as follows:

$$R = \frac{N_{tp}}{N_{tp} + N_{fn}}$$

The F1 Score, $F1$, is defined as follows:

$$F1 = 2\frac{PR}{P + R} = \frac{2N_{tp}}{2N_{tp} + N_{fp} + N_{fn}}$$

The F1 score is a sort of conglomerate value based on precision and recall.

A confusion matrix is essentially a table of predictions and actual values. It helps to articulate the different types of misclassifications.

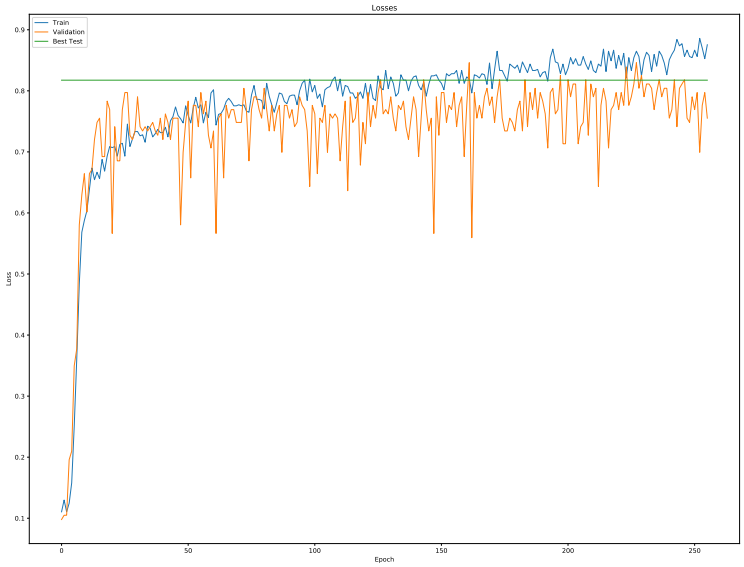
Results

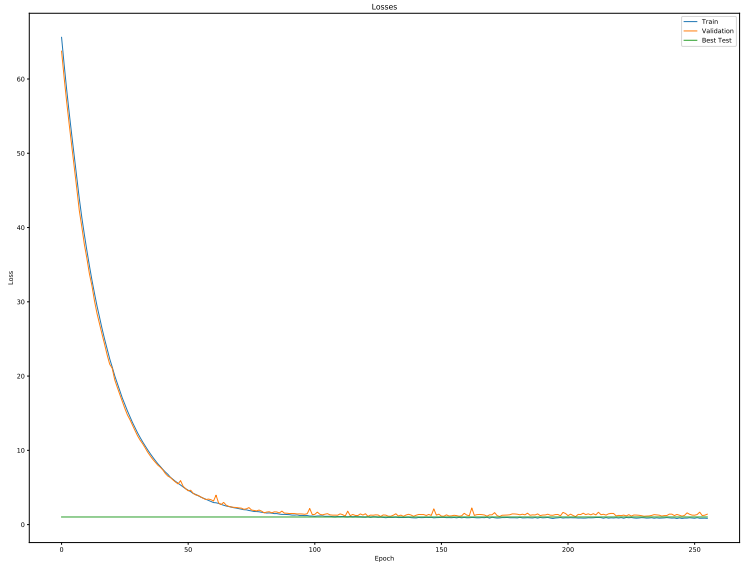
The usage for the codebase written for this project is as follows:

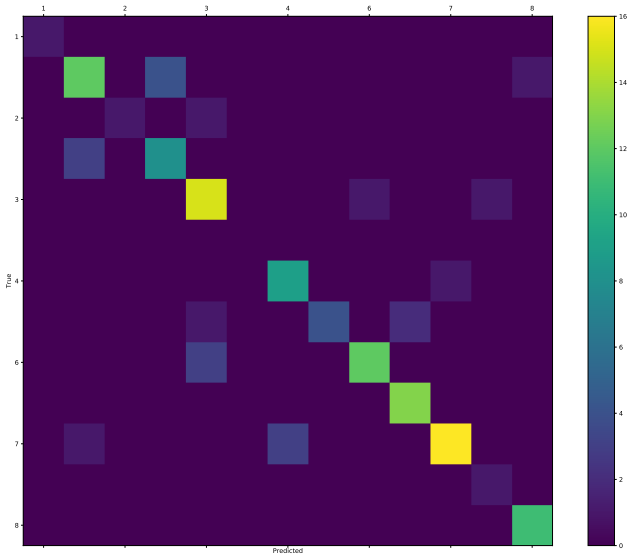
```
from activityrecognition import HARProject
proj.preprocess_data()
proj.generate_models()
proj.evaluate_models()
```

The model was trained on 256 epochs with a batch size of 16. Due to difficulties with the markdown output of images, please see the following files:

- Accuracy: `./output/acc.pdf`
- Loss: `./output/loss.pdf`
- Confustion Matrix: `./output/cm.pdf`


Losses
Train
Validation
Best Test
Loss
Epoch
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
50
100
150
200
250


Losses
Train
Validation
Best Test
Loss
Epoch
60
50
40
30
20
10
0
0
50
100
150
200
250



Conclusion

Overall, I am happy with the results of this project. I believe it would be a useful application of IoT technology in detecting the user's current activity, which is a precursor for a plethora of different applications.

Acknowledgments

These data come from this repository.