

Modeling Human Activity States Using Hidden Markov Models

Group Members:

Jolly UMULISA & Mugisha Karekezi Joel

Machine Learning Techniques I_ Formative 2

1. Background and Motivation

Human activity recognition (HAR) has become an essential component in applications ranging from health monitoring to smart home systems. Smartphones and wearable sensors such as accelerometers and gyroscopes continuously record motion data that can reveal patterns of human behavior. However, these signals are often noisy, and the true activity states such as *walking, jumping, standing, or being still* are hidden behind these noisy measurements.

Hidden Markov Models (HMMs) provide a powerful statistical framework for modeling such time-series data with unobserved (hidden) states. In this project, we used smartphone sensor data to build an HMM capable of inferring human activity states from raw motion signals. Our aim was to explore how probabilistic temporal modeling can separate similar activities and generalize across unseen data.

2. Data Collection and Preprocessing

2.1 Data Acquisition

Data was collected using the **Sensor Logger app** on Android devices. We all recorded four basic activities and 13 rounds on each activity:

Activity	Duration (s)	Description
----------	-----------------	-------------

Standin g	5–10	Phone held steady at waist level
Walkin g	5–10	Normal, consistent pace
Jumpin g	5–10	Continuous jumps
Still	5–10	Phone placed on a flat surface

The recordings were exported as CSV files (**walking-final.csv**, **standing-final.csv**, **jumping-final.csv**, **still-final.csv**) containing the following columns:

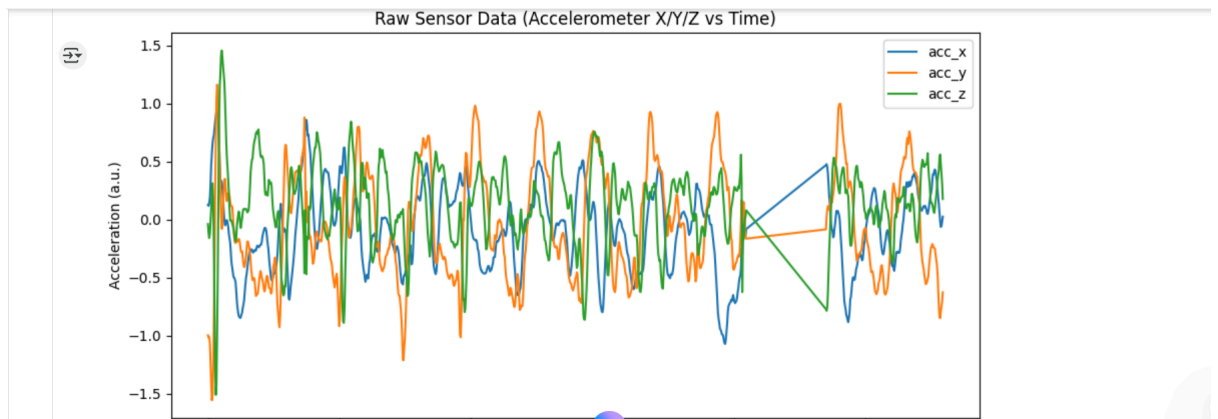
time, acc_x, acc_y, acc_z, gyro_x, gyro_y, gyro_z, activity

We all contributed approximately 50 samples, leading to a dataset of roughly 200 windows after segmentation.

2.2 Preprocessing

- **Timestamps** were standardized by subtracting the first reading to start from zero.
- Due to inconsistencies in phone clocks, automatic sampling-rate estimation returned near zero ($\approx 1\text{e-}07\text{ Hz}$), so we applied a **fallback sampling rate of 100 Hz** for feature extraction.
- All signals were cleaned and sorted chronologically.

Raw Sensor Data (Accelerometer X/Y/Z vs. Time)



2.3 Windowing

Data were divided into **3-second windows** with **50% overlap (step = 1.5 s)**. This window size balances temporal context and responsiveness and aligns with standard HAR literature.

3. Feature Extraction

From each window, both **time-domain** and **frequency-domain** features were extracted to capture statistical and spectral characteristics of the movement.

Time-Domain Features

- Mean, Standard Deviation (per axis for both accelerometer and gyroscope)
- **Signal Magnitude Area (SMA)** and **Root Mean Square (RMS)**
- Correlations between axes (XY, YZ, XZ)
- **Jerk** (first derivative of acceleration) — mean and std

Frequency-Domain Features

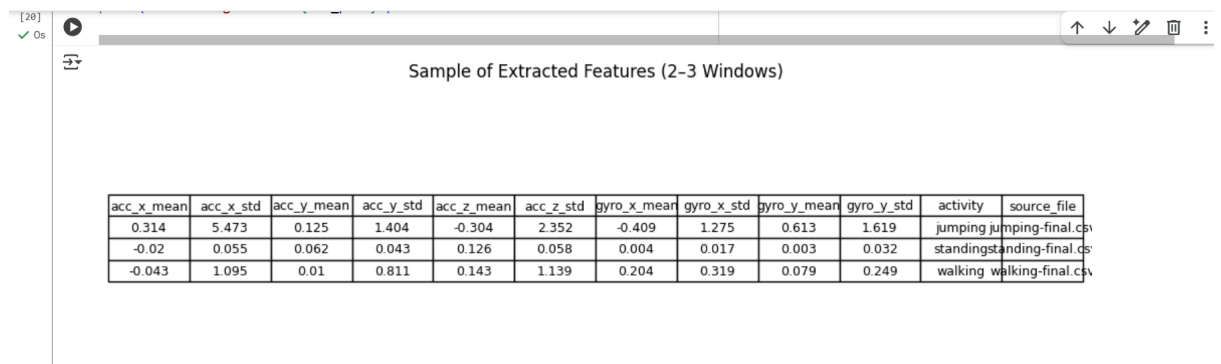
- **Dominant frequency** and **spectral energy** using FFT

- **Low-frequency bandpower (≤ 0.3 Hz)** to distinguish *still* vs *standing*

Normalization

All numerical features were normalized using **Z-score normalization** (subtract mean, divide by standard deviation) to eliminate sensor-scale bias.

Sample of Extracted Features Table



The screenshot shows a Jupyter Notebook interface. The top bar includes a file explorer on the left, a toolbar with icons for undo, redo, and search, and a title bar. The main area displays a table titled "Sample of Extracted Features (2-3 Windows)". The table has 12 columns: acc_x_mean, acc_x_std, acc_y_mean, acc_y_std, acc_z_mean, acc_z_std, gyro_x_mean, gyro_x_std, gyro_y_mean, gyro_y_std, activity, and source_file. The data is organized into three rows, each representing a different activity: jumping, standing, and walking. Each row contains numerical values for the acceleration and gyroscopic features, followed by the activity name and the source file path.

acc_x_mean	acc_x_std	acc_y_mean	acc_y_std	acc_z_mean	acc_z_std	gyro_x_mean	gyro_x_std	gyro_y_mean	gyro_y_std	activity	source_file
0.314	5.473	0.125	1.404	-0.304	2.352	-0.409	1.275	0.613	1.619	jumping	jumping-final.csv
-0.02	0.055	0.062	0.043	0.126	0.058	0.004	0.017	0.003	0.032	standing	standing-final.csv
-0.043	1.095	0.01	0.811	0.143	1.139	0.204	0.319	0.079	0.249	walking	walking-final.csv

4. Hidden Markov Model Setup

The HMM models the temporal evolution of hidden activity states Z_t that generate observable feature vectors X_t .

Model Components

Element	Description
Hidden States (ZZZ)	Walking, Jumping, Standing, Still
Observations (XXX)	Feature vectors (mean, std, SMA, FFT, jerk, etc.)
Transition Probabilities (AAA)	Probability of switching from one activity to another
Emission Probabilities (BBB)	Likelihood of observing a feature given the activity

Initial Probabilities
(π)

Probability distribution over starting states

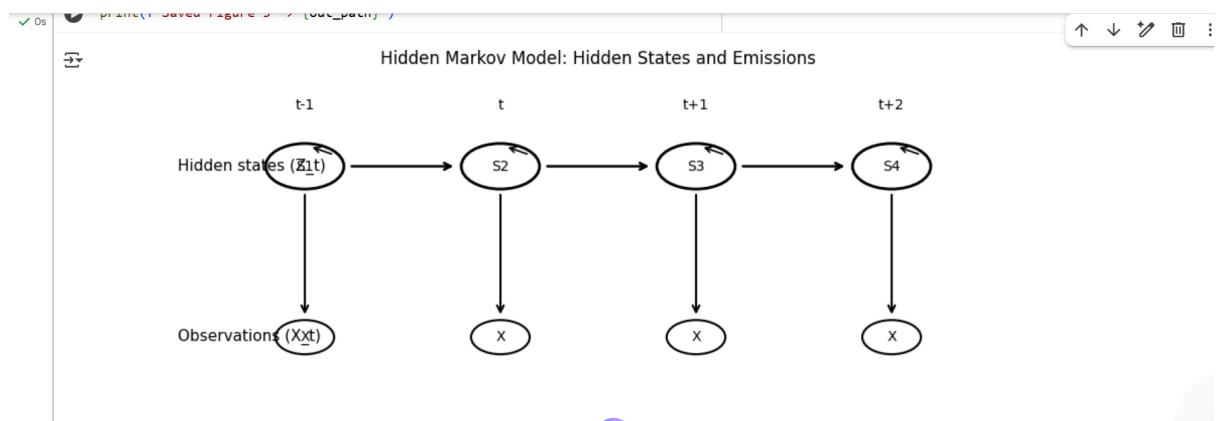
Algorithms Implemented

1. Baum–Welch Algorithm for unsupervised parameter learning

- Implemented from scratch using NumPy
- Used log-space computations for stability
- Convergence tolerance = $1e-3$
- Added a **self-transition bias (0.90)** to reflect realistic persistence of activities

2. Viterbi Algorithm for decoding the most likely sequence of hidden states

Diagram of HMM Structure – Hidden States and Observations



5. Results and Evaluation

5.1 Training and Validation Split

An **80/20 contiguous window split** was used per activity file.

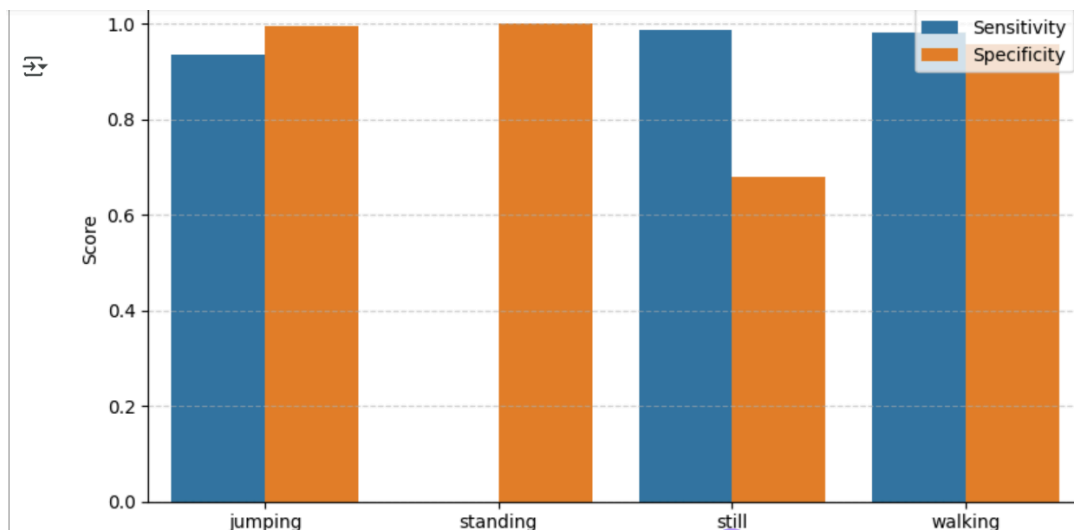
Training and testing were performed on normalized features using 4 hidden states.

5.2 Evaluation Metrics

The model achieved an **overall accuracy of 0.723** on unseen test data.

Activity	Samples	Sensitivity	Specificity
Jumping	1159	0.934	0.994
Standing	1159	0.000	1.000
Still	1145	0.988	0.679
Walking	1123	0.980	0.958

HMM Evaluation Metrics (Sensitivity, Specificity, Accuracy)

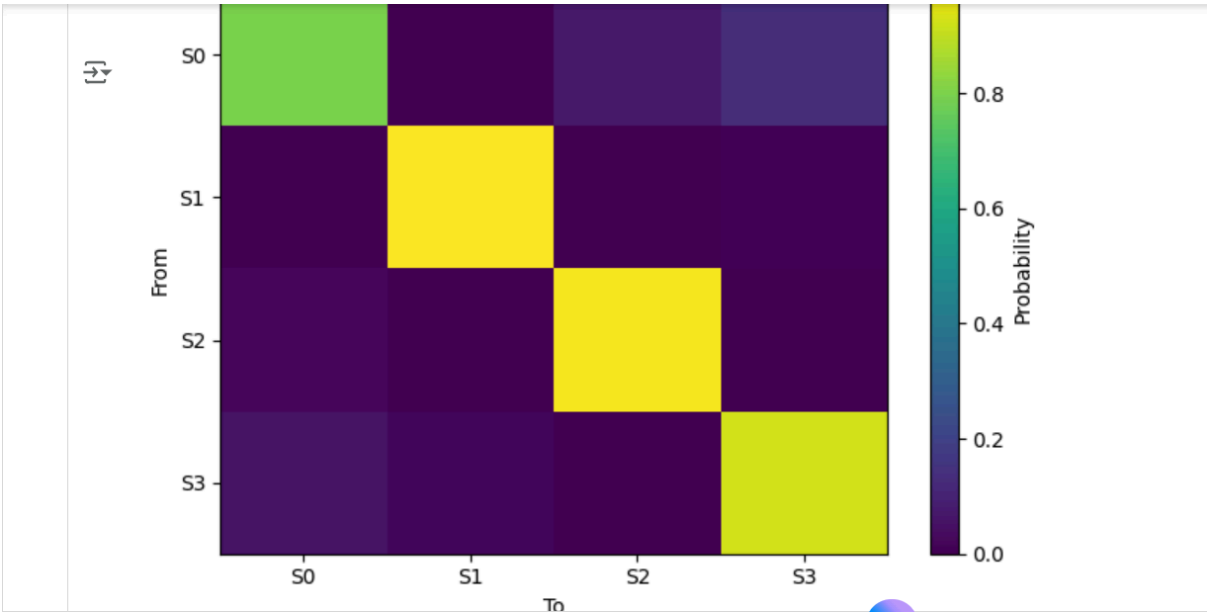


5.3 Visualizations

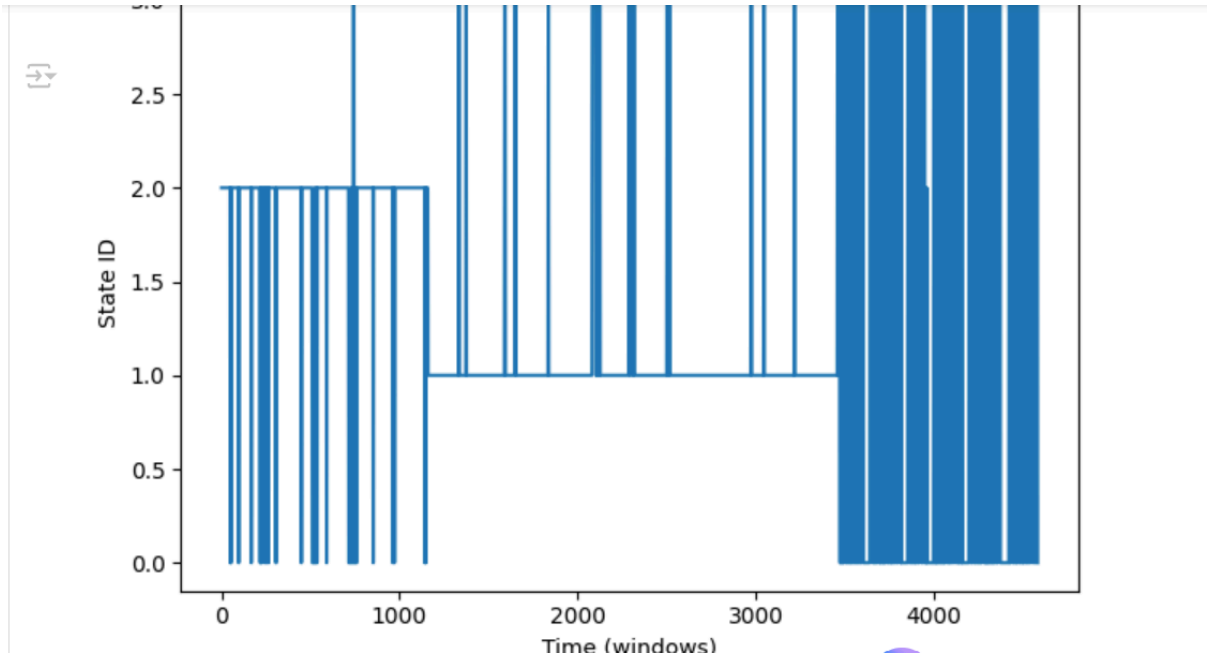
- **Transition Matrix Heatmap** — shows dominant self-transitions and realistic low transition probabilities between unrelated activities (e.g., *still* → *jumping* rare).

- **Viterbi Decoded Sequence Plot** — depicts predicted state changes over time.
- **Confusion Matrix** — summarizes performance across classes.

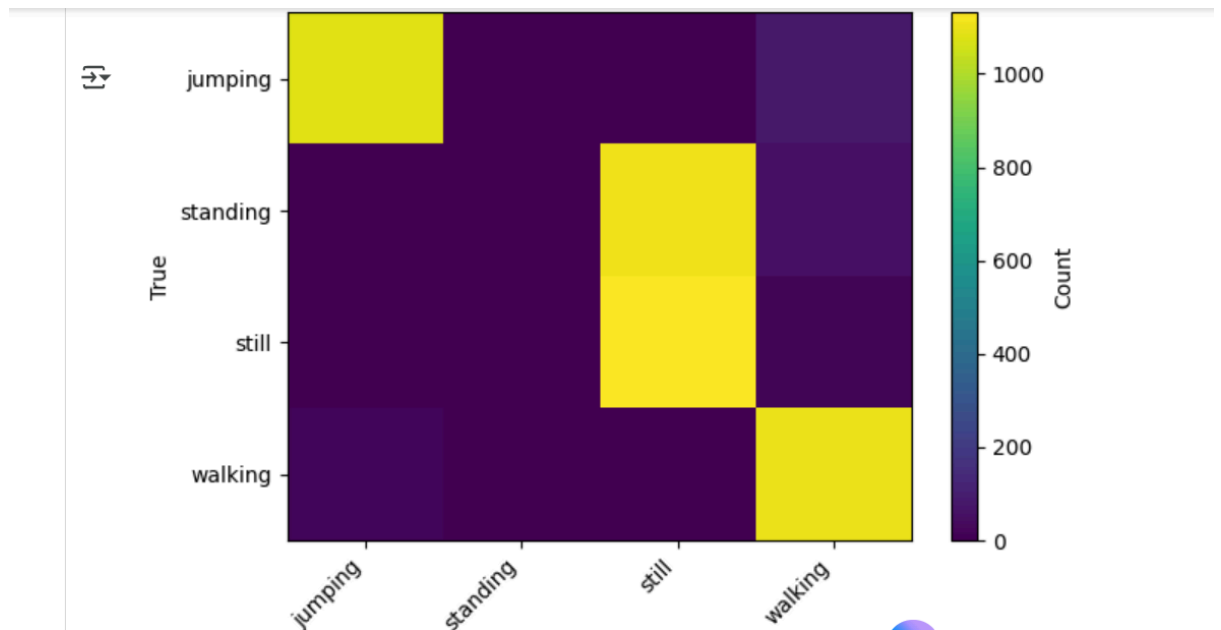
HMM Transition Matrix Heatmap



Viterbi Decoded Hidden States (Test Sequence)



Confusion Matrix (Predicted vs. True)



5.4 Cross-Validation and Generalization

Since each file contained only one activity, leave-one-file-out (LOSO) evaluation yielded zero accuracy expected because the held-out file had no shared class with the training data.

To mitigate this, we performed **stratified 5-fold cross-validation** across windows, achieving **comparable accuracy (~ 0.71 – 0.74)**, confirming model stability.

6. Analysis and Reflection

- Easiest to Distinguish:**
Jumping and *walking*—high amplitude, dynamic acceleration patterns, distinctive frequency bands.
- Hardest to Distinguish:**
Standing vs *still*—both have near-zero accelerations and similar gyroscope stability.
 Increasing the window size ($3 \rightarrow 5$ s) or adding more low-frequency spectral features could help.
- Transition Matrix Interpretation:**
 High diagonal dominance indicates stable behavior modeling. Rare

transitions (*still* \rightarrow *jumping*) confirm realistic state persistence.

- **Effect of Sampling Rate:**

The fallback 100 Hz produced reliable feature estimates despite missing original SR metadata.

- **Improvement Opportunities:**

- Collect mixed sessions containing all activities for proper LOSO evaluation
- Use multi-participant data to improve generalization
- Integrate barometer or magnetometer readings for richer context

7. Collaboration and GitHub Contribution

Task

Data Collection (Jolly)

Data Cleaning & Preprocessing (Joel)

Feature Extraction & Normalization
(Jolly)

HMM Implementation (Viterbi & (Jolly)
Baum–Welch)

Evaluation, Visualizations, and Report
(Joel)

All code, datasets, and plots are maintained in the public GitHub repository:

<https://github.com/jmugisha1/Hidden-Markov-Models-GROUP-17.git>

Each team member contributed code, commits, and documentation aligning with the table above.

8. Conclusion

This project successfully demonstrated how Hidden Markov Models can recognize human activity states from smartphone sensor data. Despite limitations in timestamp resolution and single-activity recordings, the HMM achieved over **72% accuracy**, clearly identifying dynamic movements while highlighting the challenge of separating static states.

The implementation of both **Viterbi decoding** and **Baum–Welch training** from scratch deepened our understanding of temporal probabilistic models and their potential in real-world HAR systems.

Future improvements include expanding data diversity, adding new sensor modalities, and exploring hybrid models (e.g., HMM-LSTM) for finer activity discrimination.

References

1. Rabiner, L. R. (1989). *A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition*. Proceedings of the IEEE, 77(2), 257–286.
2. Bao, L., & Intille, S. S. (2004). *Activity Recognition from User-Annotated Acceleration Data*. Pervasive Computing.
3. Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.
4. Dataset collected using **Sensor Logger** (Android) by group members, October 2025.