ESE-650: LEARNING IN ROBOTICS

Gesture Recognition using Hidden Markov Models

Ankit Vora

Aim:

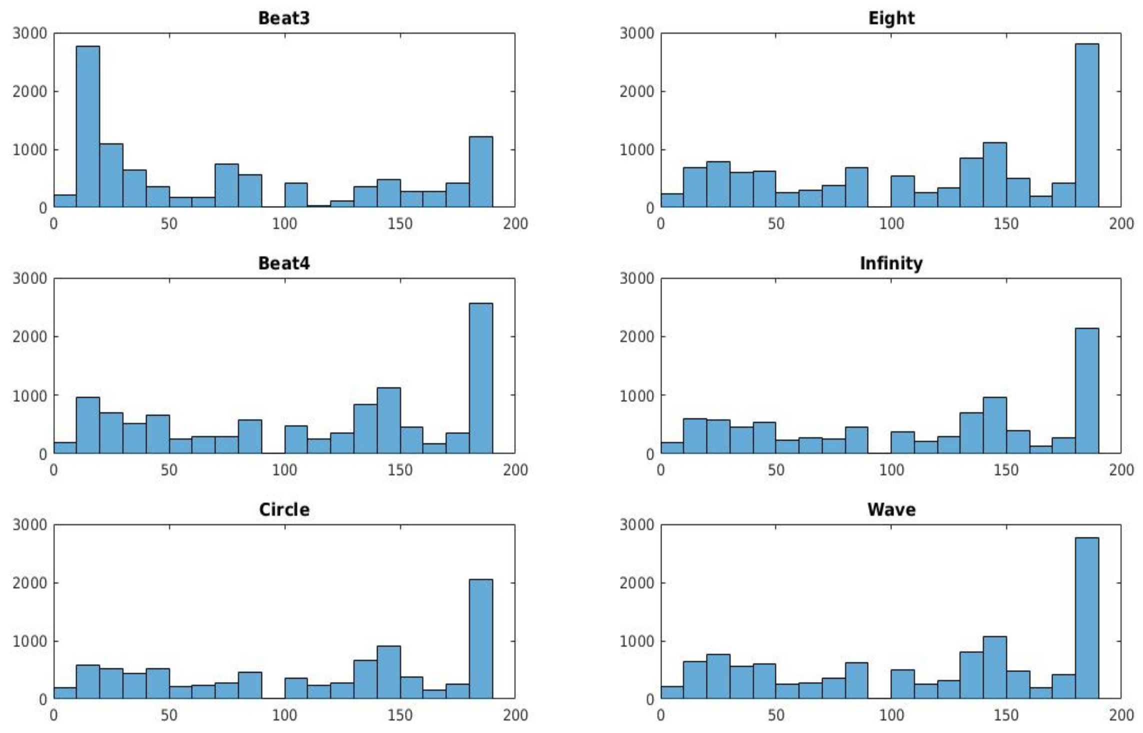
To classify gestures made using a cellphone given the onboard IMU data. This has to be achieved by using Hidden Markov Models (HMM as it's famously known). The gestures are categorized as 'Beat3', 'Beat', 'Circle', 'Eight', 'Infinity' and 'Wave'.

Approach

Design

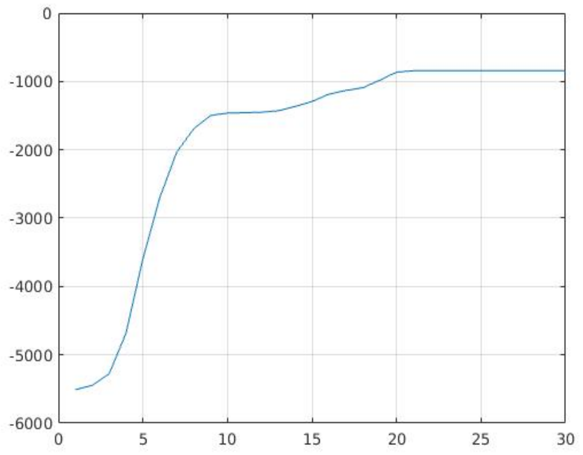
1. **Clustering**

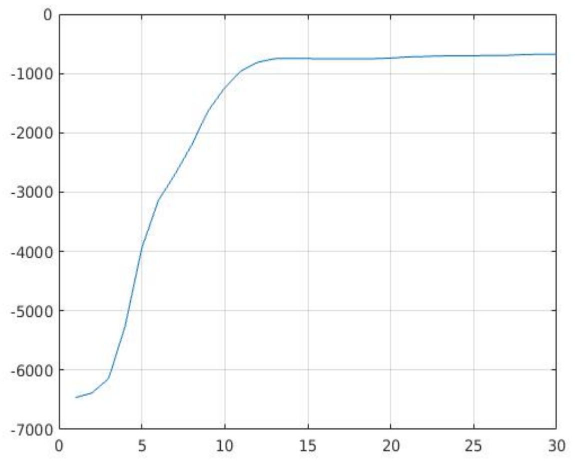
- The first step towards designing the filter is to collect the entire dataset and cluster them into M clusters. The 'M' is an important design parameter and I've explained how to choose it in the later section. The clustering is done using the MATLAB Kmeans algorithm. Post clustering I check my cluster assignments to see which clusters are used and which ones are untouched. Also I visualize the clusters used for each different class. Here's the visualization:



2. **Forward Backward Algorithm**

- The approach I followed was very similar to what was mentioned in the paper[1]. In addition to that, I also followed the tutorial[2] which explains the calculations for alpha beta with example. This helped in understanding exactly how the probabilities change with time.
- My algorithm runs for a fixed number of iterations = 30. This is because the EM converges much before 30 iterations. Given below are the graphs of log P(O|λ) for each iteration. It shows how the quantity exponentially increases with time and stabilizes after few iterations.





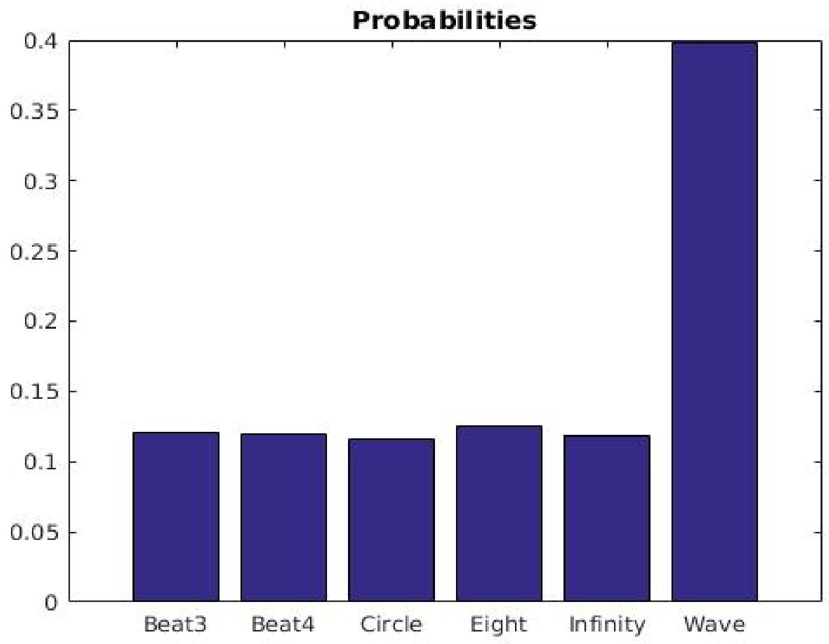
- Initialization: I tried a bunch of different initialization for the A & B matrices. I began with left-right HMM model with diagonal and the next element. In addition to that I also tried with equal probabilities in state transition matrix. But either of these didn't work. Hence I went ahead with 'random' initialization.
- Choosing N and M: N ≡ Number of states and M ≡ Number of observation. These two quantities are two of the most important parameters in HMM. I used the cross validation method to design these parameters. The cross validation set consisted of permuted combinations of N_{states} ϵ [10 12 14 16 18 20] & N_{obs} ϵ [20 30 40...190 200]
  All these 114 combinations are crossvalidated and I choose the best combination based on the difference between the top 2 probabilities. Finally I chose the following parameters:

  $N_{states} = 12$ & $N_{obs} = 200$

3. **Performance**

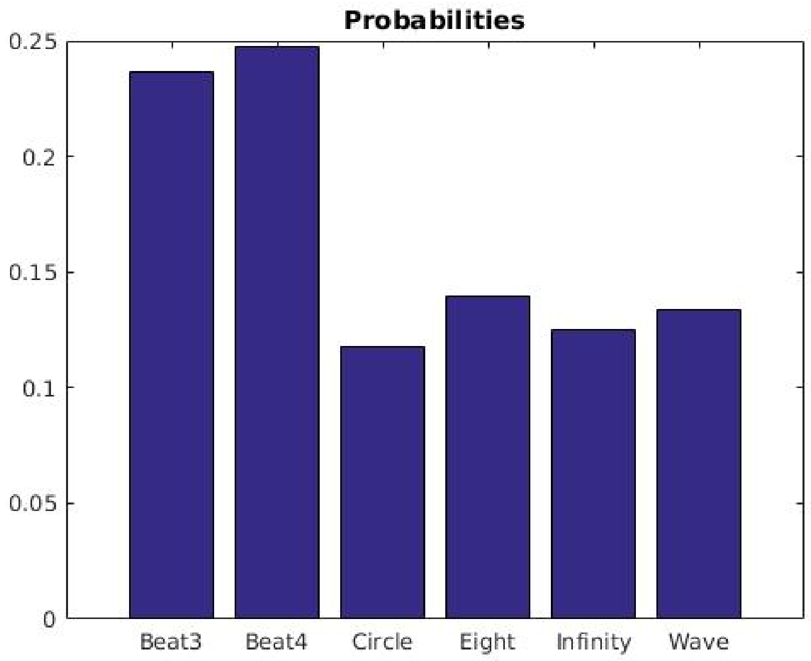
- The performance on the training dataset was very good. I got all correct on the held out as well as training datasets. The performance on the testing dataset is shown below:

Dataset:1



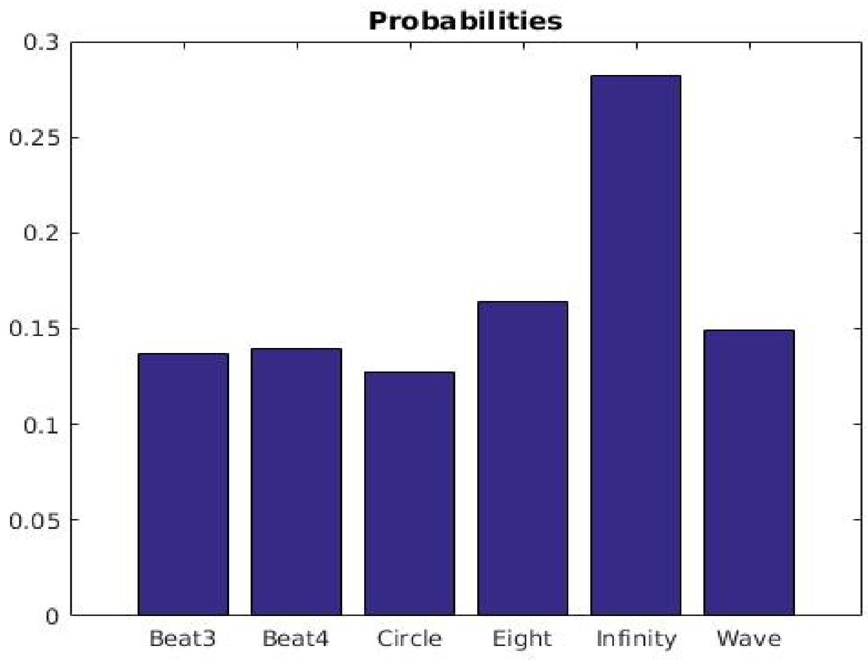
Predicted: **Wave**

Dataset:2



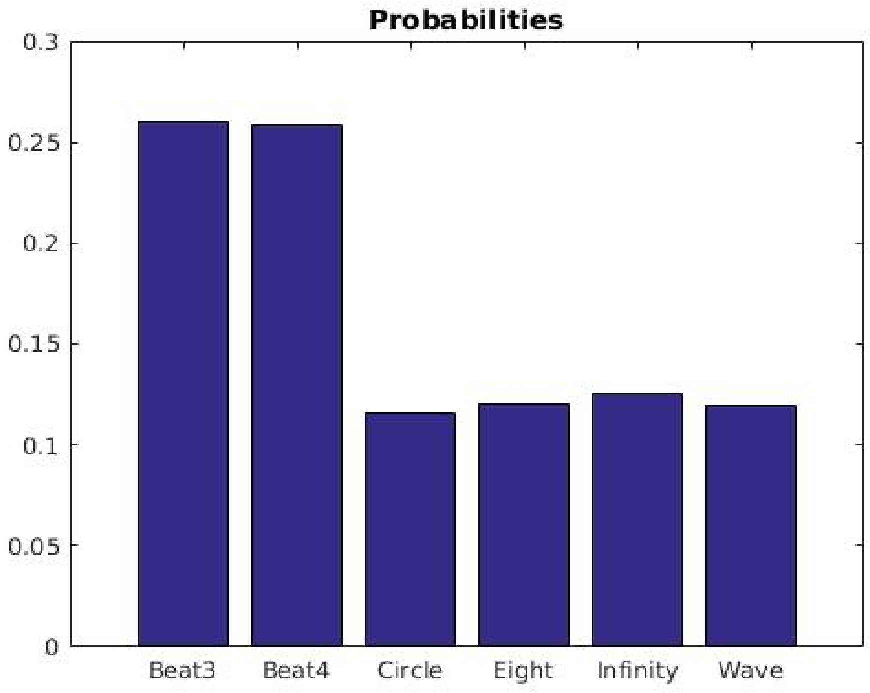
Predicted: **Beat4**

Dataset:3



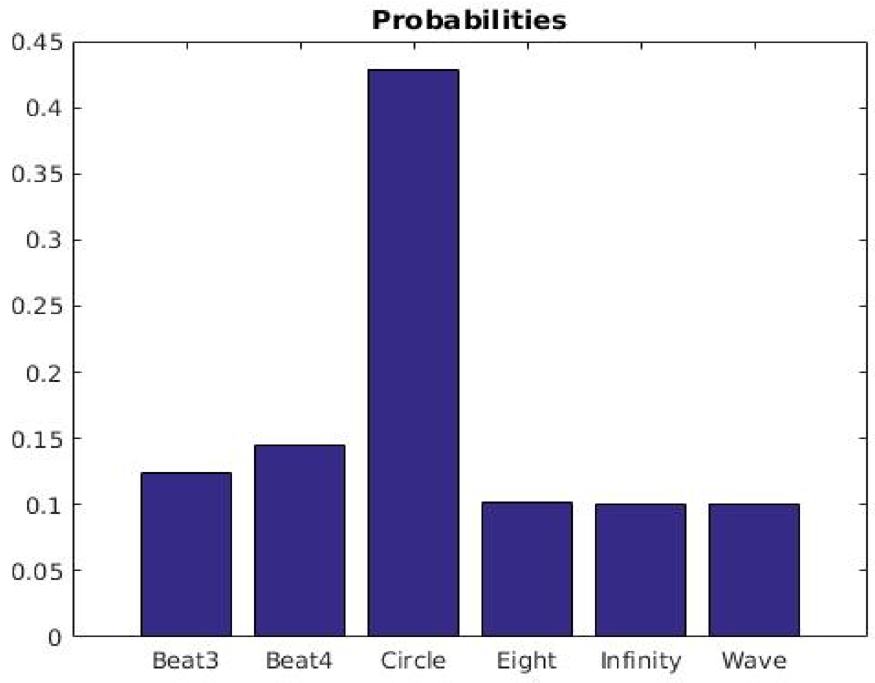
Predicted: **Infinity**

Dataset:4



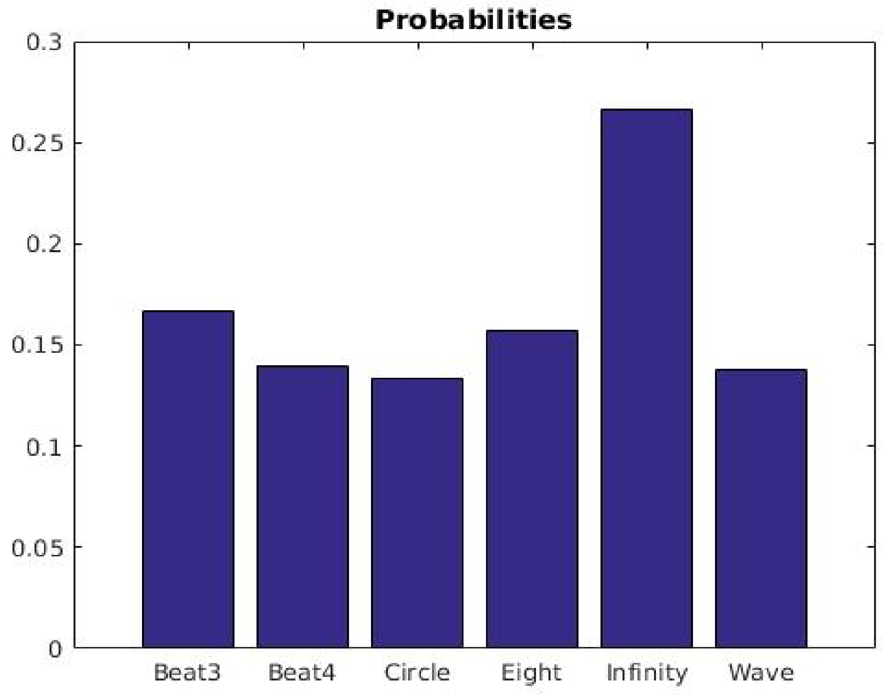
Predicted: **Beat3**

Dataset:5



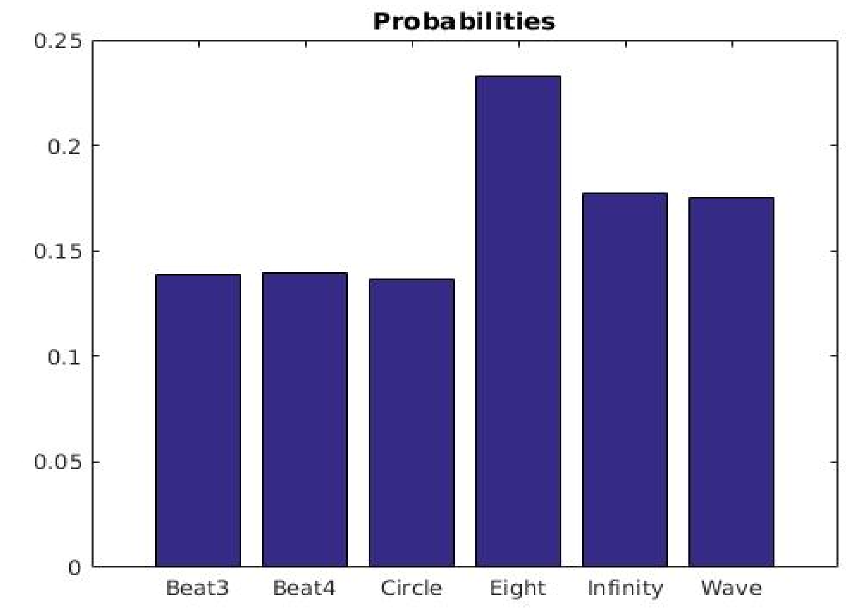
Predicted: **Circle**

Dataset:6



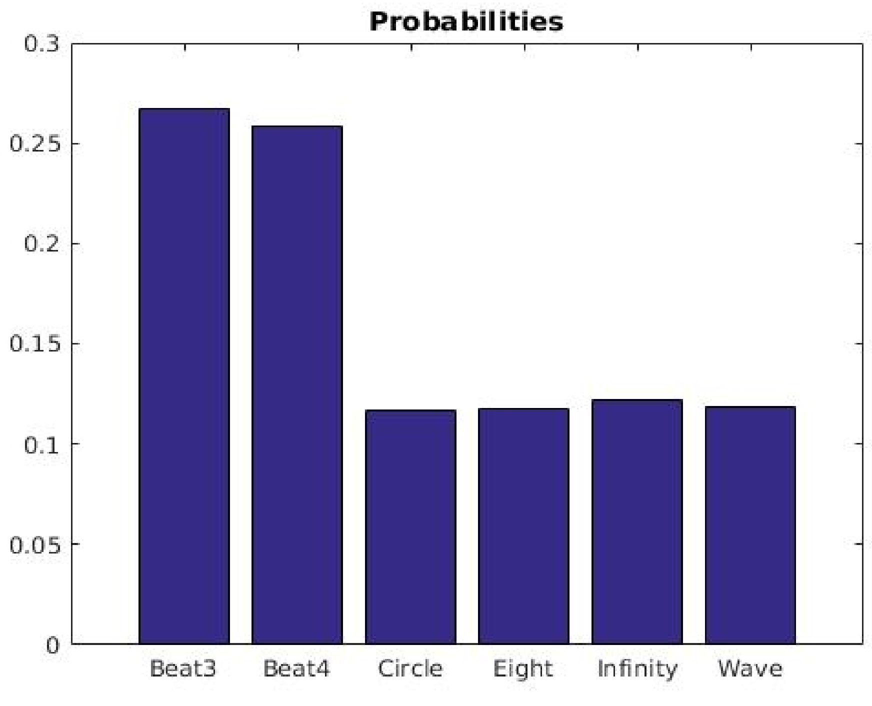
Predicted: **Infinity**

Dataset:7



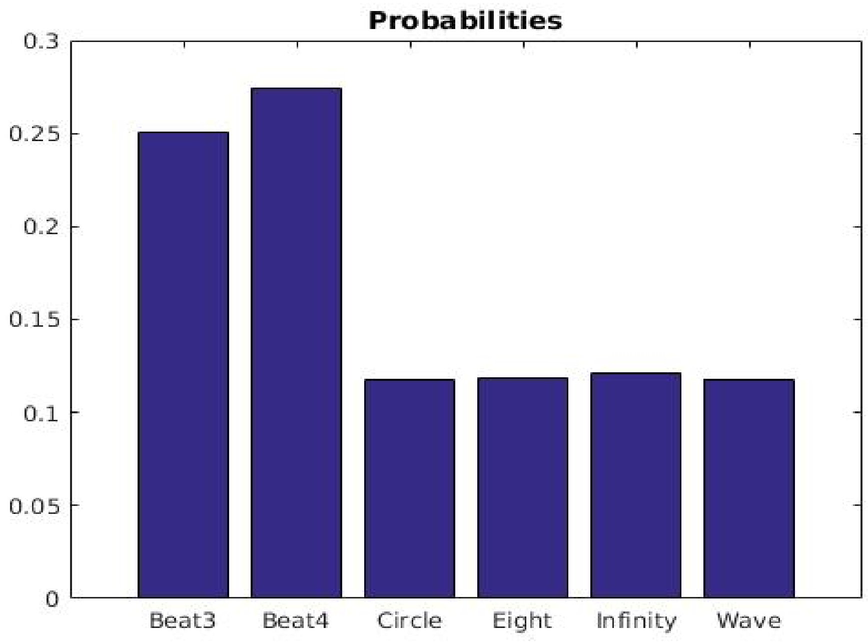
Predicted: **Eight**

Dataset:8



Predicted: **Beat3**

- As one can see above, the last prediction was wrongly classified. Hence I modified the number of observation to 190 and retrained the model to get the correct predictions. The new plot of the last dataset is shown below:



```
Predicted: Beat4
```

Observations and Improvements

- Initialization of the parameter matrix is important. Since I have a random initialization, the training and hence the output varies. One improvement could be to come up with a good way of initializing these parameters.
- Another important aspect is the lack of data. Hence it is very difficult to check the robustness of the models if the amount of data is less.

Running the code

- There's a 'testMain.m' file in the folder that needs to be executed. In the file, you can observe that the location of test files is given as './test/'. **This means the test files should be in the 'test' folder . Hence copy the '.txt' test files in test folder. Please note that there shouldn't be any other files other than the test files in .txt format.**
- Once executed, the code will output the graphs one by one. There's a 'pause' which stops the code after each execution and allows you to see the results. To continue, **press any key**.

References

1. Lawrence Rabiner , 'A tutorial on Hidden Markov Models and Selected Applications in Speech Recognition.
2. James Allen, Fall 2003 - CSC - 248/448: Speech Recognition and Statistical Language Models