ESE650 Project 3:

Gesture Recognition using Hidden Markov Models

Nischal K N
nischal@seas.upenn.edu

Abstract
Hidden Markov Model(HMM) is a statistical model of Markov model, in which some hidden state of the system is modeled. The hidden state is a state which no sensor can measure directly or cannot be defined. This project aims to detect the type of motion performed by the user with a mobile phone. It uses the information from the IMU to model the state of motion of the body(hidden) using HMM.

Contents

1. Introduction

In this project data from the IMU of a mobile phone is used to determine the type of motion traced by the user. The motion can be one of 6 types, viz., beat 3, beat 4, circle, wave, infinity and eight. The data obtained from the sensor is first subjected to some pre-processing described in section 2. Then an HMM model is trained using forward backward algorithm for each type of gesture. The different parameter selections including number of hidden states, number of observation classes, initial state of transition matrices,etc for HMM training are described in 4. The trained models are then used to find the log likelihood of the give test set for each gesture. A confidence of prediction is also determined. The testing and training process is explained in section 3. The performance of the test set and the results are presented in section 5.

2. Dataset and Pre-processing

The dataset is a 6 axis IMU data consisting of 3 axis accelerometer and 3 axis gyroscope data. The bias and scale of the data is already accounted for. A simply median filter was applied over this data to remove high frequency components. This was performed using the inbuilt matlab function medfilt1. The raw data and output after median filter is shown in 1. it is seen that the median filter output is almost the same as the input. There is no noticeable change even in the output. Hence it was not used in the final implementation.

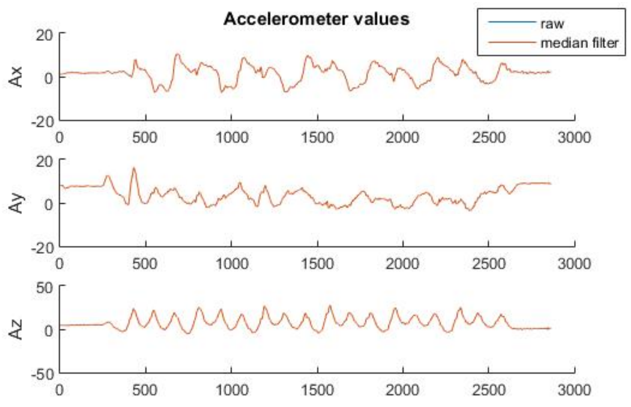

Figure 1. Median Filtering

The data across all the training sets were clustered using K-means clustering. The inbuilt kmeans matlab function is used for clustering. The maximum number of iterations was set to 500 and the number of clusters were determined from cross validation explained in section 4. Each 6 dimension datapoint was mapped to the a cluster number to which it belonged. This 1 dimension vector was used as the observation sequence to perform HMM.

3. Methodology

The implementation follows the method described in []. The training is implemented using the forward backward and Baum–Welch restimation algorithm. Then the forward algorithm is used to determine the alpha's and hence compute the log likelihood of the the test sequence with respect to each gesture model.

3.1 Training

The forward backward algorithm is used to compute the posterior of all the hidden variables using the given sequence of observations of each training dataset. The number of hidden states N is determined by cross validation. The overview of the algorithm is described below,

1. The state transition matrices A & B and initial state π were randomly initialized and also normalized. The intial state of hidden variables α and β were initialized as,

$$\alpha_1(i) = \pi_i B_i(O_i)), 1 \leq i \leq N$$
$$\beta_1(i) = 1, 1 \leq i \leq N$$

where O is the observation vector obtained from clustering the IMU data using k-means and N is the number of hidden states.

2. alpha is determined for t = 1:T-1 using the induction step.

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) A_{ij}\right] b_j(O_{t+1})$$

At each step alpha is normalized using the sum of all the N elements of α defined as c_t. This helps to prevent underflow.

$$c_t = \sum_{i=1}^{N} \alpha_t(i)$$

3. beta is estimated from the end, i.e t = T-1:1 using,

$$\beta_t(i) = \sum_{j=1}^{N} A_{ij} B_j(O_{t+1}) \beta_{t+1}(j)$$

Similar to α, β is normalized using the sum of all the elements of β defined as d_t. This helps to prevent underflow.

$$d_t = \sum_{i=1}^{N} \alpha_t(i)$$

4. gamma and Xi are calculated as

$$\gamma(i) = \frac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^{N} \alpha_t(i)\beta_t(i)}$$
$$\xi(i,j) = \frac{\alpha_t(i) A_{ij} B_j(O_{t+1}) \beta_{t+1}(j)}{\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_t(i) A_{ij} B_j(O_{t+1}) \beta_{t+1}(j)}$$

5. Then Baum–Welch restimation equations are used to determine new A, B and Pi using

$$\pi(i) = \gamma(i)$$
$$A_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)}$$
$$B_j(k) = \frac{\sum_{t=1, s.t. O_t = v_k}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}$$

6. This process is repeated unless the log likelihood given by the below equation reaches a threshold

$$log[P(O \mid \lambda)] = -\sum_{t=1}^{T} log c_t$$

The variation of log likelihood over iterations is shown in Fig. 2. It is seen that it converges to a constant value after a few iterations. A threshold of 5×10^{-4} could be used to terminate the execution.

7. Two methods were tested for determining the model of each gesture

   - The process described above was repeated for different datasets with random initialization of A, B & Pi and an average was taken for each gesture.
   - Similar process was followed to determine A,B and Pi for the first dataset of each gesture. Then for the next dataset of the same gesture, the forward backward process was initialized to model parameters A,B and Pi of the first/previous dataset of the same gesture.

It was observed that both the models produced a very similar output, but however the latter was less robust as there was a lot of disagreement between beat 3 and beat 4 which have similar motion.

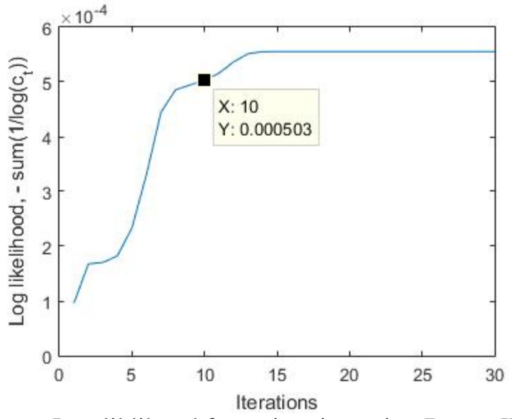

Figure 2. Log liklihood for estimation using Baum–Welch restimation equations

3.2 Testing

To predict the motion of the new gesture, the log likelihood of the test set with all the trained gesture models was calculated and the gesture with the highest value determined the prediction.

1. Assign each datapoint of the test set to the respective cluster generated by k-means in section 2. This is performed by calculating the minimum distance of the data point to all the cluster centers. This gives us the observation vector O.

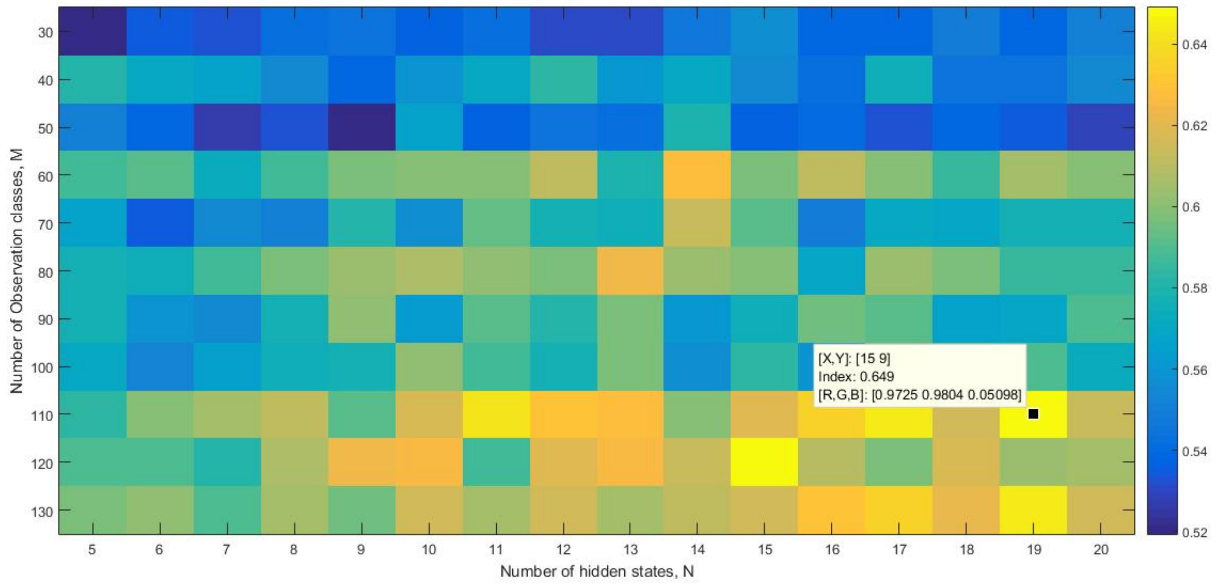

Figure 3. Cross validation results

2. Determine the hidden state variable α similar to the above training process using the model parameters A, B and Pi for each gesture.

3. Calculate the normalization factor c_t at each time step for each gesture model.

4. Prediction of the system is the model with the highest log likelihood given by,

$$log\left[P(O\mid\lambda)\right] = -\sum_{t=1}^{T} logc_t$$

5. The confidence was defined as the difference between the first and second log likelihood after normalizing all the likelihood with the maximum value.

4. Selecting Parameters

The algorithm uses a number of parameters such as N, number of hidden states and M, number of observation classes. Selecting the right number affects the performance of the system. Hence to determine an optimum value of N and M, cross validation was performed with N ranging from 5 to 20 in steps of 1 and M ranging from 30 to 130 insteps of 10.

For each pair of N and M LOOCV was performed by randomly choosing 12 train sets as test sets. The median confidence was then used to select the right parameters. Fig. 3 represents the cross validation results in the form of a color coded matrix, with yellow representing the maximum value. It was seen that the maximum confidence is obtained with N = 19 and M = 110.

Another parameter that affected the performance was how the model parameters was initialized. Two approaches were experimented

1. A was initialized as a left to right matrix, in which each state could propagate to only 1 other state. B was initialized to a uniform distribution matrix with columns summing to 1. π was initialized to first state.

$$A = \begin{bmatrix} \alpha_1 & 0 & \dots & 1-\alpha_N \\ 1-\alpha_1 & \alpha_2 & \dots & 0 \\ 0 & 1-\alpha_2 & \dots & 0 \\ \dots & \dots & \ddots & \dots \\ 0 & 0 & \dots & \alpha_N \end{bmatrix}$$

$$B = ones(N,M)\times 1/M$$

$$\pi = [1;0;0;\dots;0]$$

2. Another method used to initialize A, B and π is random initialization. There are initialized with a random value and the then normalized across columns such that

$$\sum_{j=1}^{N} A_{ij} = 1 \quad \sum_{j=1}^{N} B_{ij} = 1 \quad \sum_{i=1}^{N} Pi_i = 1$$

It was seen that the latter method produced more robust results with higher confidence compared to the former. The random initialization was used in this implementation.

5. Experiments and Results

The results for various testing sets is tabulated in the Table 1. The log likelihood graphs are shown in Fig. 4 to 11 each for individual test sets. It was seen that most of the test sets were classified correctly with high confidence. But one abnormality that was observed is that the result of test4.txt has a very low confidence and quite often misinterpreted as beat 4 over multiple runs. This is because of the random

Table 1. Predictions for all datasets

Test Set	P1	P2	P3	Confidence
test1.txt	wave	eight	inf	0.8724
test2.txt	beat 4	beat 3	wave	0.2039
test3.txt	inf	eight	beat 3	0.7516
test4.txt	beat 3	beat 4	wave	0.0316
test5.txt	circle	beat 4	beat 3	0.8422
test6.txt	inf	eight	beat 3	0.7920
test7.txt	eight	inf	beat 3	0.7592
test8.txt	beat 4	beat 3	wave	0.2423

P1:Prediction 1, P2:Prediction 2 and P3:Prediction 3

initialization. It is also affected by the training set chosen to train the model as seen in Fig. 12.

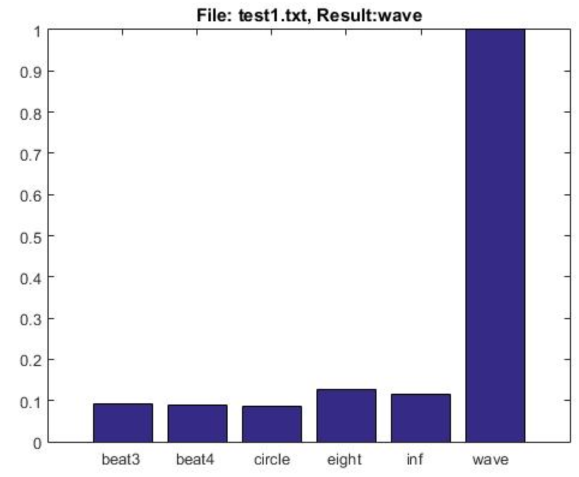

Figure 4. Log likelihood for test1.txt

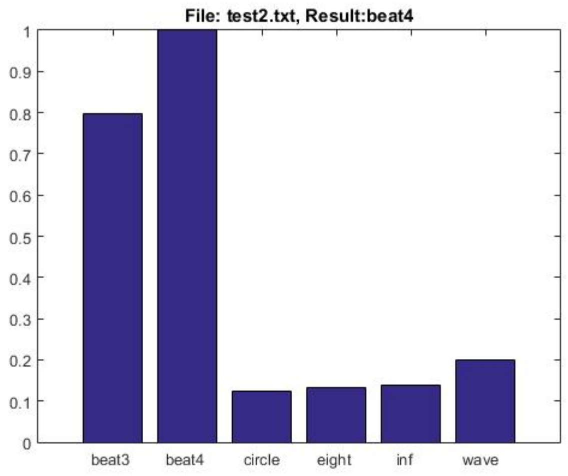

Figure 5. Log likelihood for test2.txt

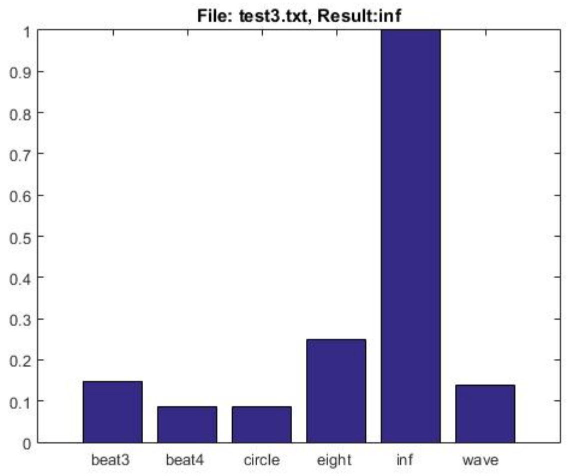

Figure 6. Log likelihood for test3.txt

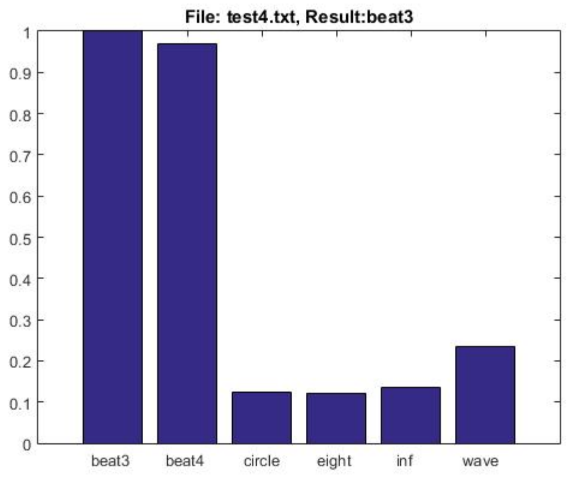

Figure 7. Log likelihood for test4.txt

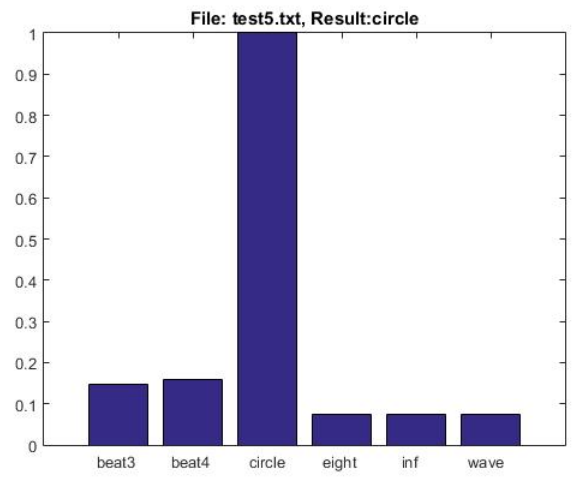

Figure 8. Log likelihood for test5.txt

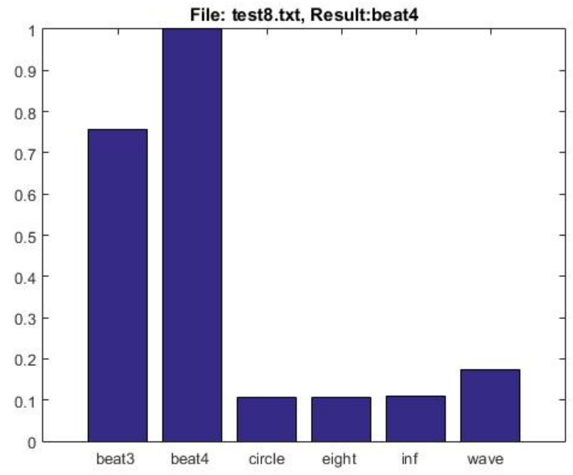

Figure 11. Log likelihood for test8.txt

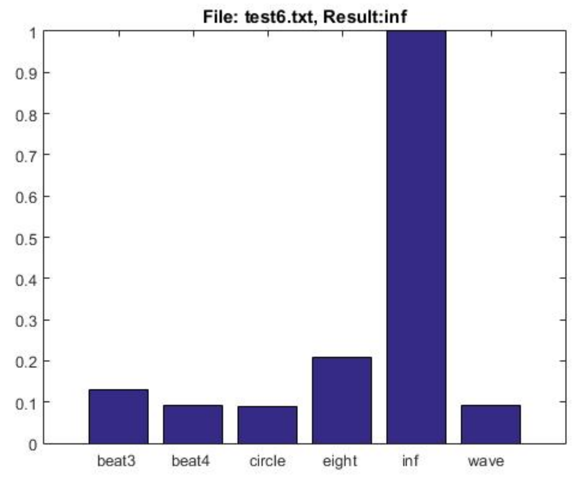

Figure 9. Log likelihood for test6.txt

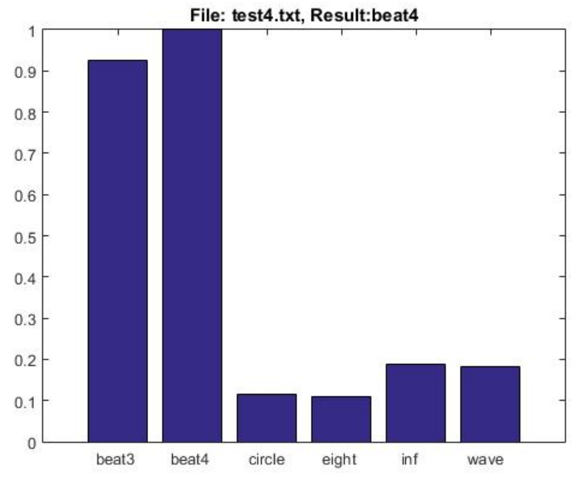

Figure 12. Log likelihood for test4.txt which was incorrectly classified on few runs

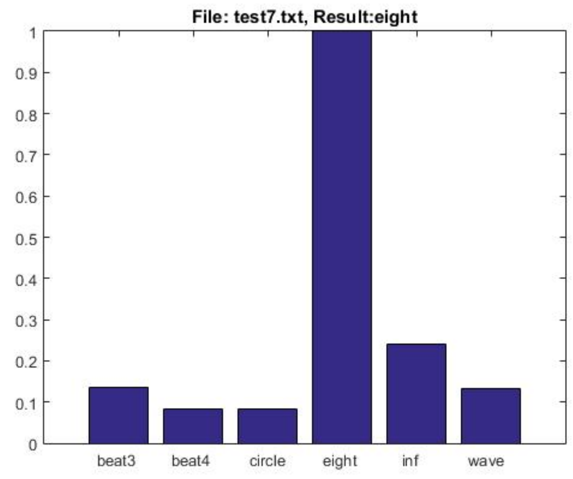

Figure 10. Log likelihood for test7.txt