# Team 2 - Aaron Kaplowitz, Todd Morse, Dan Snyder

## Assignment 2 Part 4 Writeup

### Question 1

The data collection process consists of 4 key parts

1. The user selects an activity from a predefined list
2. The user performs said activity and the app sends it to the data collection server
3. We retrieve the labeled data from the server and run the classifier training script on it
4. We run (and leave running) the activity recognition script while the user performs any activity, and we see if it's right!

Let's provide some more concrete detail about each step:

1. In the Android app, there's a dropdown of 5 options: [None], Sit, Walk, Run, and Jump. The user (who essentially is a research subject, and was Aaron Kaplowitz), selects one of the 5 options. They then begin data collection by hitting the toggle onscreen.

2. After the user has started the Accelerometer Service, they then perform the activity that they selected above. Aaron did each of these in order for 5 minutes and switched the label to the new activity when switching. While he did each activity, the app sends data to the collection server. The phone was in a jeans pocket which

each activity was performed, with the stop of the phone facing up and the display facing out. Each parcel of data contains a label that specifies which activity was being performed. After each activity has some data stored on the server, we can train the classifier.

3. We run the classifier training script. It learns about the types of data points that correspond to each activity (i.e. the phone being stationary with the display facing up corresponds to sitting).

4. Now that it knows what signifies a specific activity, it can be run on unlabeled data. When doing this, we expect that it will remember the trends it found during training and will be able to pick activities out of the unlabeled data.

One thing that could potentially make a difference is the orientation in which the phone is held when recording labeled data vs when trying to identify unlabeled data. We perform magnitude calculations over each window in the Python scripts so that the phone can be held either upside-down or rightside-up and the data will still be valid. Additionally, there's some normalization provided for us in the Android app that helps with this. What will likely cause problems is when you rotate the phone 90 degrees and attempt to detect activities - now the axes are completely different as opposed to flipped.

## Question 2

Our classification algorithm uses a decision tree classifier over a 10-fold cross-verification. Our parameters (technically hyperparameters) are a maximum depth of 3 and a maximum features of 5. Our other algorithm was a logistic regression with no explicit parameters, but we initialized a 10-fold cross validation with no shuffle, and no random state. We ended up going with the former of the two because it had a higher accuracy as

well as a higher precision and recall.

## Question 3

It's pretty good!

Here's the accuracy, precision, and recall for our collected training data:

```
average accuracy:
0.854342544792

average precision for each activity:
[   sitting,    walking,    jumping,    running ]
[ 0.83479864  0.94505756  0.48098291  0.74430278]

average recall for each activity:
[   sitting,    walking,    jumping,    running ]
[ 0.94968848  0.80273551  0.77967278  0.79901718]
```

This is slightly worse than when run on the sample data, but that's attributable to there being more sample datapoints than observed ones.

Here's the same values for the provided sample data:

```
average accuracy:
0.94165621079
average precision:
[ 0.92993968  0.94734665 ]
average recall:
[ 0.89305838  0.96627251 ]
```

The sample data also has half as many labels, so our classifier should have a much higher random success rate (1/2 vs 1/4), so this likely also contributes to the sample data's higher numbers.

Empirically, our classifier trained on our data works really well! When running activity recognition, it takes a few datapoints to get itself going, and subsequently is pretty much spot-on. It occasionally has trouble differentiating running from walking, but it's correct the vast majority of the time. I'd say that the activity/precision/recall matches up to the empirical recognition data.

## Contributions and Misc

For part 4, Aaron and Dan worked on the Android code while collaborating with Todd on the Python. Aaron created the training data and did the real-world testing.

This project was **HARD**.