James Donnelly
Jfd235
ECE5242
9 March 2023

Project 2 Report

Introduction

This report outlines the problem, processes, and results surrounding the second project of ECE5242. This project served as an introduction to Hidden Markov Models through the form of a gesture recognition system. I am leaving this project with what I believe to be a newfound and solid understanding of this topic, which I hope to make evident throughout the report.

Problem Statement

The basis of the problem is to design an algorithm that will accurately identify certain gestures given a set of IMU (Inertial Measurement Unit) data. Provided was a set of data from eight distinct gestures to train and test our models on.

Approach Description

To solve this problem, we would be implementing a Hidden Markov Model trained on the given datasets. As the HMU data was in seven dimensions between time, acceleration, and angular data, the first step was to quantize the data vectors. A simple way of vector quantization was done with KMeans clustering. All of the test data (minus a holdout set) was concatenated, and clusters were created from this data. A range of clusters were used for training, but in the end, it was decided that 50 clusters produced the most optimal results in the quickest time. These clusters would act as observations for the HMM. A distinct HMM would be created and trained for each gesture. To train the model, the specific gesture data was concatenated and fit to the observations created above to quantize the data. These quantized values were then used to train an HMM for each gesture.

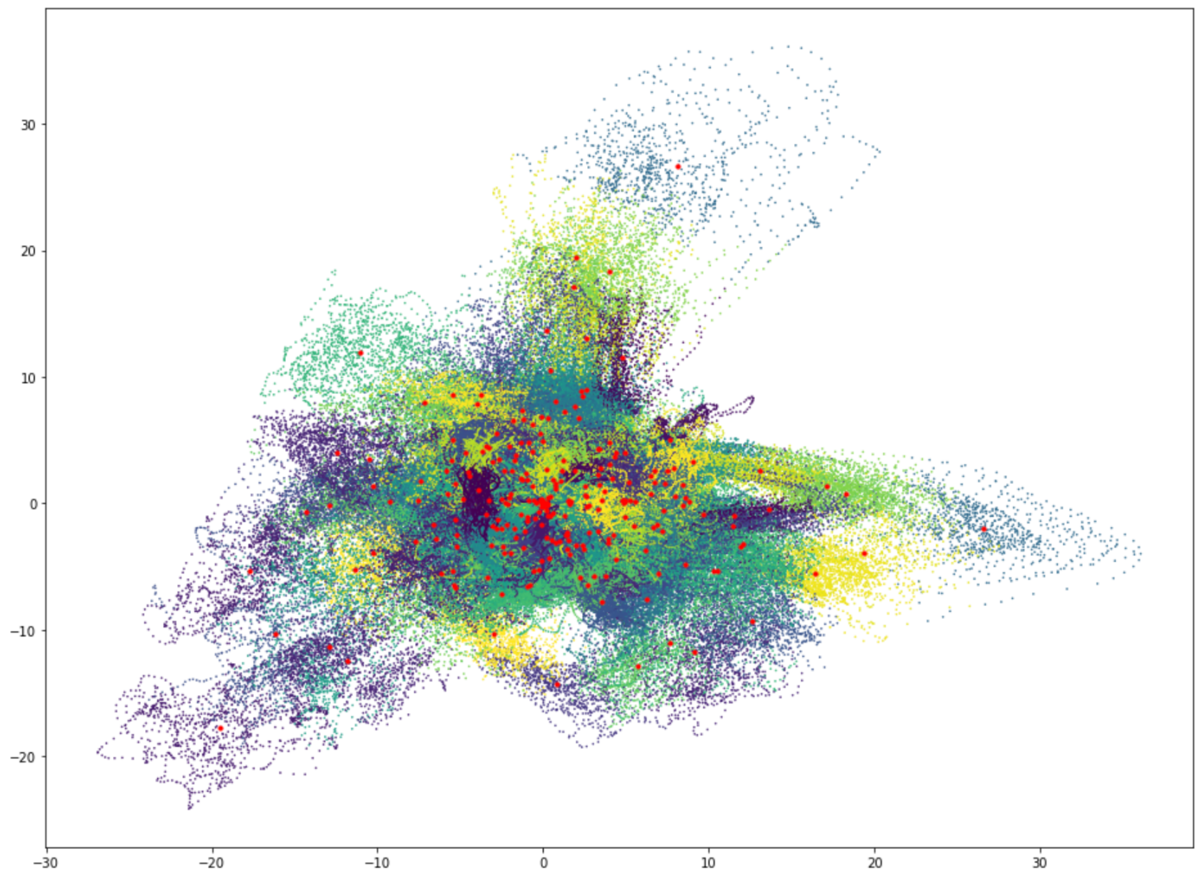

Figure 1: Visualization of IMU Data and Clusters

The Baum-Welch algorithm was used to create the HMMs. This involved creating forward/backward recursion functions, as well as an E and M step. The forward and backward functions estimate the probabilities of seeing a certain observation pattern up to the current state and model, as from the current state and model to the end, respectively. The E and M steps use this data to update the transition and emission matrices to better fit the data based on the forward/backward passes. The algorithm was followed exactly as described in the course slides, as shown below:

Data: $\mathcal{O}_{1:T}$
Definitions: K: num. possible states, M: num. possible measurements
Initialize: A, B, π

for $l = 1, \ldots, l_{max}$ **do**
  Forward-Backward calculations:
    $\alpha_1(i) \leftarrow \pi_i$
    for $1 < t \leq T,\ 1 \leq i \leq K$ **do**

$$\alpha_t(i) \leftarrow \sum_{j=1}^{K} \alpha_{t-1}(j) A_{ij} B_i(\mathcal{O}_t)$$

    end
    $\beta_T(i) \leftarrow 1$
    for $T > t \geq 1,\ 1 \leq i \leq K$ **do**

$$\beta_t(i) \leftarrow \sum_{j=1}^{K} A_{ji} B_j(\mathcal{O}_{t+1}) \beta_{t+1}(j)$$

    end
  E-Step:
    for $1 \leq i \leq K, 1 \leq t \leq T$ **do**

$$\gamma_t(i) \leftarrow \frac{\alpha_t(i)\beta_t(i)}{\sum_{j=1}^{k} \alpha_t(j)\beta_t(j)}$$

    end
    for $1 \leq i \leq K, 1 \leq j \leq K, 1 \leq t < T$ **do**

$$\xi_t(i,j) \leftarrow \frac{\alpha_t(i) A_{ji} B_j(\mathcal{O}_{t+1}) \beta_{t+1}(j)}{\sum_{i=1}^{K} \sum_{j=1}^{k} \alpha_t(i) A_{ji} B_j(\mathcal{O}_{t+1}) \beta_{t+1}(j)}$$

    end
  M-Step:
    for $1 \leq i \leq K, 1 \leq j \leq K$ **do**

$$A_{ji} \leftarrow \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \sum_{i=1}^{K} \xi_t(i,j)}$$

    end
    for $1 \leq i \leq K, 1 \leq o \leq M$ **do**

$$B_i(o) \leftarrow \frac{\sum_{\{t \text{ s.t. } \mathcal{O}_t = o\}} \gamma_t(i)}{\sum_{t=1}^{T} \gamma_t(i)}$$

    end
end

Figure 2: Baum-Welch Algorithm

Once this algorithm was implemented with Python, I checked basic toy datasets using the premade HMMLearn Python library to verify my outputs, checking that the outputs matched exactly. The model worked for smaller inputs, but quickly fell apart using larger datasets due to underflow/overflow. To remedy this, scaling was implemented in the forward/backward portions of the algorithm. These scale factors cancel out in the E and M steps, and were ultimately used to calculate the log likelihood, as described later. Once the scale factors were implemented, the model was able to handle larger datasets, and again matched the output of HMMLearn.

Finally, before the algorithm could be used to train and test the data, one more thing had to be taken care of. Namely, each iteration of the algorithm, values of 0 in the emission matrix had to be scaled up to a small number ($1*10^{-8}$). This is because otherwise the model may prematurely force values to zero, as opposed to just making them unlikely.

Finally, with the algorithm in place, the model could be trained, and the test data could be classified. The log likelihood for each dataset was calculated by taking the negative of the sum of the log of the scale factors after passing the data through the forward recursion of the Baum-Welch algorithm. The results of this are shown below:

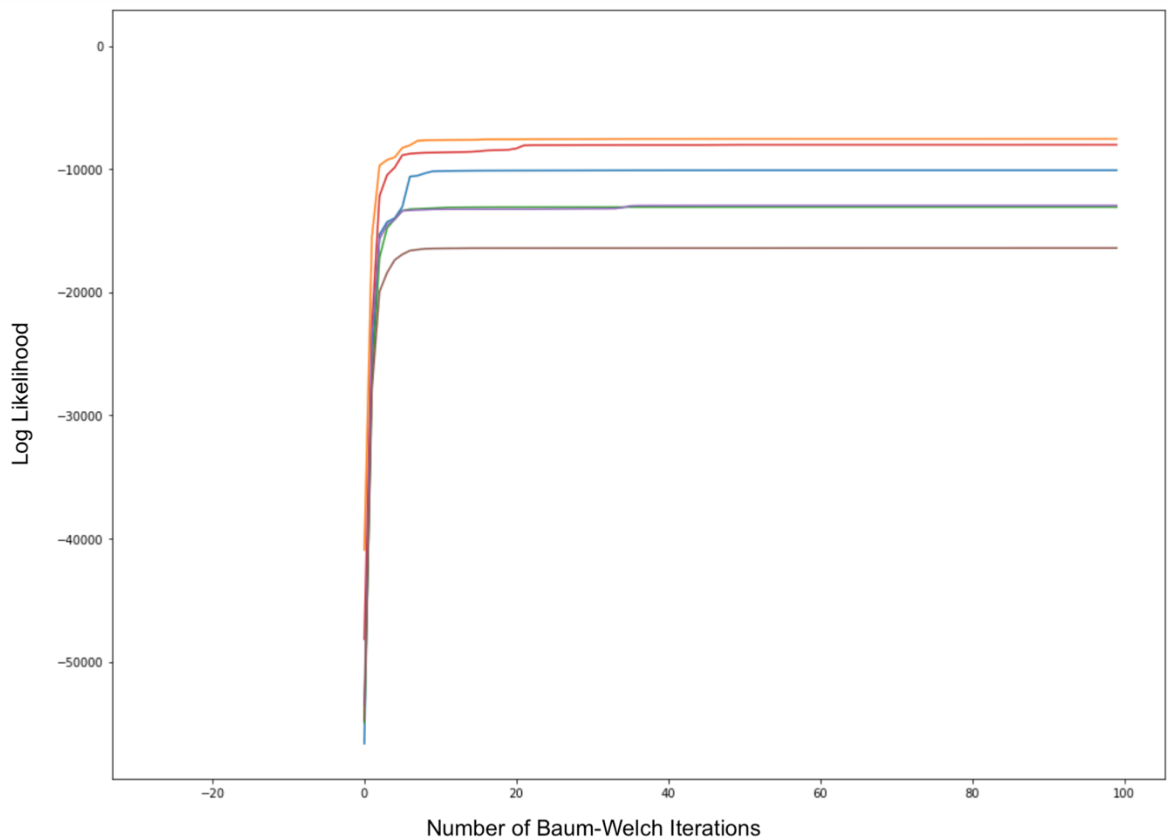

Figure 3: Log Likelihoods

Results

The output of the program was successfully able to produce the likelihood of each test dataset falling under the classification of a certain gesture. The results are as follows:

```
Evaluation for test7.txt:
Eight Score: -4585.4648891108
Inf Score: -6324.246970258767
Beat3 Score: -8234.99495875906
Beat4 Score: -8302.148613233352
Wave Score: -11386.935671977146
Circle Score: -14036.558726891939
Detected Gesture: Eight
```

```
-------------------------------------
Evaluation for test6.txt:
Inf Score: -766.3598395066556
Eight Score: -4830.033672549272
Beat3 Score: -9057.08943885544
Beat4 Score: -10073.619730186236
Wave Score: -14791.505360064555
Circle Score: -15477.303949378706
Detected Gesture: Inf
-------------------------------------
Evaluation for test4.txt:
Beat4 Score: -1780.7851940844027
Beat3 Score: -6544.585048289448
Inf Score: -7617.860339456066
Wave Score: -8236.102008405396
Eight Score: -12316.72627938073
Circle Score: -13988.94869824849
Detected Gesture: Beat4
-------------------------------------
Evaluation for test5.txt:
Circle Score: -2838.7759257775656
Beat3 Score: -5793.319778975688
Beat4 Score: -6346.606485259186
Eight Score: -12328.168192061734
Wave Score: -12691.849032583355
Inf Score: -12691.849032583355
Detected Gesture: Circle
-------------------------------------
Evaluation for test1.txt:
Wave Score: -270.253129381035
Eight Score: -4970.000887397306
Inf Score: -5039.506852311782
Beat4 Score: -5429.830879326129
Beat3 Score: -5659.548093771853
Circle Score: -5757.289253891479
Detected Gesture: Wave
-------------------------------------
Evaluation for test2.txt:
Beat4 Score: -1004.1182294520441
Beat3 Score: -1100.4174986394285
Inf Score: -9307.054011090608
Wave Score: -10757.278167373397
Eight Score: -11072.571550834178
Circle Score: -14842.66884718885
Detected Gesture: Beat4
-------------------------------------
Evaluation for test3.txt:
Inf Score: -2366.7926957848936
Eight Score: -7962.4006850628075
Beat3 Score: -8373.112871928002
Beat4 Score: -8712.995178969975
Wave Score: -14858.26464069804
Circle Score: -15069.294609363616
Detected Gesture: Inf
```

```
-------------------------------------
Evaluation for test8.txt:
Beat4 Score: -895.7244235693356
Beat3 Score: -1228.4398756500204
Inf Score: -8959.207352177784
Wave Score: -10263.229935302144
Eight Score: -10734.046587708153
Circle Score: -14326.04864182483
Detected Gesture: Beat4
-------------------------------------
```

Discussion

My results show the probability of each gesture being resembled. For each test file, the gestures are ranked in order from best to worst fit. In some cases, like in the case of test1, the model very confidently identifies the gesture, as the first likelihood is much higher than the rest. However, in other cases, the model is much less confident, such as in the case of test2, where it's practically a tossup between the first two choices: beat4 and beat3. When classifying the gesture, the likelihoods of the next couple values should always be taken into account to assess how much the model can be trusted.

References

While all of the code written for this project was my own, a number of references aside from course materials were used to aid in my understanding, including the following:

http://www.cs.cmu.edu/~roni/11661/2017_fall_assignments/shen_tutorial.pdf
https://medium.com/mlearning-ai/baum-welch-algorithm-4d4514cf9dbe
https://hmmlearn.readthedocs.io/en/latest/