Time-Series Gesture Classification via Hidden Markov Models

Alif Ilham Madani

March 9, 2026

1 Introduction

This project explains the implementation of hidden markov models (HMMs) for gesture recognition tasks based on inertial measurement unit (IMU) data. The algorithm first quantizes the IMU vectors and trains independent HMM models for six different gestures (Wave, Infinity, Eight, Circle, Beat3, and Beat4). The model training uses the Baum-Welch algorithm [1] and is done in log-space to avoid the zero-probabilities issue in the implementation.

2 Problem Statement

The objective is to train HMM models that can classify discrete time-series sensor data. The input data consists of recordings with 7 columns: Time (ms), 3-axis Gyroscope (W_x, W_y, W_z in rad/sec), and 3-axis Accelerometer (A_x, A_y, A_z in m/s^2). The system must:

1. Preprocess and discretize the continuous 6-dimensional observation vectors.
2. Learn the hidden state transitions and observation emission probabilities for each gesture.
3. Compute the log-likelihood of unseen test sequences and classify them based on the maximum likelihood among all gesture models.

3 Technical Approach

3.1 Data Preprocessing and Vector Quantization

Before HMM training, the continuous raw sensor data was discretized. We remove the time column and cluster the remaining 6-dimensional feature vectors using the K-Means algorithm. We use the cluster IDs to transform the continuous observations into a discrete sequence of integers. Figure 1 shows the raw IMU data, while Figure 2 shows the discretized sequence.

Additionally, flat sensor readings at the beginning and end of the gesture recordings were trimmed by detecting periods where the initial and final cluster assignments remained static. Repeated gestures in the training set were then concatenated to form a continuous training sequence for each specific gesture class, as shown in Figure 3.

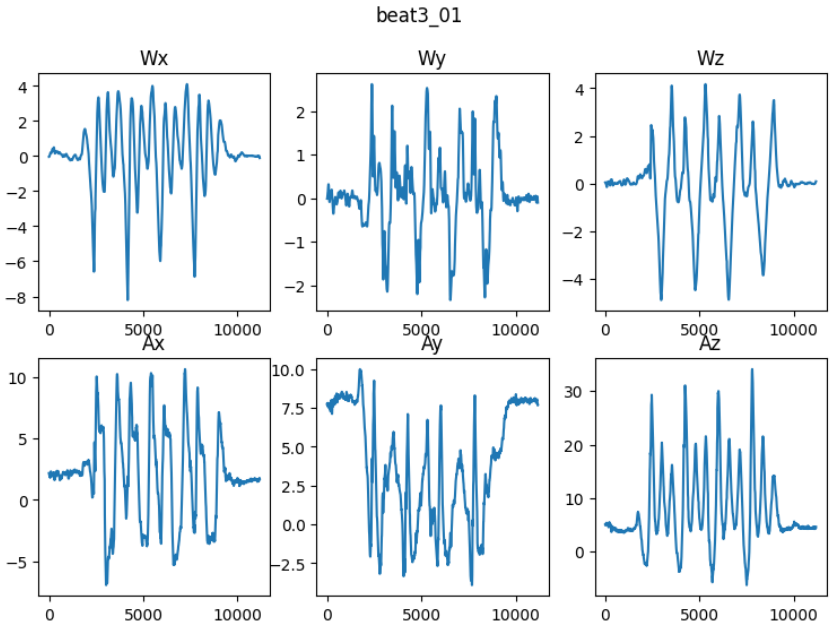


Figure 1: Raw IMU signals (6-DOF) for Beat3

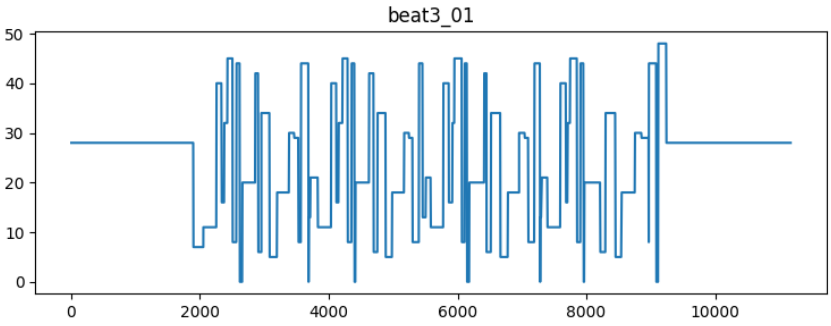


Figure 2: Discretized sequence ($K = 50$)

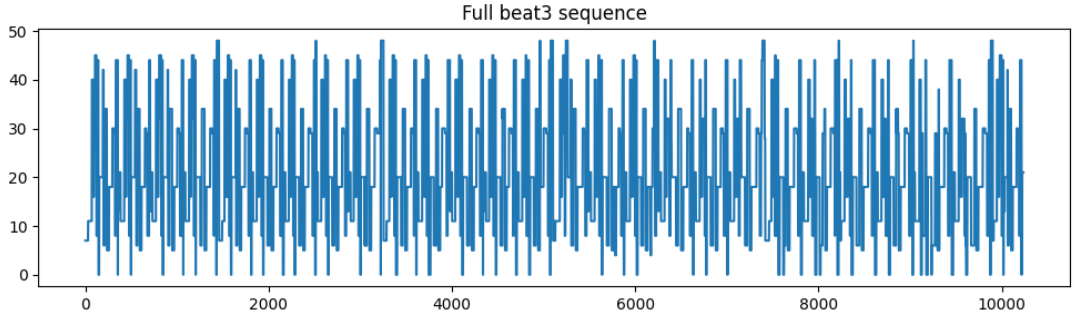


Figure 3: Concatenated sequences of Beat3

3.2 Hidden Markov Model Architecture

3.2.1 Left-to-Right Topology with Loopback

To capture the sequential nature of a gesture and simplify the training, we use a left-to-right transition structure [1], combined with a loopback mechanism to support repeated continuous gestures in the training data.

- **Initial State ($\boldsymbol{\pi}$):** The model strictly begins in state 0. Thus, $\pi_0 = 1.0$ and $\pi_i = \epsilon$ for $i > 0$.
- **State Transition Matrix (A):** Each state i has a 50% probability of remaining in state i and a 50% probability of transitioning to state $i+1$. The final state $N-1$ has a 50% probability of looping back to state 0.
- **Emission Matrix (B):** Initialized randomly and normalized across all M observation clusters.

3.3 Training HMM with Baum-Welch (EM Algorithm)

We train the model parameters $(\mathbf{A}, \mathbf{B}, \boldsymbol{\pi})$ directly from the discretized training sequences using 10 iterations of the Baum-Welch algorithm.

3.3.1 Log-Space Calculations for Underflow Prevention

Multiplying a long sequence of probabilities rapidly leads to numerical underflow. Consequently, all Expectation-Maximization (EM) steps—including the Forward (α) and Backward (β) algorithms—were executed entirely in log-space. We heavily utilized the `logsumexp` trick [2] for numerical stability when marginalizing over states:

$$\ln\left(\sum_i \exp(x_i)\right) \approx \max_i(x_i) + \ln\left(\sum_i \exp(x_i - \max_i(x_i))\right) \tag{1}$$

3.3.2 Zero Emission Probability Handling

During the maximization step, if a particular observation o_t is never seen in the training data for a state, its standard calculated emission probability becomes exactly 0. During inference, this would force the log-likelihood of the entire sequence to $-\infty$, preventing meaningful comparison between models. To correct this, we set a minimum emission threshold $\epsilon = 10^{-20}$. This adjusts the model to treat unseen observations as highly unlikely rather than strictly impossible.

3.4 Hyperparameter Selection

Our provided data consists of two distinct formats: a primary dataset containing continuous, repeated gesture executions and an additional dataset containing exactly one single gesture motion. We utilized a holdout validation strategy. The models were trained on the primary dataset containing repeated sequences per gesture and validated on the additional dataset. This mechanism tests the model's ability to generalize from a multiple-gesture distribution to a single-gesture data distribution, similar to the test data.

We performed a grid search evaluating $M \in \{50, 75, 100\}$ and $N \in \{10, 15, 20\}$. As shown in Table 1, multiple configurations achieved perfect classification accuracy on the holdout validation set. To differentiate these configurations, we evaluated the minimum, average, and maximum log-likelihood assigned to the true gesture classes. The results showed that $N = 15$ hidden states and $M = 50$ observation clusters provided the most optimal fit (highest average true log-likelihood of -344.16) without overfitting.

Clusters (M)	**States (N)**	**Accuracy**	**Min True LL**	**Avg True LL**	**Max True LL**
50	15	1.000	-508.73	-344.16	-159.75
75	15	1.000	-890.01	-745.46	-570.49
75	10	1.000	-1006.77	-787.28	-601.14
100	20	1.000	-3275.96	-1416.97	-657.05
100	15	1.000	-3359.38	-1452.77	-758.69
100	10	1.000	-3397.57	-1461.67	-763.03
50	20	0.833	-511.83	-376.80	-159.34
50	10	0.833	-639.25	-459.78	-239.80
75	20	0.833	-1056.66	-801.64	-611.35

Table 1: Hyperparameter grid search results utilizing holdout validation.

After determining the optimal hyperparameters ($N = 15$, $M = 50$), the final models were trained using both the primary and additional training datasets. This ensures the models see all data distributions available in the training.

3.5 Inference (Classification)

To classify an unknown test sequence $\mathbf{O} = \{o_1, o_2, \ldots, o_T\}$, we compute its log-likelihood given each gesture model λ_k using the termination step of the Forward algorithm:

$$\ln P(\mathbf{O}|\lambda_k) = \text{logsumexp}_{i=1}^{N}(\alpha_T(i)) \tag{2}$$

The predicted gesture $\hat{y}$ is the class that yields the maximum log-likelihood:

$$\hat{y} = \underset{k}{\text{argmax}} \ln P(\mathbf{O}|\lambda_k) \tag{3}$$

3.6 Implementation Overview

The algorithm pipeline is summarized below:

Algorithm 1 Gesture Training and Inference Pipeline

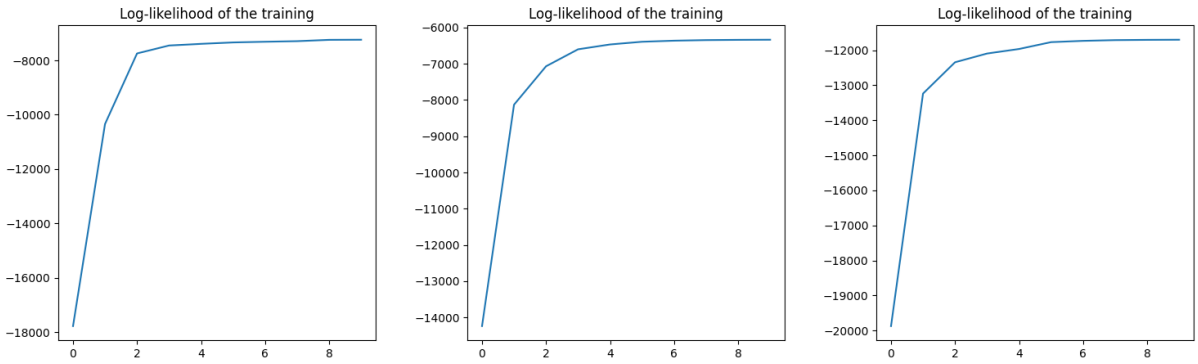
procedure TRAIN($RawData$)
    Extract $X \leftarrow$ 6-DOF IMU readings from $RawData$.
    Fit K-Means on X to get discrete clusters.
    for each gesture class $k \in \{Wave, Inf, Eight, Circle, Beat3, Beat4\}$ **do**
        $Seq_k \leftarrow$ TrimIdleStates(Cluster IDs for k)
        Initialize $\lambda_k = (\mathbf{A}_{N\times N}, \mathbf{B}_{N\times M}, \boldsymbol{\pi}_N)$ in log-space.
        for iteration $= 1 \dots 10$ **do**
            Compute α (Forward) and β (Backward) in log-space.
            Update $\mathbf{A}, \mathbf{B}, \boldsymbol{\pi}$ using EM.
            Replace exactly 0 probabilities in $\mathbf{B}$ with $\ln(\epsilon)$.
        end for
        Save trained λ_k.
    end for
end procedure

procedure INFERENCE($TestSequence$)
    $Seq \leftarrow$ Predict K-Means clusters for $TestSequence$.
    $Seq \leftarrow$ TrimIdleStates(Seq)
    for each $\lambda_k \in TrainedModels$ **do**
        $LL_k \leftarrow$ ForwardAlgorithm(Seq, λ_k)
    end for
    Sort LL descending.
    return Top 3 Predictions.
end procedure

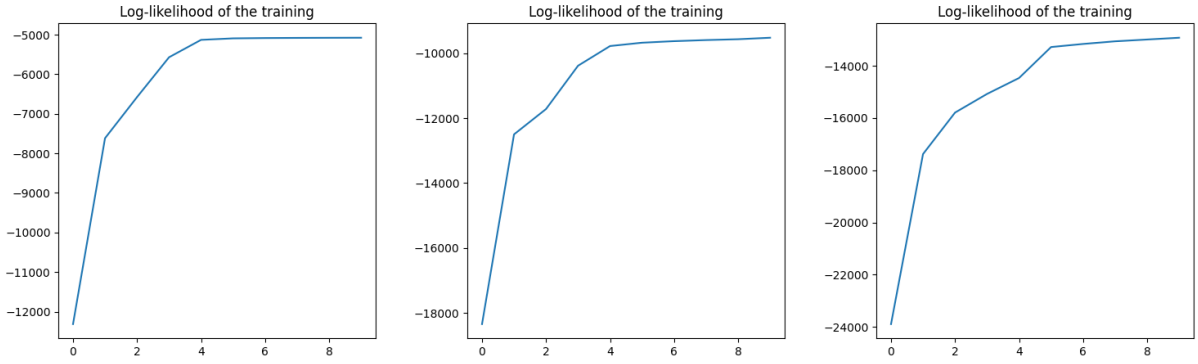
4 Results and Discussion

4.1 Training Convergence

The Baum-Welch algorithm was run for 10 iterations for each of the six gesture models. The log-likelihood of the training sequence increased with each epoch before plateauing, indicating model convergence. Figure 4 illustrates the training progression for all six gesture models.



(a) Wave (b) Infinity (c) Eight



(d) Circle (e) Beat3 (f) Beat4

Figure 4: Log-likelihood per epoch during Baum-Welch training for each gesture model.

4.2 Performance on Test Dataset

The trained models were applied to the unseen test dataset. Because the test data labels were unknown, performance is evaluated based on the log-likelihood of each model. Table 2 summarizes the classification outcomes, displaying the top three model predictions and their respective log-likelihoods for each test sequence.

Test File	**Prediction 1 (LL)**	**Prediction 2 (LL)**	**Prediction 3 (LL)**
test1	inf (−336.78)	eight (−6908.09)	beat3 (−15036.63)
test2	beat3 (−858.14)	beat4 (−1212.80)	inf (−13712.61)
test3	inf (−335.69)	eight (−7991.22)	beat3 (−17935.20)
test4	beat4 (−631.14)	beat3 (−1324.55)	inf (−14878.43)
test5	circle (−203.51)	beat4 (−10372.31)	eight (−12480.01)
test6	eight (−376.64)	inf (−8102.34)	wave (−13221.58)
test7	wave (−279.92)	inf (−8212.69)	eight (−8755.47)
test8	beat4 (−845.86)	beat3 (−1870.73)	inf (−15193.20)

Table 2: Inference results on the test dataset showing the top 3 gesture predictions.

4.3 Discussion and Limitation Analysis

We observed that structurally distinct gestures exhibited massive log-likelihood differences. For instance, in `test1`, the model's top prediction (`inf`) had a log-likelihood over 6,500 points higher than the second choice. Conversely, for `test2`, `test4`, and `test8`, the margin between `beat3` and `beat4` was much narrower (under 1,100 points). These smaller differences means that `beat3` and `beat4` have similar sub-motions, resulting in closely overlapping hidden state emissions.

By relying on K-Means to discretize the continuous 6-dimension IMU space, we drop spatial information. If a user performs a gesture with a slightly different amplitude, it may fall into a neighboring K-Means cluster that has a near-zero emission probability in the trained HMM. We can consider training Continuous Gaussian Mixture Models (GMM-HMMs) although this will increase the complexity.

The log-likelihood is a sum of log-probabilities over time. Consequently, longer gestures yield lower absolute log-likelihoods. While this does not affect the comparison across models for the same test sequence, we need to normalize the log-likelihoods to compare different sequences.

References

[1] L. 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.

[2] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, *et al.*, "SciPy 1.0: fundamental algorithms for scientific computing in python," *Nature methods*, vol. 17, no. 3, pp. 261–272, 2020.