Assignment 3: Activity Recognition (15P)

Goals

You are familiar with the basic principles of machine learning. You can implement a simple machine learning classifier to detect activities based on real-time sensor data.

1 Gathering Training Data 5P

Recommended Packages: *DIPPID, pandas*

Create a program called *gather_data.py* which captures sensor data (accelerometer and gyroscope) from the DIPPID device and saves it to a CSV file with the following structure:

File Name: `your_name-activity-number.csv` (e.g., *susi-running-1.csv*)
Columns: `id, timestamp, acc_x, acc_y, acc_z, gyro_x, gyro_y, gyro_z`

This sensor data will be used to train a machine learning classifier later. It should be possible to use the DIPPID device to start capturing data and stop capturing when a fixed time has passed or a certain amount of data has been gathered.

Record at least five data sets for each of the following activities: `running`, `rowing`, `lifting`, `jumpingjacks`. A depitcion of those activities can be seen in Figure 1. Resample your data to 100 Hz to save disk space. In addition to including the data in your own repository, upload it to a repository shared with others (see GRIPS) so all of you have a larger data set.

Uploading your data is due to May 19 to ensure that all of you have access to the same body of train data in time.

Score

(2P) data is logged correctly

(1P) log files are named and structured appropriately

(1P) logging can be started with the DIPPID device

(1P) enough data sets captured

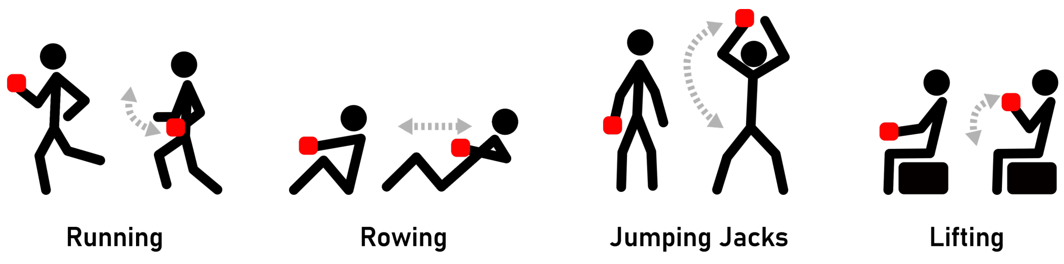

Figure 1: Four different fitness activities.

2 Activity Recognition 10P

Recommended Packages: *pandas, numpy, pyglet, sklearn*

Create a program called *fitness_trainer.py*. The application should help users with their fitness training by displaying activities and tracking whether they were executed correctly for a certain amount of time. When the program is started, it should read training data from CSV files, split them into a train an test data set, and train a machine learning classifier. Most certainly, it is required to pre-process training data before training (e.g. filtering, normalization, transformation into frequency domain, ...).

After the training is finished, evaluate the model's accuracy using the test data set. The program should then predict activities based on sensor data from the DIPPID device. The prediction should run continuously without requiring further intervention by the user. Visualize the fitness trainer nicely using pyglet.

Score

(1P) the program loads training data correctly

(2P) training data is pre-processed appropriately

(1P) a classifier is trained with this training data when the program is started

(3P) the classifier recognizes activities correctly

(1P) prediction accuracy for a test data set is printed

(1P) prediction works continuously without requiring intervention by the user

(1P) the fitness training application works and looks nice

Code Quality: bad code quality and missing documentation can lead to loss of points.