Hidden Markov Model for Gesture Recognition

Daniel Pak

***Keywords*— Hidden Markov Model (HMM), Inertial Measurement Units (IMU), gesture recognition**

I. INTRODUCTION

Hidden Markov Model (HMM) is a model for guessing the states of a system over time with the assumption that the system is not subject to control input and only measurements of the states are observable. The Markov assumption is well represented by Fig. 1, which shows that the current state only depends on the previous state, and the current measurement depends only on the current state. These assumptions allow for large simplifications in calculations of the joint probabilities of states and measurements over time.

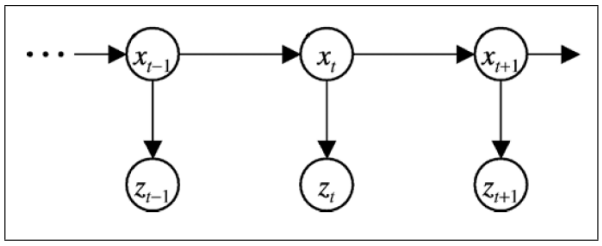

Fig. 1. HMM state/observation diagram

II. PROBLEM FORMULATION

The main challenge of this project is to recognize a set of arm gestures using recorded data from a 6-DOF IMU (accelerometer and gyroscope). There were 5 training sets for each gesture, all with several repetitions of a given gesture. There were also 2 extra training sets for each gesture, which seemed like a single repetition instead of multiple repetitions as in the main training set.

III. TECHNICAL APPROACH

A. Discretization of Observation Space

Since the IMU provided data in continuous space, they needed to be converted into values in discrete space to apply discrete HMM. The chosen method for discretization was scikit-learn's k-means algorithm with 30 means for each gesture.

Here are two notable details of implementation:

- In order to ensure all measurements were weighted equally in calculating the distance of each point from the means, I normalized the data such that the minimum values of both accelerometer and gyroscope measurements were both -1, and the max +1. More specifically, I had one normalizing value for all accelerometer measurements and another normalizing value for all gyroscope measurements, so not each axis of freedom was normalized from -1 to +1, but each type of measurement was normalized from -1 to +1.
- I also truncated the stationary parts at the beginning and end of each dataset to prevent them from having too much weight in the transition and emissions probabilities.

B. Baum-Welch Algorithm

After successful discretization of measurements, I used the Baum-Welch algorithm (EM algorithm for HMM) to calculate the model parameters for each gesture. The model parameters of interest were the prior (π), transition probabilities (T), and emission probabilities (B).

1) Initialize model parameters to be random values that normalize to 1 across the correct axes (depends on the definition of matrices and conditional probabilities). There are 30 observation classes as defined by the number of means in the k-means algorithm, and 10 state classes (an assumption made for the model).
2) (E-step) Calculate intermediate variables using the forward and backward procedures of the algorithm:

$$\alpha_t^{(k)}(i) = p(z_{0:t}, x_t = s_i)$$

$$\beta_t^{(k)}(i) = p(z_{t+1:T}|x_t = s_i)$$

where $t =$ time point, $k =$ specific dataset of a given gesture, $i =$ state class.
3) (E-step) In order to prevent underflow, multiply α and β at each time step by the normalizing constant before continuing to the next time step:

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

4) (E-step) Calculate the desired statistics of hidden states using α and β from the previous step:

$$\gamma_t^{(k)}(i) = p(x_t = s_i|z_{0:T})$$

$$\xi_t^{(k)}(i,j) = p(x_t = s_j, x_{t+1} = s_i|z_{0:T})$$

5) (M-step) Using γ, ξ, and observations, update the model parameters π, T, and B Note that the M-step is unaffected by the scaling terms because it all cancels out in the end (refer to Rabiner's paper for detail).
6) Repeat from step 2 until the maximum sum of absolute value of change in parameter values reaches below 0.01

C. Viterbi Decoding

Viterbi decoding is one way of calculating the joint probability of the most likely sequence of states and a set of observations. It can provide more information such as the actual sequence of hidden states, but we can stop at calculating the joint probabilities since we are interested in classifying the gesture (the classification will be based on maximum joint probability)

Once the model parameters are determined, the equation for Viterbi decoding can simply be applied and the joint probability can be calculated. One minor change was that I took the log of everything to prevent underflow, which should not adversely affect the forward pass or the classification since log is a monotonically increasing function. Only the forward pass is needed as previously mentioned, because the sequence of hidden states that give the maximum probability is unnecessary for classifying the gesture.

D. Classification of testset

Since there is a separate k-means model for each gesture, I calculated the discretization of the testset separately for each gesture. Then, I used those discretized values to calculate the joint probability of the most likely sequence of states and the observed measurements using Viterbi decoding. The final classification was the gesture with the highest joint probability.

IV. RESULTS

A. Classification Results

Classification results are displayed in Fig. 2. The models were trained using the original five datasets for each gesture, which had multiple repetitions of a given gesture. The files with numbers 31 and 32 were the extra training sets, which had only one repetition of a given gesture. Therefore, I considered those files (31 & 31) as validation sets. Of the 30 original training sets, 27 were classified correctly (90% accuracy). Of the 12 validation sets, 4 were classified correctly (33 % accuracy). Although I was unable to carefully analyze the reason for poor performance of validation sets, one reason may be the fact that the models were trained on multiple repetitions whereas the validation sets were all single repetition. Also, less repetitions means less observations and thus less information with which to classify the data.

The classification results for the testing set is displayed in Fig. 3 (multiple) and Fig. 4 (single). Since the real labels were not provided, I cannot assess its accuracy.

B. Visualization of model parameters

In order to visualize the model parameters, I produced heat maps for the transition and emissions probabilities and a plot for the prior. Examples of these results for one gesture are shown in Fig. 5, 6, and 7. These plots give some intuition to how the algorithm optimized the model parameters. The prior makes sense because the state is pretty consistent across datasets. The transition matrix makes sense because the state is mostly continuous and the states around a particular state are more likely to be in a similar state as the current state. The emission matrix is a bit harder to make a conclusion with the visualization.

V. DISCUSSION

There were a few notable realizations from implementing this algorithm. First was the threshold value for the stopping point of the Baum-Welch algorithm. With the threshold set at 0.01, the model performed significantly better than when the threshold was at 0.0001, although the exact percentage difference in the classification error was not recorded. The classification was also leaned heavily towards a specific gesture when the threshold was low, possibly implying that the model for one of the gestures was overfit to the entire dataset. Second was the common misclassification between beat3 and beat4 gestures, and inf and eight gestures. Since the two groups of motions have very similar general shapes, it is understandable that the algorithm had a difficult time distinguishing them. I would have liked to try various filtering and other discretization methods for the dataset such that the similar motions can be accurately distinguished.

There were also some changes to the algorithm that I would have liked to tested with more time. One was changing the number of states and observations defined by the initialization of transition and emission matrices. The number of discrete states and observations likely would sway the accuracy of the algorithm, and it would have been nice to plot the training and testing errors with systematic changes in these parameters. Second was the implementation of the k-means algorithm, which could have been applied to the entire dataset instead of having a separate k-means model for each gesture. This would have had the benefit of having less specific but more encompassing discretized values for the every gesture. It would have most likely been a bias-variance tradeoff with the k-means on entire dataset having higher bias and lower variance.

different N and M values Possibility of training k means differently (total data)

VI. REFERENCES

Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. Proceedings of the IEEE, 77(2), 257-286.

All other information was acquired from Unviersity of Pennsylvania's ESE650 lectures, discussions on Piazza, official websites of Python and the respective libraries.

{'beat3_01.txt': 'beat3',
'beat3_02.txt': 'beat3',
'beat3_03.txt': 'beat3',
'beat3_06.txt': 'beat3',
'beat3_08.txt': 'beat3',
'beat3_31.txt': 'beat4',
'beat3_32.txt': 'beat3',
'beat4_01.txt': 'beat4',
'beat4_03.txt': 'beat4',
'beat4_05.txt': 'beat4',
'beat4_08.txt': 'circle',
'beat4_09.txt': 'wave',
'beat4_31.txt': 'beat4',
'beat4_32.txt': 'wave',
'circle12.txt': 'circle',
'circle13.txt': 'circle',
'circle14.txt': 'circle',
'circle17.txt': 'circle',
'circle18.txt': 'circle',
'circle31.txt': 'wave',
'circle32.txt': 'circle',
'eight01.txt': 'eight',
'eight02.txt': 'eight',
'eight04.txt': 'eight',
'eight07.txt': 'eight',
'eight08.txt': 'eight',
'eight31.txt': 'eight',
'eight32.txt': 'wave',
'inf11.txt': 'inf',
'inf112.txt': 'inf',
'inf13.txt': 'inf',
'inf16.txt': 'inf',
'inf18.txt': 'inf',
'inf31.txt': 'eight',
'inf32.txt': 'eight',
'wave01.txt': 'circle',
'wave02.txt': 'wave',
'wave03.txt': 'wave',
'wave05.txt': 'wave',
'wave07.txt': 'wave',
'wave31.txt': 'beat3',
'wave32.txt': 'beat4'}

Fig. 2. Test results with training set

{'test9.txt': 'circle',
'test10.txt': 'circle',
'test6.txt': 'inf',
'test1.txt': 'circle',
'test7.txt': 'eight',
'test8.txt': 'circle',
'test12.txt': 'beat4',
'test2.txt': 'beat4',
'test5.txt': 'wave',
'test3.txt': 'eight',
'test4.txt': 'beat3',
'test11.txt': 'inf'}

Fig. 3. Test results with ”multiple” testing set

{'test9.txt': 'beat3',
'test10.txt': 'circle',
'test6.txt': 'eight',
'test1.txt': 'beat3',
'test7.txt': 'wave',
'test8.txt': 'wave',
'test12.txt': 'wave',
'test2.txt': 'beat4',
'test5.txt': 'beat4',
'test3.txt': 'eight',
'test4.txt': 'beat4',
'test11.txt': 'eight'}

Fig. 4. Test results with ”single” testing set

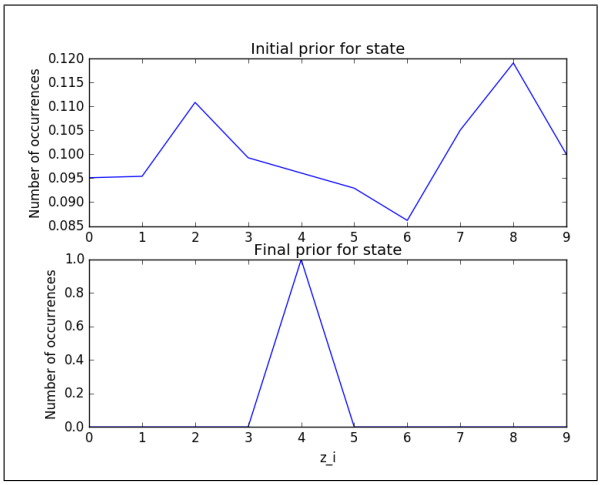

Fig. 5. Prior before and after Baum-Welch algorithm

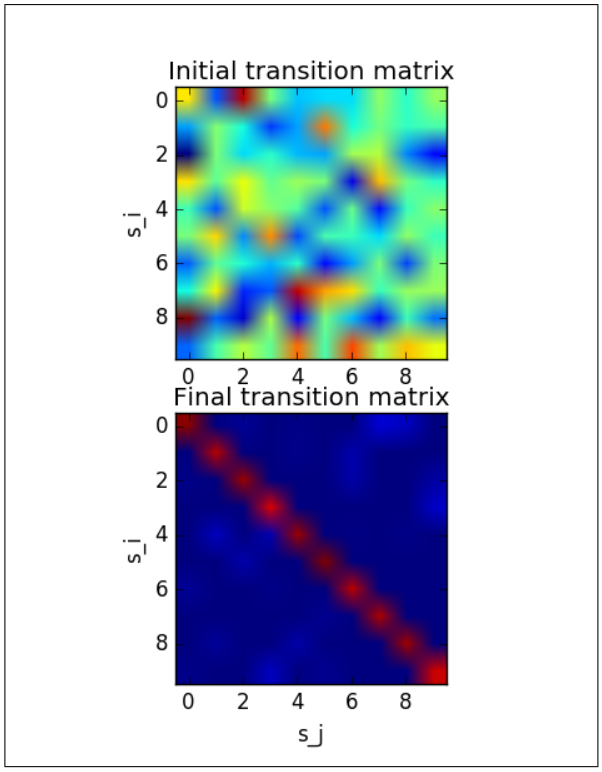

Fig. 6. Transition matrix before and after Baum-Welch algorithm

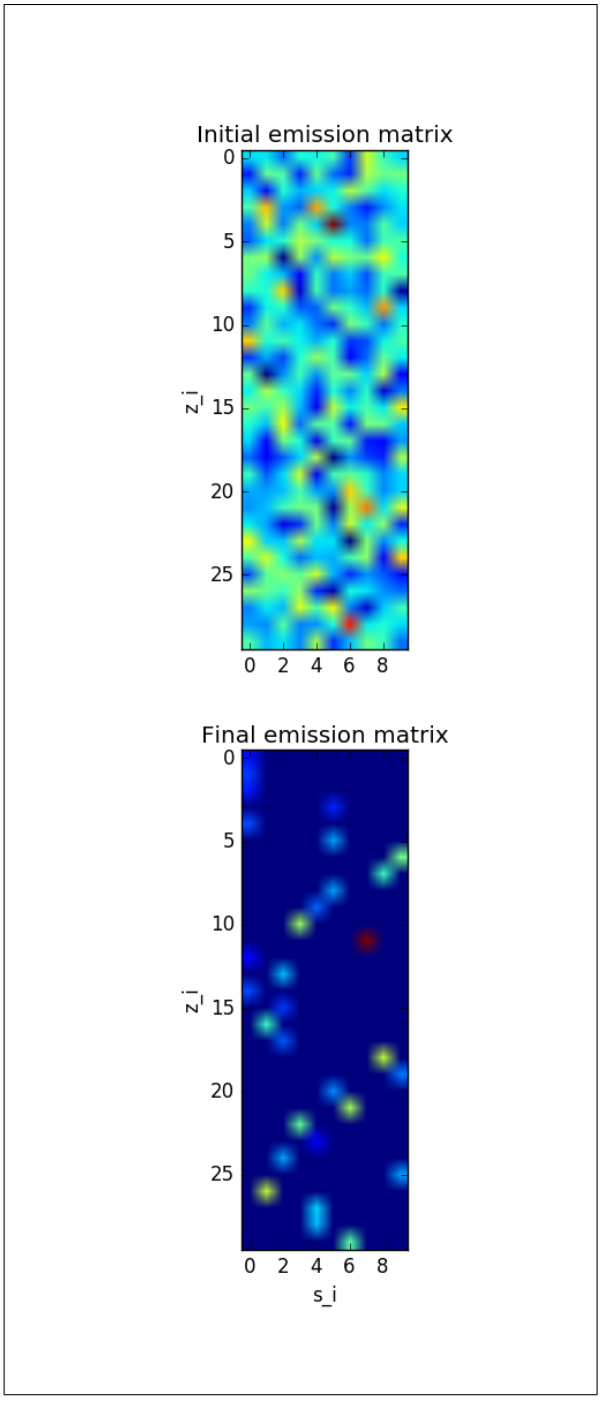

Fig. 7. Emission matrix before and after Baum-Welch algorithm