ECE5242 Project 2: Gesture Recognition

Code Due Date: **2/28/2018 at 1:40pm** on CMSX, <NetID>_project2.zip
Report Due Date: **3/5/2018 at 11:59pm** on CMSX, <NetID>_project2.pdf

In this project, you will use IMU sensor readings from gyroscopes and accelerometers to train a set of Hidden Markov Models and recognize different arm motion gestures. You should then be able to classify unknown arm motions in real-time using your algorithm.

Training Data Download: Now available at https://upenn.box.com/v/ECE5242Proj2-train
Test Data Release: 2/28/2019 1:40pm, https://upenn.box.com/v/ECE5242Proj2-test
The data sets contain the raw IMU readings collected from a consumer mobile device and corresponding labels that describe the arm motions associated with the movements. You can find the coordinate system used here:
http://developer.android.com/reference/android/hardware/SensorEvent.html
*Data format for each column (7 columns in total) in the IMU data is ts , Ax, Ay, Az, Wx, Wy, Wz (Time (millisecond), 3x Accelerometer (m/s2), 3x Gyroscope (rad/sec))

Upload: on CMSX
(1) **Code** (due 2/28/2019 1:40pm, <NetID>_project2.zip) : Do not include training/test data.
(2) **Write-up** (due 3/5/2019 11:59pm ,<NetID>_project2.pdf) : Write a project report including the following sections: Introduction, Problem Formulation, Technical Approach, Results and Discussion. Make sure your results include proper visualization (e.g. log likelihood) of your gesture classification.
*Clearly presenting your approach in the form of report and presentation and having good algorithm performance are equally important.

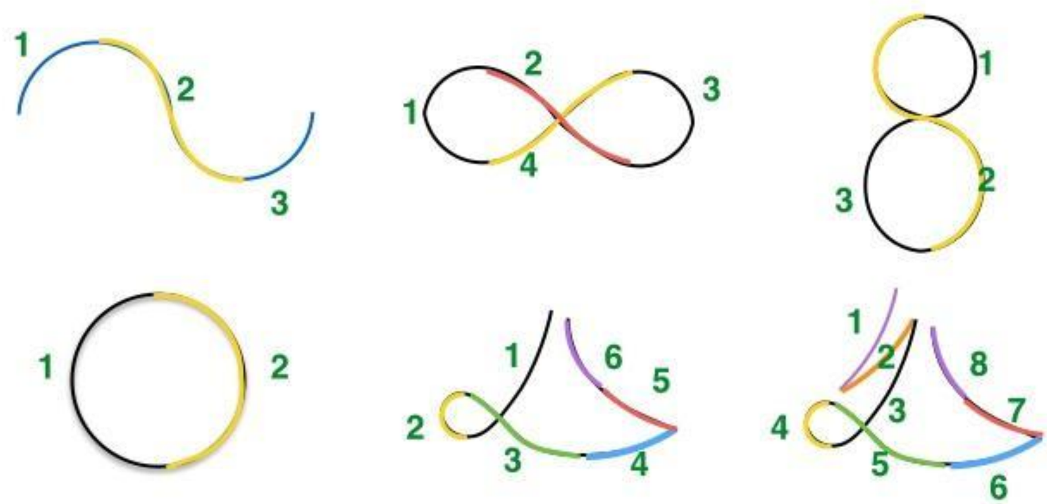
1. GETTING INTUITION. Experiment with filtering and quantizing the raw sensor information. Given your filtered observations, you can then train HMMs for each motion gesture class using the Expectation-Maximization procedure.

2. IMPLEMENT HMM. You will use HMM models to describe the corresponding motion gestures. You should initially use a simple left-to-right HMM's in the dynamical models. Write down the corresponding model parameters that will need to be learned from the training data. Use the Baum-Welch algorithm to learn the model parameters. **Read the Rabiner HMM Tutorial posted on CMSX or Blackboard.**

3. TESTING. You should then make sure that your program can take input sensor readings from unknown gestures. You can then compute the log likelihood under the different HMM models, and then show how certain you can classify the unknown motion as a gesture. **We will release test data on the due date of the code. Please include the results of your classification for test set in the report, accuracy of the test set predictions will be worth 50% of the points of the coding portion of the project (which is half the total points including the report).**

Tips:

1) The data contains six different motions: Wave, Infinity, Eight, Circle, Beat3, Beat4. The names indicate their corresponding motions. (Disregard the numbers by the motions).



2) Experiment with filtering and discretization the raw data. This will give you an idea how to use the data or what features to extract. Feature extraction, as you've learned in Project 1, is a very important part of the learning process. One way to get discrete states from your features is to run K-means clustering, and each new state will be assigned to its closest cluster.

3) Choose the number of hidden states (N) and observation classes (M) with cross validation.

4) Try different initialization strategies.

5) Like in Project 1, you can set a maximum number of iterations or use a threshold to detect convergence of the learning algorithm.

6) Be aware about underflow issues. You can either use do your calculations in logspace or use the scaling procedure described by Rabiner in Section V of the paper.

7) You will learn a model for each gesture of the probability of that gesture given your sequence of observations. The prediction will simply by the argmax of all the models (the model that assigns the highest probability to the sequence of observations).