COSC522 - Assignment 3 Report

Cameron V. Adkins — Gabriel A. Abeyie — Purnachandra Anirudh Gajjala
November 17, 2022

1 Overview

For this project, we created a classical machine learning model that predicts movement type based on inputs from three standard sensors: an accelerometer, a gyroscope, and an orientation sensor. The movement classes were walking, climbing up, climbing down, and standing up; our additional activity was chosen to be riding a bus.

The collected samples were preprocessed and then put through the standard machine learning pipeline to create a classifier model using the RandomTreeClassifier. The produced model predicts movement type with 100% accuracy in cross-validation.

2 Data Collection

Data collection was performed over the course of two months leading up to the model's creation. We made use of the Sensor Logger application across both Apple and Android devices to record various examples of the required activities.

Of particular note (and difficulty) was the presence of discontinuities in the collected data. It seems that there is a bug in the android version of the application that causes the sensor to not collect data for extended periods of time (Fig. 1). We were able to overcome this by re-sampling the data as described in pre-processing below.

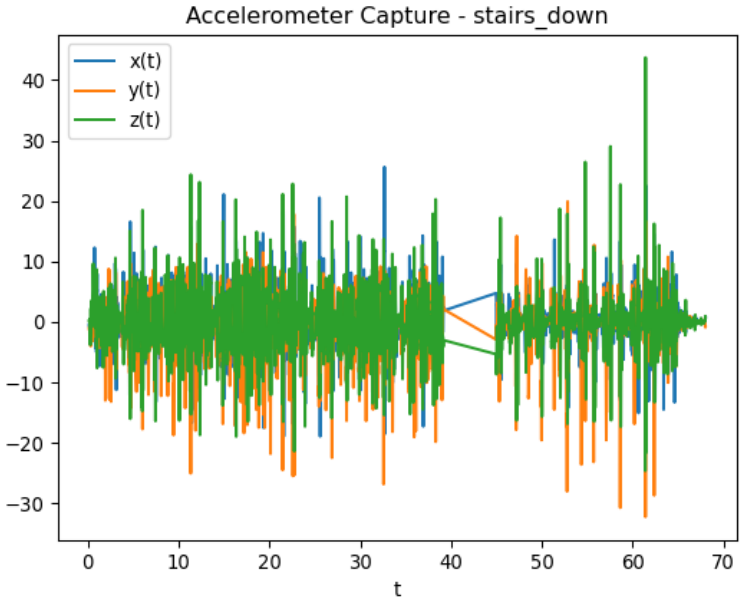

Figure 1: A capture of accelerometer data demonstrating the discontinuities.

Separately, while orientation data was available through the sensors we took care to ensure that all data was collected in the same orientation to make preprocessing simpler. This was done by positioning the phone in the left pocket with the screen facing outwards. Although this does not mirror real world usage, the produced model should still be valid since data can be reoriented before sending it to the model.

Lastly, it should be noted that not all samples are the same length. For instance, our extra activity (the bus ride) is one single sample but consists of around 40 minutes of data. Our preprocessing cuts these samples into 30 second windows and treats each one as a separate run for training the model.

3 Pre-processing and Feature Engineering

Before we train our model, we want to perform some preprocessing on the input data. First, we perform re-sampling to obtain a more constant sampling rate. This is done using the resample method from SciPy signal toolkit - this basically interpolates the signal to a 'continuous' signal and then reapplies the sampling. We obtain the initial sampling rate by taking the average of the deltas between samples in the first second of the recording. This also has the nice effect of removing the previously mentioned discontinuities since the interpolation will generate data for the missing periods.

Secondly we also apply a low-pass filter (LPF) to remove some higher frequency noise from the signal. We set the -3db point on the filter to 20 Hz. This removes a good deal of the interference from the sensor itself and allows for a more precise model.

Third, we also perform rather aggressive windowing. Our windows by default use 30 second intervals with half-overlap. The exception lies in the samples for standing which were recording in quick succession - as a result the window size for this is only 4 seconds. Each window is treated a separate sample for the model to train on.

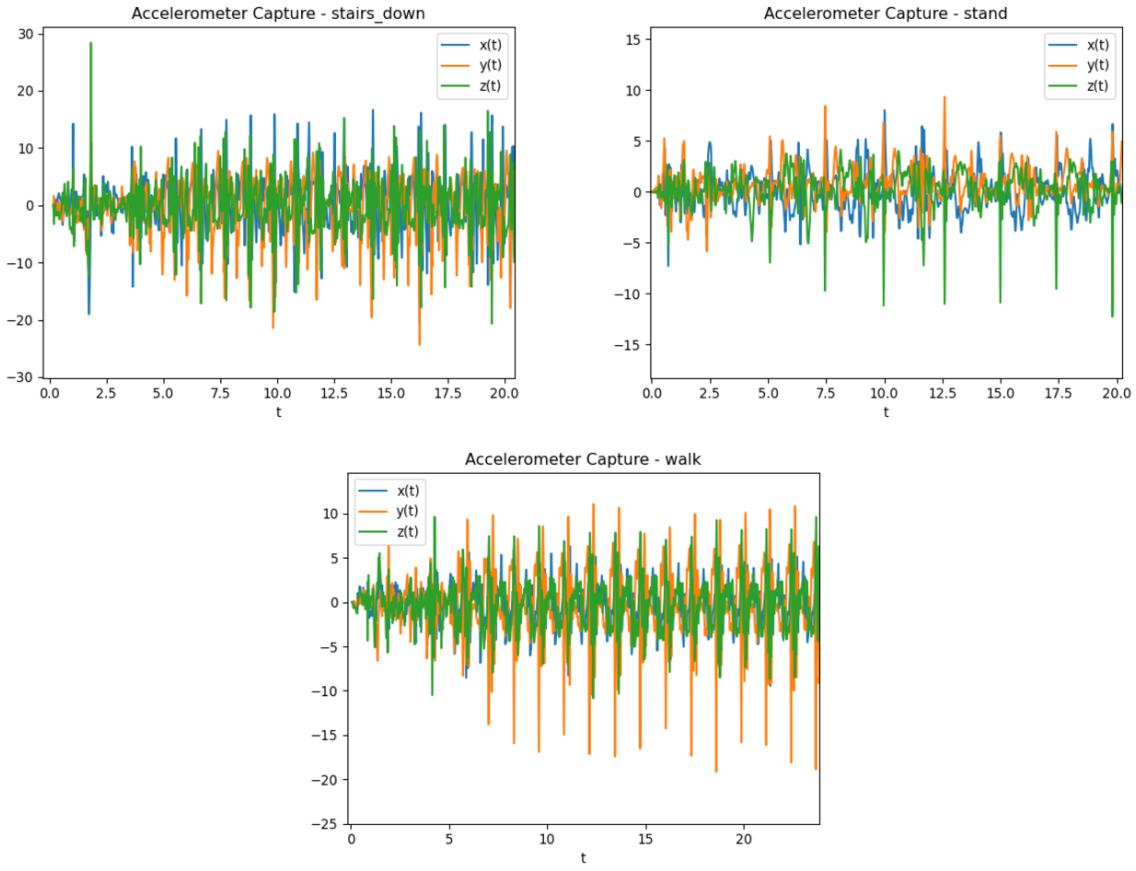

Figure 2: Acceleration data for various activities. Notice how the walking will encode to a high SnR where as the other two will encode to a lower SnR in different axes.

Somewhat surprisingly, feature engineering ended up being the least difficult aspect of the project. Due to the amount of preprocessing, our features ended up being simply the signal-to-noise ratio (SnR) of each of the input signals. More precisely, that is the x, y, and z axes on all inputs except orientation. Orientation uses quaternions and therefore has axes qx, qy, qz, and qw.

We believe the reason this performs so well is that SnR encapsulates a good deal of information about a signal. For instance, the walking data has a number of quick spikes with relatively high power which correlates to a high SnR. The stairs or standing up on the other hand produces a number of high frequency spikes with surrounding noise (Fig. 2). Additionally, passing the input signal through a LPF beforehand also reduces the range that our SnR can reside in, which helps the scaler after feature selection. All this combined with the differences in which axis corresponds to which activity makes for a fairly robust feature set to classify on.

Finally, before we create the model we also use the SMOTE method to balance the data. This single step brings our accuracy up from around 97% to 100%.

4 Results

The cross-validation results for our model were exceedingly good given the features provided to the model. As mentioned, we were able to achieve 100% cross-validation results.

```
Average Cross Validation Score from Training:
1.0


Confusion Matrix:
[[288   0   0   0   0   0]
 [  0 286   0   0   0   0]
 [  0   0 254   0   0   0]
 [  0   0   0 252   0   0]
 [  0   0   0   0 285   0]
 [  0   0   0   0   0 291]]


Missing classifications (if any): set()
Test Statistics:
              precision    recall  f1-score   support

         bus       1.00      1.00      1.00       288
         sit       1.00      1.00      1.00       286
 stairs_down       1.00      1.00      1.00       254
   stairs_up       1.00      1.00      1.00       252
       stand       1.00      1.00      1.00       285
        walk       1.00      1.00      1.00       291

    accuracy                           1.00      1656
   macro avg       1.00      1.00      1.00      1656
weighted avg       1.00      1.00      1.00      1656



Testing Accuracy: 1.0
```

The notebook and collected data is attached to this submission, please let us know if you have any questions.