# IMU Based Gesture Recognition using Hidden Markov Model

Nitin J. Sanket
School of Engineering and
Applied Science
University of Pennsylvania
Email: nitinsan@seas.upenn.edu

***Abstract*—This project presents an approach for recognizing gestures from accelerometer and gyroscope data. The raw data is pre-processed and passed though a trained hidden markov model. Log-likelihood is then calculated for each class and the class with the highest likelihood is chosen and the result. A confidence metric is defined and bar plots of results are presented. Low-pass filtering helps to obtain a more robust output.**

## I. Problem Statement

The aim of the project was to recognize gestures from the given accelerometer and gyroscope data.

## II. Pre-processing

An experiment with different averaging filters was performed. The following filters were tried:

1) **Raw Data:**
   No pre-processing was done. Just raw data was fed into the HMM. The output was reasonably accurate ( 90%), however the sequences corresponding to beat3 and beat4 had very little difference in confidence ( 2-5%).
2) **Exponential Smoothing:**
   Exponential Smoothing performed better than raw data as it could remove some noise in the data. This also boosted the difference in confidence between beat3 and beat4 sequences to about 5-12%. The equations governing exponential smoothing (from Wikipedia) are given below:

$$\begin{aligned} s_0 &= x_0 \\ s_t &= \alpha \frac{x_t}{c_{t-L}} + (1-\alpha)(s_{t-1} + b_{t-1}) \\ b_t &= \beta(s_t - s_{t-1}) + (1-\beta)b_{t-1} \\ c_t &= \gamma \frac{x_t}{s_t} + (1-\gamma)c_{t-L} \\ F_{t+m} &= (s_t + mb_t)c_{t-L+1+(m-1) \mod L} \end{aligned}$$

   Here $\alpha$ is the data smoothing factor and lies between 0 and 1. Also, $\beta$ is the trend smoothing factor and also lies between 0 and 1. And, $\gamma$ is the seasonal change smoothing factor and lies between 0 and 1. $F$ is the filtered data and $x$ is the raw data.
3) **Average Envelope:**
   Generally, for speech signals top and bottom envelopes are computed so as to capture the peak information. A simple mean of these envelopes will give the average envelope of the underlying signal. Using this feature doesn't make sense for orientation as minor changes in direction are lost which is important for beat3 and beat4. As expected, this dropped the confidence and also resulted in misclassifications.
4) **Savitzky-Golay Filter:**
   The Savitzky-Golay Filter internally computes the smoothing polynomial coefficients, performs delay alignment, and takes care of transient effects at the start and end of the data sequence. As expected, this filter works very well and boosts the confidence between beat3 and beat4 sequences to about 4-9%.
5) **Median Filter:**
   The simplest low-pass filter is a median filter (it just computes the median over a specified window size) and surprisingly, it outperformed all other methods giving a confidence boost of 6-14% for beat3 and beat4.

A comparision of the aforementioned filters on the accelerometer data (X direction) is shown in Fig. 1.

Other pre-processing techniques which were tried are summarized below:

1) **Standardizing Data:**
   The data was standardized by making mean zero and standard deviation one. It boosted the confidence score but also resulted in some misclassifications as it was

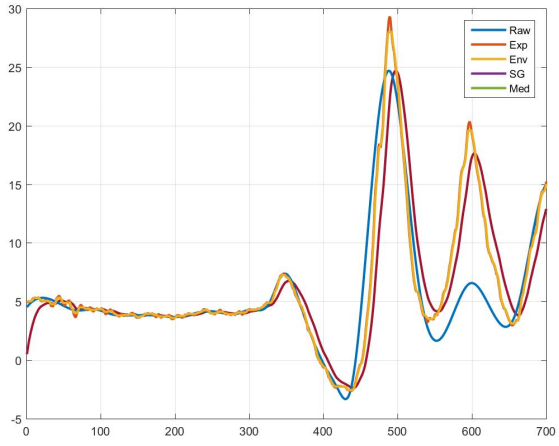


Fig. 1: Comparison of different filters for filtering data.

highly susceptible to K-means cluster centers.

2) **Extracting Orientation Using a Complementary Filter:**
A simple complementary filter was used to compute the euler angles from the accelerometer and gyroscope data. The idea behind the complementary filter [1] is that attitude is computed separately from both accelerometer and gyroscope. The IMU body axis is defined as Z up, X forward and Y left (looking from IMU's point of view). The accelerometer is symmetric with respect to Z axis hence it doesn't provide any information about Yaw ($\psi$) angle. The other two angles [2] are calculated as follows:

$$Roll, \phi = \tan^{-1}\left(\frac{a_y}{\sqrt{{a_x}^2 + {a_z}^2}}\right)$$

$$Pitch, \theta = \tan^{-1}\left(\frac{-a_x}{\sqrt{{a_y}^2 + {a_z}^2}}\right)$$

However, generally the IMU is never completely vertical, and hence the Yaw can be retrieved as

$$Yaw, \psi = \tan^{-1}\left(\frac{\sqrt{{a_x}^2 + {a_y}^2}}{a_z}\right)$$

The gyroscope values are integrated to obtain the angles. The integration is performed using quaternions and then the euler angles are computed from the quaternions. The update quaternion is calculated as below:

$$\begin{aligned}
\alpha_\Delta &= |\vec{\omega}_k|\,\Delta t \\
\vec{e}_\Delta &= \tfrac{\vec{\omega}_k}{|\vec{\omega}_k|} \\
q_\Delta &= \left(\cos\left(\tfrac{\alpha_\Delta}{2}\right), \vec{e}_\Delta \sin\left(\tfrac{\alpha_\Delta}{2}\right)\right)
\end{aligned}$$

To calculate current state quaternion the following equations are used

$$\begin{aligned}
\alpha &= |\vec{\omega}_k| \\
\vec{e} &= \tfrac{\vec{\omega}_k}{|\vec{\omega}_k|} \\
q_k &= \left(\cos\left(\tfrac{\alpha}{2}\right), \vec{e}\sin\left(\tfrac{\alpha}{2}\right)\right)
\end{aligned}$$

Now, the new state quaternion is described as ($k$ is the current state and $k+1$ is the next state),

$$q_{k+1} = q_k q_\Delta$$

The euler angles are calculated from the quaternion as,

$$\begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} = \begin{bmatrix} \tan^{-1}\left(\frac{2(q_0q_1+q_2q_3)}{1-2(q_1^2+q_2^2)}\right) \\ \sin^{-1}\left(2(q_0q_2 - q_3q_1)\right) \\ \tan^{-1}\left(\frac{2(q_0q_3+q_1q_2)}{1-2(q_2^2+q_3^2)}\right) \end{bmatrix}$$

The angle measurements are blended as follows in the complementary filter,

$$\begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix}_{CF} = \begin{bmatrix} \alpha & 0 & 0 \\ 0 & \beta & 0 \\ 0 & 0 & \gamma \end{bmatrix} \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix}_{acc} + \begin{bmatrix} 1-\alpha & 0 & 0 \\ 0 & 1-\beta & 0 \\ 0 & 0 & 1-\gamma \end{bmatrix} \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix}_{gyro}$$

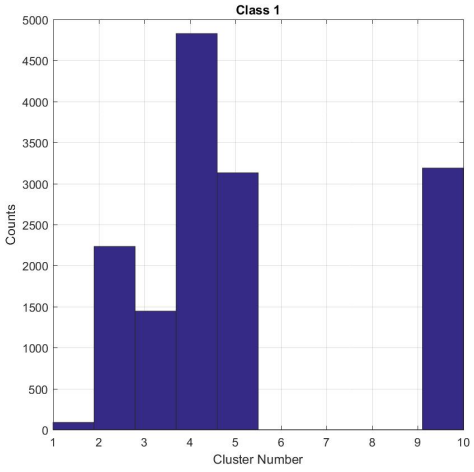


Fig. 2: Histogram of cluster disribution for beat3.

Here $\alpha, \beta$and, $\gamma$ are the mixing parameters. $\alpha$ was chosen as 0.75, $\beta$ was chosen as 0.75 and $\gamma$ was chosen as 1.0. Effectively the gyro measurements are high pass filtered to remove drift and accelerometer measurements are low pass filtered to remove noise.
The results using orientation were bad because while recording the gestures there was very minimal change in orientation and the main information would've been in the translation which we don't have. Also a simple eucledian distance is the wrong metric to use for K-Means clustering for euler angles as it does not respect the periodicity in the angles and difference between angle space in $\mathcal{SO}(3)$ cannot be calculated using square distance.

## III. Vector Quantization

A simple K-Means clustering was performed on the filtered data. The distance metric used was euclidian distance. The histogram plots of the cluster assignments per class is are shown in Figs. 2, 3, 4, 5, 6 and 7 for the classes beat3, beat4, circle, eight, inf and wave respectively. The same histogram for all data is shown in Fig. 8. Clearly beat3 and beat4 look very similar. This indicates that raw counts cannot distinguish between them and hence the temporal order is very important, this is the motivation for the usage of a HMM. Also, for reference, the cluster assignments for accelerometer values is shown in Figs. 9, 10, 11, 12, 13 and 14 for the classes beat3, beat4, circle, eight, inf and wave respectively. The same histogram for all data is shown in Fig. 15. It is funny to observe that we see a circle for the pattern eight (Fig. 12) and wave for the pattern wave (Fig. 14) in the accelerometer values.

## IV. Hidden Markov Model (HMM)

In this project, a discrete HMM was used. The HMM has the following parameters:
$N$ possible states.
$M$ possible observations (number of clusters).

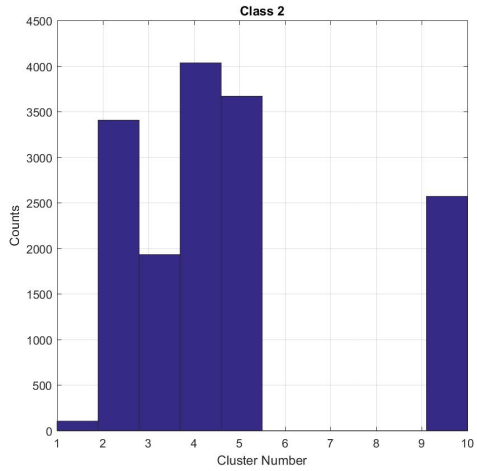


Fig. 3: Histogram of cluster disribution for beat4.

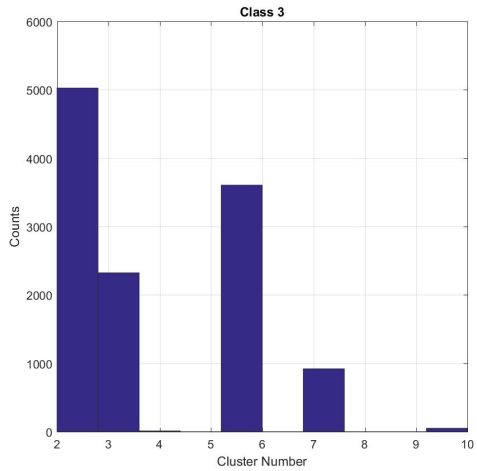


Fig. 4: Histogram of cluster disribution for circle.

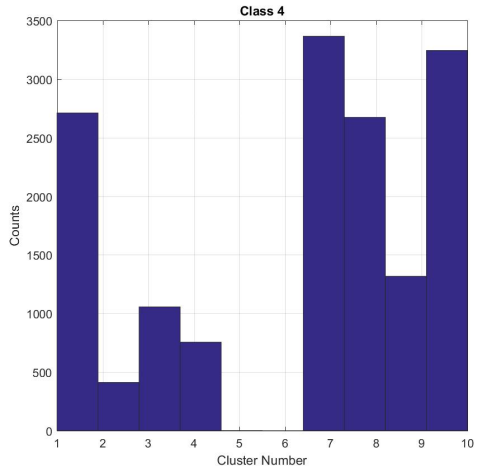


Fig. 5: Histogram of cluster disribution for eight.

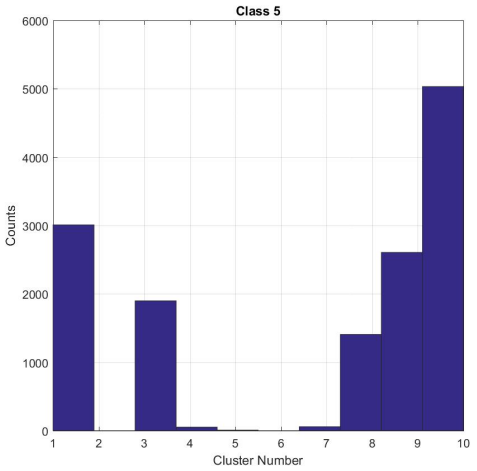


Fig. 6: Histogram of cluster disribution for inf.

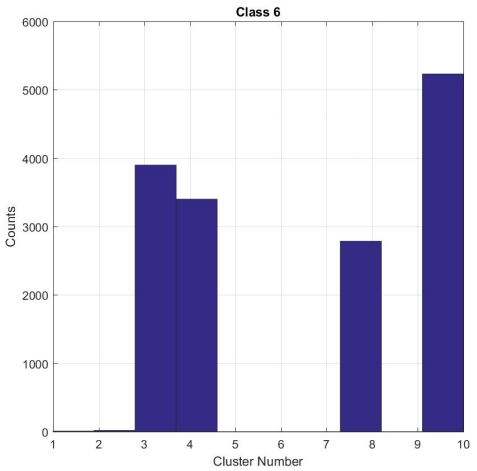


Fig. 7: Histogram of cluster disribution for wave.

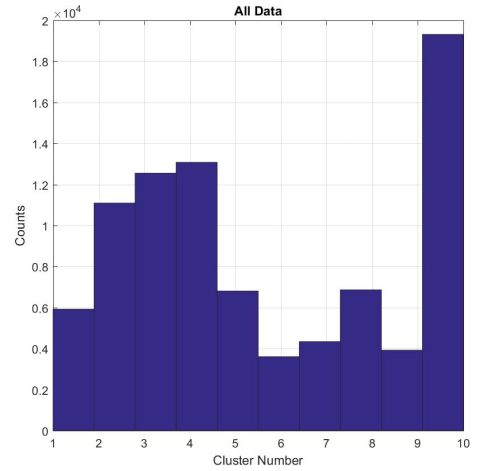


Fig. 8: Histogram of cluster disribution for all classes.

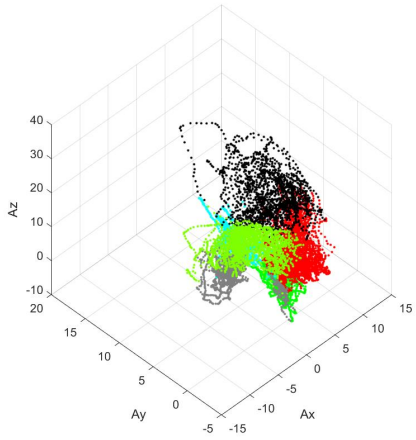


Fig. 9: Clusters shown in difference colors in accelerometer space for beat3.

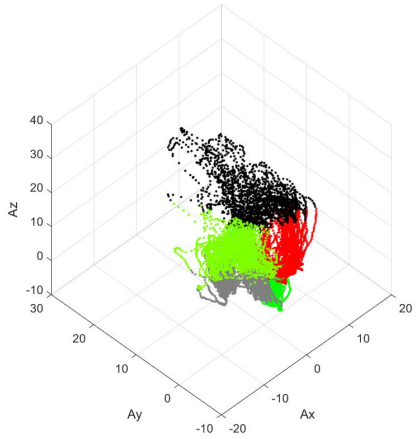


Fig. 10: Clusters shown in difference colors in accelerometer space for beat4.

$A$ is a $N \times N$ state transition matrix.
$B$ is a $N \times M$ observation generation matrix.
$\pi$ is the prior on state probabilities.
Let's call $\lambda = (A, B, \pi)$. We are trying to estimate $\lambda$ which best describes a sequence.

### A. Baum-Welch Algorithm [3]

Here, $b_j(O_k)$ means that we are picking the $O_k^{th}$ column of $B$, $T$ is the sample length, $a_{ij}$ is the $i^{th}$ row and $j^{th}$ column element of $A$.

### B. Architecture of State Transition Matrix $A$

Both Left-to-right and ergodic structure were tried. The Left-to-right structure was faster to train but was more susceptible to noise than the ergodic structure maybe due to lesser complexity. Hence, the ergodic structure was used. A sample

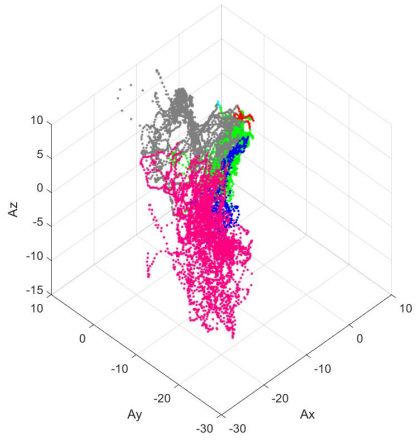


Fig. 11: Clusters shown in difference colors in accelerometer space for circle.

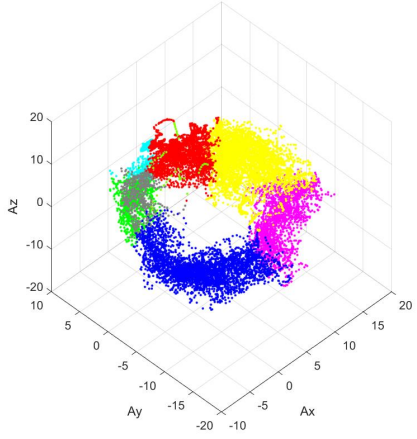


Fig. 12: Clusters shown in difference colors in accelerometer space for eight.

Left-to-right structure used (for $N = 4$) is shown below:

$$A = \begin{bmatrix} a & 1-a & 0 & 0 \\ 0 & a & 1-a & 0 \\ 0 & 0 & a & 1-a \\ 1-a & 0 & 0 & a \end{bmatrix}$$

Here $a$ is the probability of remaining in the same state, which is calculated as

$$a = 1 - \frac{N}{T}$$

where $N$ is the number of hidden states and $T$ is the number of samples.

### C. Initialization of $A, B, \pi$

$A$ was initialized to a $N \times N$ random matrix and normalized such that each column sums to 1. Random initialization performed better than uniform initialization as

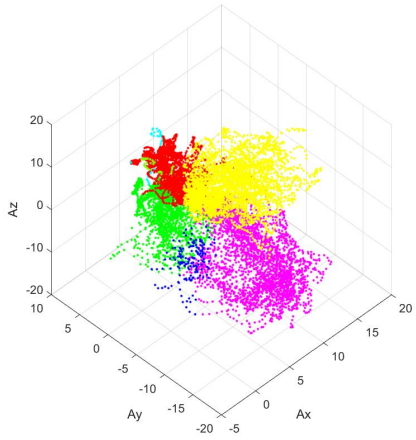


Fig. 13: Clusters shown in difference colors in accelerometer space for inf.

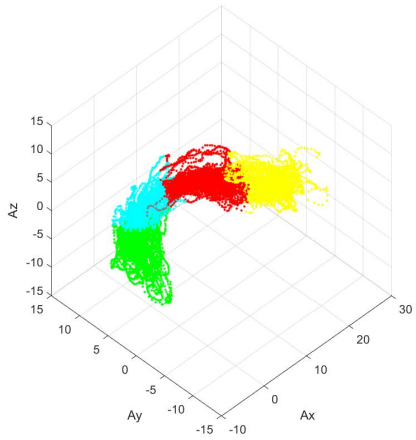


Fig. 14: Clusters shown in difference colors in accelerometer space for wave.

uniform distribution is close to a sub-optimal local minima and takes a long time to converge with poor results. Though there is no guarantee that random initialization would perform better, from experiments it was observed that, generally, random initalization outperformed uniform distribution.
Similarly, $B$ was initialized to a $N \times M$ random matrix and normalized such that each row sums to 1. Here also, random initialization outperformed uniform distribution.
Similarly, $Pi$ was initialized to a $N \times 1$ random vector and normalized such that the norm is 1. Here also, random initialization outperformed uniform distribution.

### D. Training

Each file of each pattern was training using the Baum-Welch Algorithm (Refer to 1). The final model parameters were obtained by averaging all these values for each pattern.

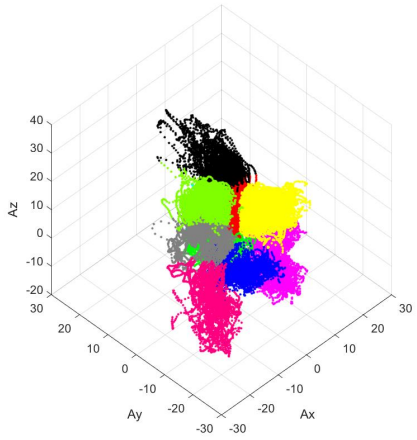


Fig. 15: Clusters shown in difference colors in accelerometer space for all classes.

These are desribed by the following equations:

$$\bar{a}_{ij} = \frac{\sum_{k=1}^{K} a_{ij}^k}{K}$$
$$\bar{b}_{ij} = \frac{\sum_{k=1}^{K} b_{ij}^k}{K}$$
$$\bar{\pi}_i = \frac{\sum_{k=1}^{K} \pi_i^k}{K}$$

Here, $K$ is the number of files for each pattern. One should note that, however, there is no theoretical guarantee that an average of local minimas will be a local minima.

### E. Convergence Criterion

The maximum number of iterations was chosen as 25 from cross-validation (in-order to avoid overfitting). However, an early termination criterion was defined to end the code early if the change in log probabilities was too low. The convergence criterion was defined as,

$$|P_{t+1} - P_t| \leq \delta$$
$$P_t = \frac{-1}{\sum_{t=1}^{T} \log \sum_{i=1}^{N} \alpha_t(i)}$$

Here $\delta$ was chosen as $10^{-7}$.

### F. Finding Optimal $M$ and $N$

The right way of finding the combination of optimal $M$ and $N$ would be to have a meshgrid of values and do cross-validation on each of them. However, this is painfully slow. Hence to settle for a local minima, I found the $M$ value with $N$ fixed first and then fixed $M$ to this optimal value and found $N$. A plot of this cross-validation is shown in Figs. 16 and 17. The cross-validation was based on maximizing the minimum value of confidence scores throughout the dataset, this is like maximizing the lower bound to maximize the objective function by using Jensen's inequality.

**Data**: Vector Quantized Data
**Result**: $\lambda = (A, B, \pi)$
**Expectation Step**
% Forward Procedure
Initialize $\alpha_1(i) = \pi_i b_i(O_1)$
% Normalize $\alpha_1(i) = \frac{\alpha_1(i)}{\sum_{i=1}^{N} \alpha_1(i)}$
**for** $t \in [1, T-1]$
**do**
  % Induction Step
  $\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right] b_j(O_1)$
  % Normalize
  $\alpha_{t+1}(i) = \frac{\alpha_{t+1}(i)}{\sum_{i=1}^{N} \alpha_{t+1}(i)}$
**end**
% Backward Procedure
Initialize $\beta_T(i) = 1$
% Normalize $\beta_T(i) = \frac{\beta_T(i)}{\sum_{i=1}^{N} \beta_T(i)}$ **for** $t \in [T-1, 1]$
**do**
  % Induction Step
  $\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)$
  % Normalize $\beta_t(i) = \frac{\beta_t(i)}{\sum_{i=1}^{N} \beta_t(i)}$
**end**
% Solution to Problem 2 to estimate $\gamma$
$\gamma_t = \frac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^{N} \alpha_t(i)\beta_t(i)}$
% Solution to Problem 3 to estimate $\xi$
$\xi_t(i,j) = \frac{\alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}$
**Maximization Step**
$\bar{\pi}_i = \gamma_1(i)$
$\bar{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)}$
$\bar{b}_j(k) = \frac{\sum_{t=1}^{T-1} \gamma_t(i), s.t.\, O_t = v_k}{\sum_{t=1}^{T-1} \gamma_t(i)}$
% Laplace Smoothing $B(B < 10^{-12}) = 10^{-12}$
% Normalize A, B and Pi

**Algorithm 1:** Numerically stable Baum-Welch Algorithm for HMMs

### G. Testing and Confidence Calculation

Each new test file is tested against all the pattern models. To calculate log-likelihood score for each pattern, the test data-sample is passed through the Forward step of Baum-Welch Algorithm (Refer to 1). The log-likelihood is computed as:

$$P_t = \frac{-1}{\sum_{t=1}^{T} \log \sum_{i=1}^{N} \alpha_t(i)}$$

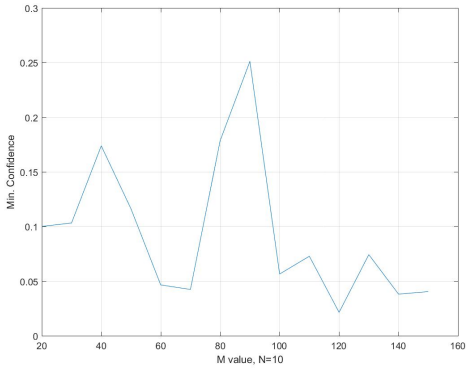


Fig. 16: Min. Confidence score for finding optimal $M$.

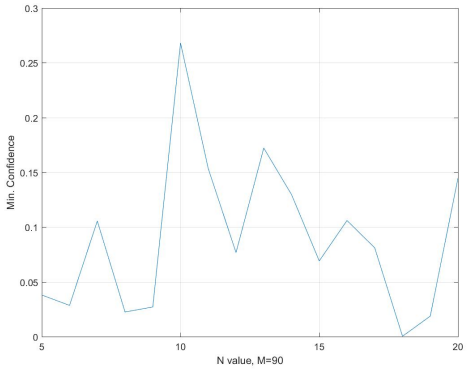


Fig. 17: Min. Confidence score for finding optimal $N$.

The confidence score ($\chi$) is defined as the percentage difference between the first most-likely ($P_1$) and second most-likely class ($P_2$).

$$\chi = \left(\frac{P_1 - P_2}{P_1}\right) \times 100$$

## V. Other interesting things I tried

### A. Early Termination in Testing

As HMM can run on varying length data samples, I tried to decide if only by using a few samples we can predict the pattern. The early termination criterion was based on confidence score, if the confidence was higher than a value, the code would terminate. However, this was very noisy and susceptible to how much idle state the pattern had and the speed at which the pattern was performed. If, a speed matching algorithm was used and the idle sequences removed, this would've made the testing much much faster. As, none of the aforementioned techniques were used, this method failed miserably.

### B. Initialization of $A, B, \pi$ from previous sample

Instead of initializing $A, B, \pi$ randomly for every file in a pattern, I tied to initialize the next pattern by the model obtained from the previous file of the same pattern. This had the tendency to overfit the last sample as expected and hence

resulted in worse results than simply randomizing for every file and taking the average.

## VI. Results

The bar plots for the Training Set are given in Figs. 18 to 59. The overall accuracy was 95.24% with 2 miss-classifications. Left to right in all figures is beat3, beat4, circle, eight, inf and wave.

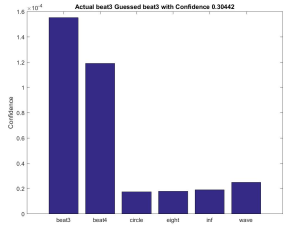


Fig. 18: Training Set Sample 1.

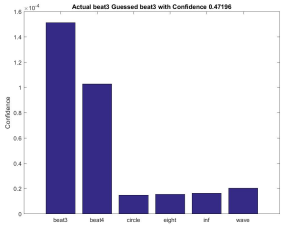


Fig. 19: Training Set Sample 2.

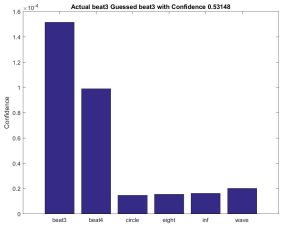


Fig. 20: Training Set Sample 3.

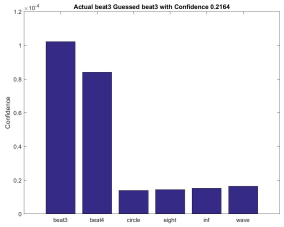


Fig. 21: Training Set Sample 4.

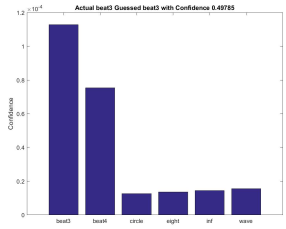


Fig. 22: Training Set Sample 5.

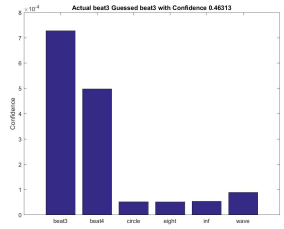


Fig. 23: Training Set Sample 6.

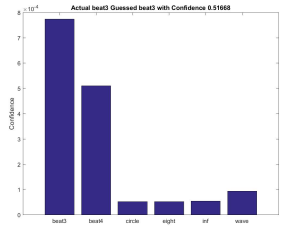


Fig. 24: Training Set Sample 7.

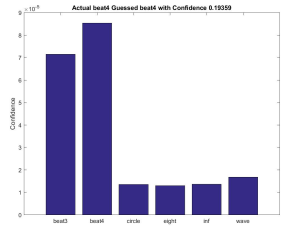


Fig. 25: Training Set Sample 8.

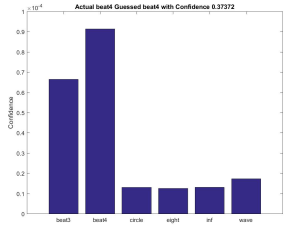


Fig. 26: Training Set Sample 9.

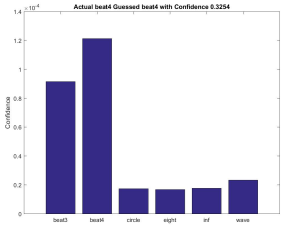


Fig. 27: Training Set Sample 10.

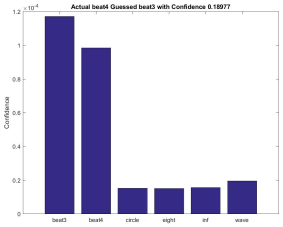


Fig. 28: Training Set Sample 11.

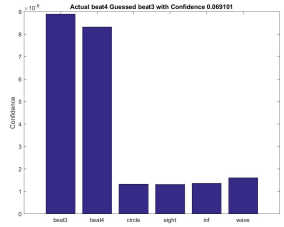


Fig. 29: Training Set Sample 12.

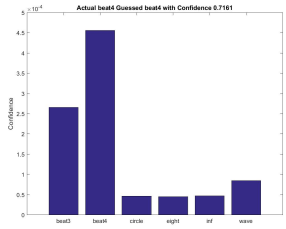


Fig. 30: Training Set Sample 13.

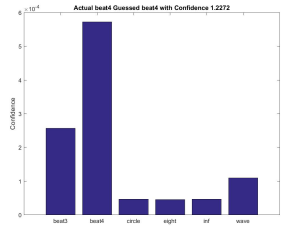


Fig. 31: Training Set Sample 14.

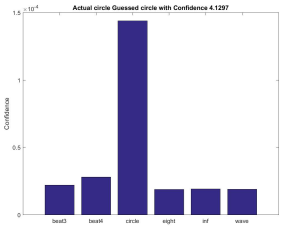


Fig. 32: Training Set Sample 15.

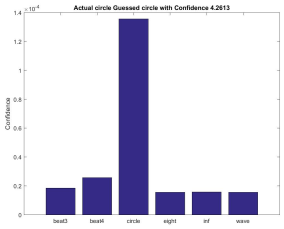


Fig. 33: Training Set Sample 16.

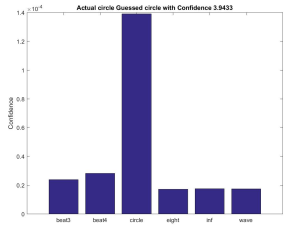


Fig. 34: Training Set Sample 17.

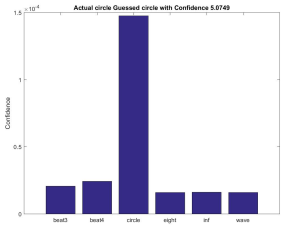


Fig. 35: Training Set Sample 18.

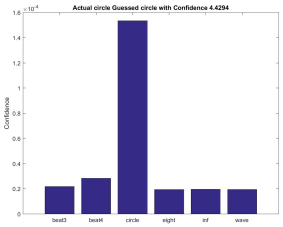


Fig. 36: Training Set Sample 19.

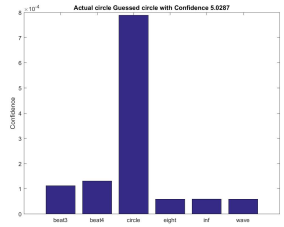


Fig. 37: Training Set Sample 20.

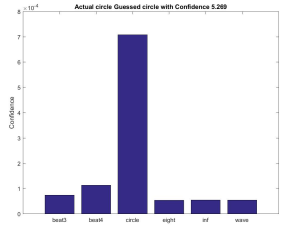


Fig. 38: Training Set Sample 21.

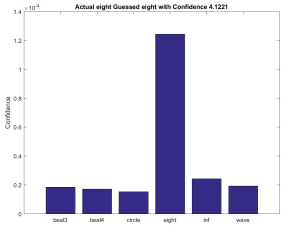


Fig. 39: Training Set Sample 22.

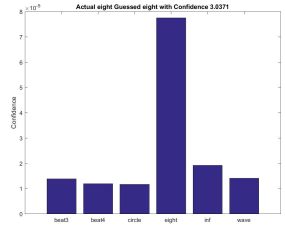


Fig. 40: Training Set Sample 23.

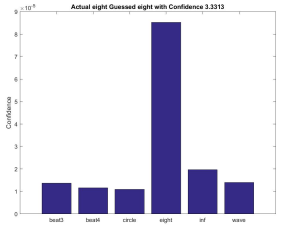


Fig. 41: Training Set Sample 24.

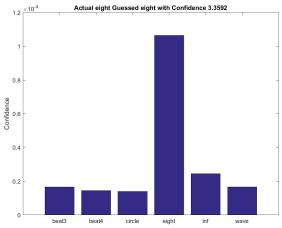


Fig. 42: Training Set Sample 25.

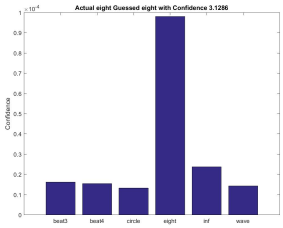


Fig. 43: Training Set Sample 26.

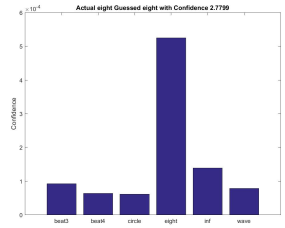


Fig. 44: Training Set Sample 27.

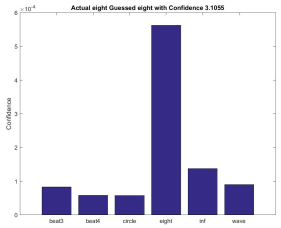


Fig. 45: Training Set Sample 28.

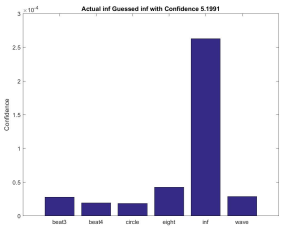


Fig. 46: Training Set Sample 29.

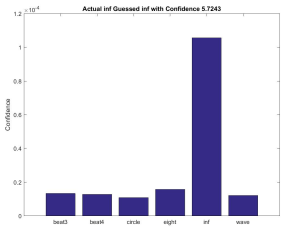


Fig. 47: Training Set Sample 30.

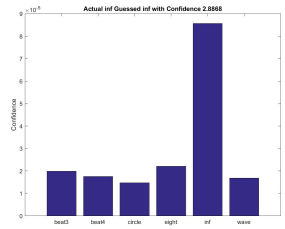


Fig. 48: Training Set Sample 31.

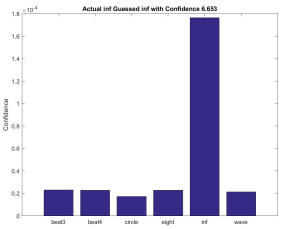


Fig. 49: Training Set Sample 32.

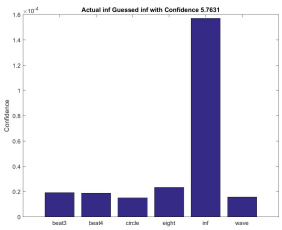


Fig. 50: Training Set Sample 33.

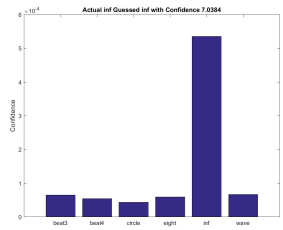


Fig. 51: Training Set Sample 34.

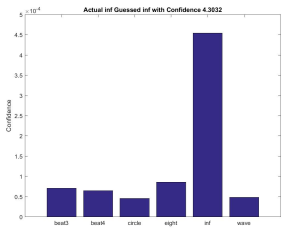


Fig. 52: Training Set Sample 35.

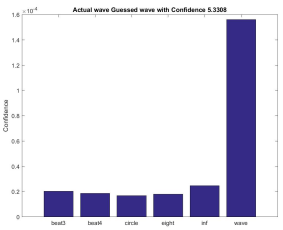


Fig. 53: Training Set Sample 36.

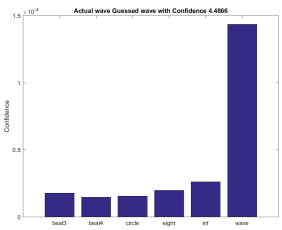


Fig. 54: Training Set Sample 37.

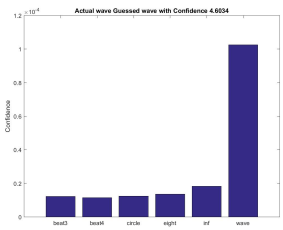


Fig. 55: Training Set Sample 38.

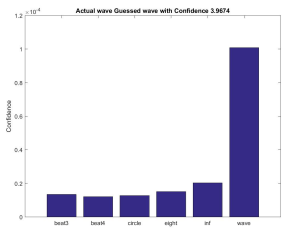


Fig. 56: Training Set Sample 39.

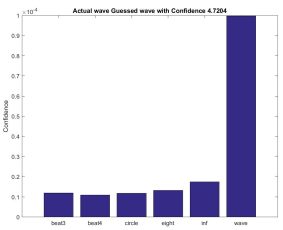


Fig. 57: Training Set Sample 40.

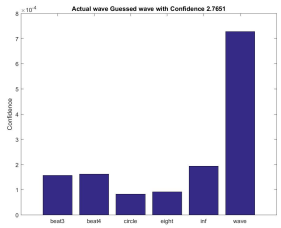


Fig. 58: Training Set Sample 41.

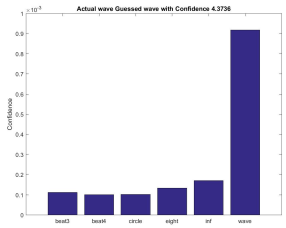


Fig. 59: Training Set Sample 42.

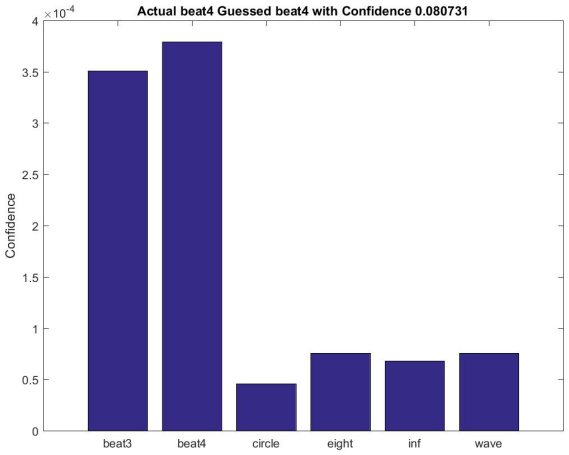


Fig. 61: Testing Set Sample 2 (Guessed beat4).

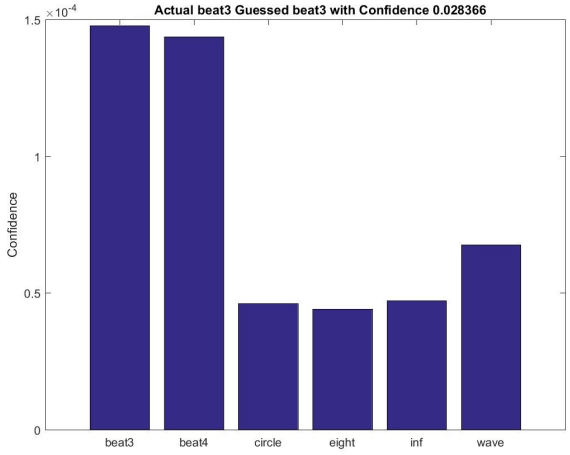


Fig. 63: Testing Set Sample 4 (Guessed beat3).

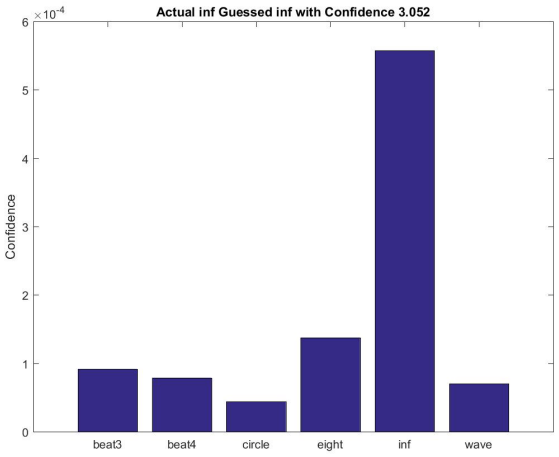


Fig. 62: Testing Set Sample 3 (Guessed inf).

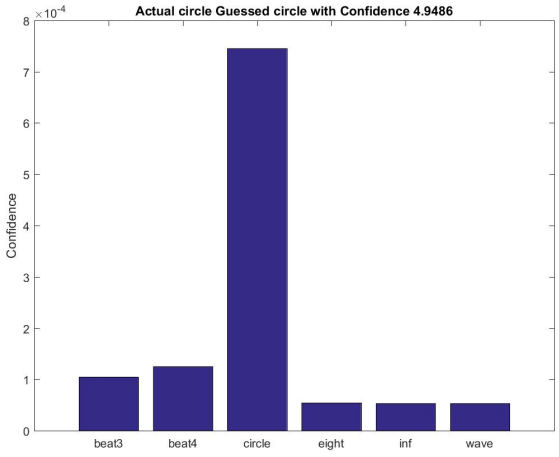


Fig. 64: Testing Set Sample 5 (Guessed circle).

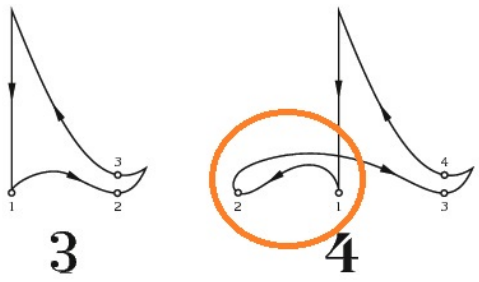


Fig. 68: Comparison of Beat3 and Beat4.

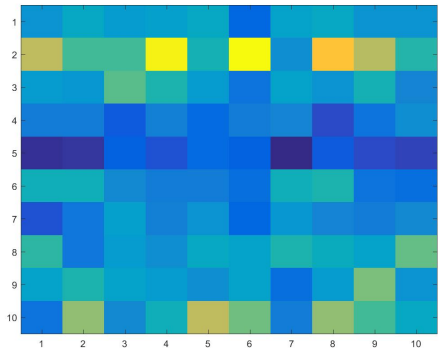

Fig. 69: $A$ for beat3.

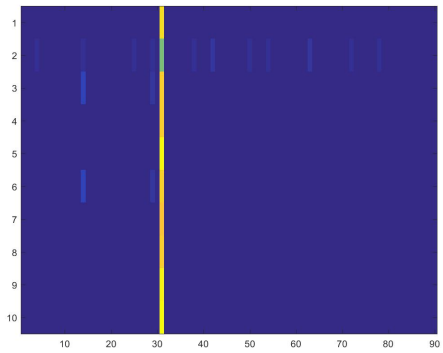

Fig. 70: $B$ for beat3.

The bar plots for the Testing Set are given in Figs. 60 to 67. The overall accuracy was 95.24% with 2 miss-classifications. Left to right in all figures is beat3, beat4, circle, eight, inf and wave. Note that I had to retrain my model as the model I submitted and presented in class used only 12 sequences to train which resulted in the miss-classification. When I used all the 42 sequences to train, I got all the answers correctly.

### A. *Analysis of results*

Clearly the confusions are mostly in beat3 and beat4 (the confidence is low, i.e., a little noise can make the prediction wrong), this is because the strokes are very similar. Refer to Fig. 68, the only difference between beat3 and beat4 is the orange highlighted left stroke. If this stroke is small (time-wise and/or amplitude-wise), the beats get easily confused.

A sample visualization of the state transition matrix $A$ and observation generation matrix $B$ for beat3 are given in Figs. 69 and 70 respectively.

## VII. Important Lessons Learnt

Simple filters can boost the performance of the system enormously.

You need to re-normalize and perform laplace smoothing to maintain numerical stability.

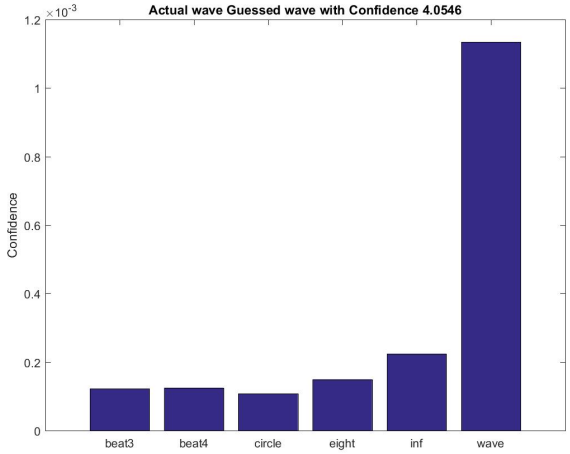


Fig. 60: Testing Set Sample 1 (Guessed wave).

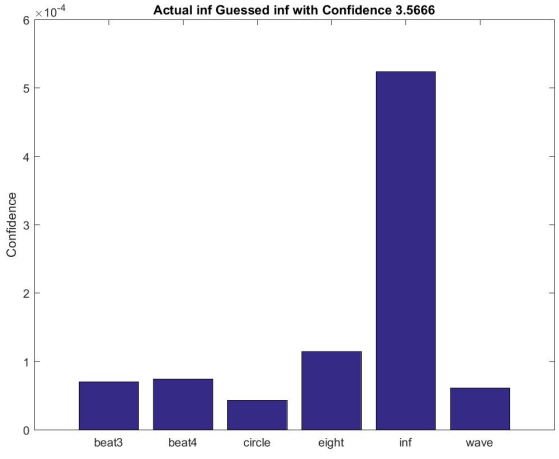


Fig. 65: Testing Set Sample 6 (Guessed inf).

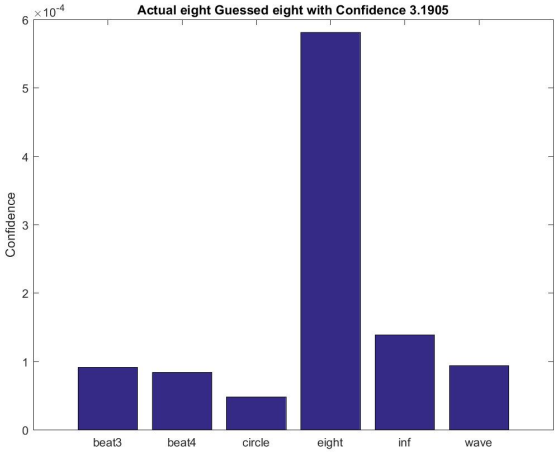


Fig. 66: Testing Set Sample 7 (Guessed eight).

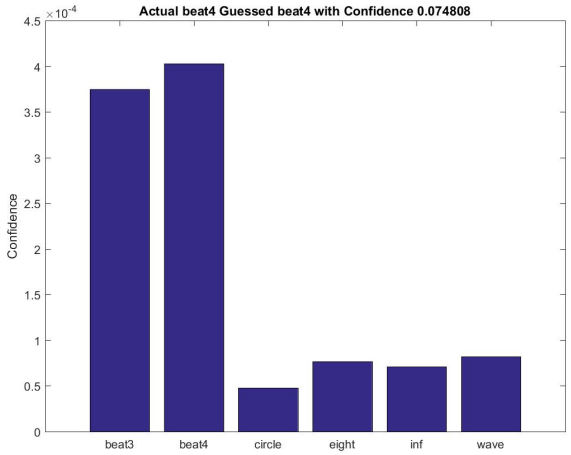


Fig. 67: Testing Set Sample 8 (Guessed beat4).

## VIII. Conclusions

Overall, HMM is a robust method for gesture recognition. Simple filtering approaches like median filter boosted the confidence of the classifier. beat3 and beat4 sequences were the most hard ones to classify. The accuracy on the training set was 95.24% and the accuracy on the test set was 100%. The performance on the test set was good though it was susceptible to changes when the code was rerun dude to random initializations.

## Acknowledgment

The author would like to thank Dr. Daniel D. Lee and all the Teaching Assistants of ESE 650 course for all the help in accomplishing this project.

## References

[1] Complementary Filter, Link
[2] Attitude from accelerometer, Link
[3] Lawrence R. Rabiner, *A tutorial on hidden Markov models and selected applications in speech recognition*, Proceedings of the IEEE Vol. 77, No. 2, pp. 257–286, 1989.